# Tech Spec - M-Pin v3.3.0 PIN Pad

## Contents

# General

This document describes the technical details of the M-Pin v3.3.0 Browser Client - the PIN Pad. It is targeted at Technical Staff - Developers, DevOps and QA personnel.

# Acronyms

| Acronym | Description |
|---|---|
| RPS | Relying Party Service |
| RPA | Relying Party Application |
| D-TA | Distributed Trusted Authority |
| ACL | Access Control List |
| OTP | One-Time Password |
| PIN | Personal Identification Number |

# References

| Title | Link |
|---|---|
| SASS | http://sass-lang.com/ |
| Grunt | http://gruntjs.com/ |
| Handlebars | http://handlebarsjs.com/ |

# System Overview

The M-Pin System consists of two groups of Services - Customer-hosted Services and CertiVox-hosted Services.

The third, but no less important component, is the Client. Currently there are two clients available - the Browser Client, also called the PIN Pad, and the Mobile Client, known as the Mobile App

## Customer-hosted Services

- **M-Pin Server** - The Authentication Server against which End-Users authenticate. The Client (PIN Pad) performs the "M-Pin Protocol" against the M-Pin server in order to authenticate an End-User.
- **D-TA** - Distributed Trusted Authority Service, managed by the Customer. Responsible for generating Client and Server Secret Shares as well as Time Permit Shares.
- **RPS** - Relying Party Service. Implements the M-Pin Protocol and Workflows on behalf of the Customer. The RPS serves as an abstraction layer between the specific implementation of the M-Pin Protocol and the RPA.
- **RPA** - Relying Party Application. This is the Application to which end-users are authenticated through the M-Pin System. This application is implemented/managed by the Customer and is strictly specific for each different Customer.

## CertiVox-hosted Services

- **D-TA** - Distributed Trusted Authority Service, managed by CertiVox. Responsible for generating Client and Server Secret Shares as well as Time Permit Shares.
- **D-TA Proxy** - Proxies request to the D-TA, validating RPS signatures. The D-TA Proxy is public-facing, while the D-TA should not be publicly accessible.
- **Time Permits Service** - A service responsible to publish Time Permits to an online storage (CDN), such as AWS S3.
- **Registration Service** - A service that handles new Customer registration.

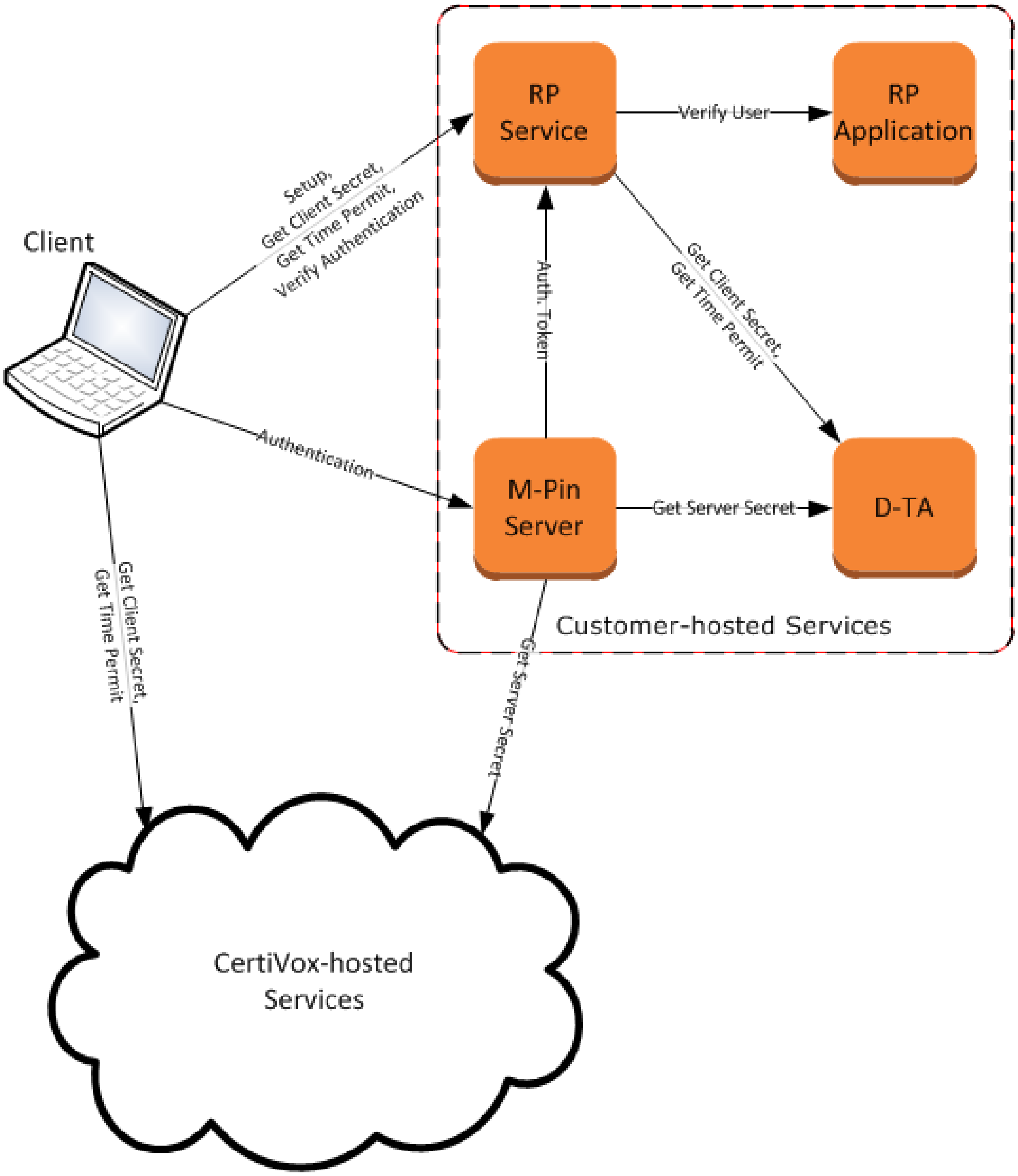
## PIN Pad

The PIN Pad is a JavaScript software component that should be integrated into the Customer's Application Web Page. The PIN Pad encapsulates all the operations and logic that needs to be performed at the front-end, in order to register and authenticate an end-user.

## Mobile App

The Mobile App is a JavaScript application, much similar to the PIN Pad. The Mobile App also carries out the operations needed to register and authenticate an end-user, but the user is authenticated to a browser session, rather than to a session on the mobile device.

The below diagram describes the main M-Pin system components and the interaction between them:



Client
Setup,
Get Client Secret,
Get Time Permit,
Verify Authentication
RP Service
Verify User
RP Application
Auth. Token
Get Client Secret,
Get Time Permit
Authentication
M-Pin Server
Get Server Secret
D-TA
Customer-hosted Services
Get Client Secret,
Get Time Permit
Get Server Secret
CertiVox-hosted Services

# System Architecture

The PIN Pad is a software component written in JavaScript and intended to be embedded into a Web Page. It is designed in a way that minimizes the effort required by the integrator, while leaving a wide set of customization options.

The PIN Pad uses a very few 3rd party libraries, which minimizes the risk of incompatibilities with such libraries used by the Customer Web Site.

## Main modules and files

### mpin.js

This is the main module of the PIN Pad, which "uses" the other modules. This is the only file that the Customer's HTML page should include in order to embed the PIN Pad. This module takes care of rendering the different pages of the PIN Pad, i.e. those pages that are specific to the Browser Client.

### mpin-all.js

This module encapsulates the logic and the flow of the M-Pin Protocol and is responsible for communicating with the back-end M-Pin Services. It is a generic module used by both the PIN Pad and the Mobile App. It is used by the mpin.js to handle the M-Pin Protocol communication. This file is automatically generated during the Build Process from the files listed in `build/mpin_deplist`. It is then concatenated to the mpin.js to form the *run-time mpin.js* file. For more details see the Build Process.

### templates.js

This is automatically generated file which includes the parameterized HTML templates in the form of JavaScript code. The file is generated during the Build Process and is afterwards concatenated to the mpin.js to form the *run-time mpin.js* file. For more details see the Build Process.

### Gruntfile.js

This is the file according which the Build Process is executed. It describes all the dependencies and actions that need to be taken during this process.

### settings.json

This is the file which specifies the actual values to be used for various parameters, used in parameterized files. During the Build Process the parameters are replaced with the values specified in this file.

**NOTE** that this file doesn't exist in that name in the source code repository. You can rename the file *setting.json_build* to *settings.json*, and set the appropriate values in it.

# Code Structure

The code is separated into several folders for easy management and templates customization, as will be explained further in this document.

## The "browser" Folder

This is the "home" folder for all the Browser PIN Pad code. It contains some files that are general for the whole application:

- index.html - sample main HTML page.
- Gruntfile.js - the main file that defines the rules for the Grunt operation (see Using Grunt)
- settings.json - the file that specifies the actual values to be used for various parameters.

## The "browser/js" Folder

This folder contains the main JavaScript file - mpin.js. During the  Build Process parameters in mpin.js are replaced with their actual values, according settings.json. Afterwards, this file is concatenated with mpin-all.js, templates.js and the 3rd part handlebars.runtime.min.js to form the *run-time mpin.js* file.

## The "browser/images" Folder

This folder contains resources for the visualization of the application. The resources are placed into a sub-folders according to the chosen visual template (CSS). The resources are usually (but not limited to) images that are used in the specific template.

## The "browser/src" Folder

Generally, this folder includes the HTML and CSS templates, each type in its own sub-folder.

## The "browser/src/sass" Folder

This folder contains visual (CSS) templates, which are used by SASS during the Build Process to produce the `main.css` file used during the application run-time. The template files have the `.scss` extension. They refer resources placed in the `browser/images` folder so it is recommended that the resources folder name inside `browser/images` will correspond to the name of the template in this folder. For more details, see View Templates

## The "browser/src/views" Folder

This folder contains HTML page templates for the application views. Those are files including parameterized HTML code and have `.handlebars` extension. During the Build Process those files form the templates.js file. The parameters are getting replaced with actual values in run-time.

## The "libs" Folder

This folder contains some library JavaScript files, used by the application. In the `jslib` folder there are the core m-pin and crypto files, from which the mpin-all.js is generated during the Build Process. The 3rd party library handlebars.runtime.min.js is also placed in the `libs` folder.

### The "build" Folder

This folder contains some Python Scripts which are run during the Build Process. The output of that Build Process is written into the `out/browser` sub-folder.

## View Templates

The Browser PIN Pad Application provides many customization options through templates for the HTML pages and CSS style sheets. The templates are generally located in the `browser/src` folder. They are converted to run-time files by `handlebars` and `sass` respectively. The process of creating the run-time files is called the Build Process.

### HTML Templates

The HTML templates are located in the `browser/src/views` folder. Those files include parameterized HTML code in which the parameters are replaced with actual values during run-time. The build script builds a templates.js which includes JavaScript code that generates HTML in run-time replacing the parameters with the correct values. During the Build Process the templates.js file is concatenated to the mpin.js file.

### CSS Templates

The Browser PIN Pad makes use of SASS to create the run-time `main.css` from the templates, located under the browser/src/sass folder. The actual templates are placed in the `templates` sub-folder as `.scss` files. Additional templates could be created in that folder as per the needs of the specific customer. The used template is chose in the settings.json file:

**settings.json**
```
{
        "URLBase": "/public/mpin",
        "templateName": "light"
}
```

## Build Process

The build process is the process of generating the run-time files from the parameterized source files, which include JavaScript, HTML and SASS files. The build process is handled by Grunt according to the Gruntfile.js file. The resulting run-time files are written into the `build/out/browser` folder and its sub-folders. As part of the build process the next activities are executed:

- All the View template files from the `browser/src/views` folder are "compiled" into the templates.js file as a JavaScript code. This task is performed by the `handlebars` utility.

- The `main.css` file is generated from `main.scss` and according the chosen CSS template, and written into `build/out/browser/css`.

- The mpin-all.js (`mpin-all.min.js`) files is generated from the files located in `libs/jslib`.

- Parameters in various parameterized files are being replaced with the actual values, and the resulting output is written in the corresponding relative path under the `build/out/browser` folder.

- The generated templates.js and mpin-all.js, and the 3rd party library handlebars.runtime.min.js are concatenated with mpin.js to form the *run-time mpin.js* file, which is written to `build/out/browser`.

The build process is triggered by the Grunt utility, as explained in Using Grunt.

## Using Grunt

Grunt is a utility that triggers and manages the Build Process according the `Gruntfile.js` file. It is used through the development process in order to build/update the run-time files. Note that in order run Grunt you need to enter the `browser` folder, where `Gruntfile.js` is located.

There are two scenarios in which Grunt is used:

- Run the build process manually, after all the code changes are made. This is the simpler case in which code changes are done and then the run-time files are explicitly built with the command `grunt build`:
- `$ cd ~/dev/frontend/browser`
- `$ grunt build`
- `Running "bgShell:makeDirs" (bgShell) task`
- `Running "bgShell:makeViews" (bgShell) task`
- `Running "bgShell:buildMPinAll" (bgShell) task`
- `MPin crypto build done.`
- `Running "bgShell:copyHandlebarsRuntime" (bgShell) task`
- `Running "bgShell:replaceURLBASE" (bgShell) task`
- `Running "bgShell:mkdirTmpSass" (bgShell) task`
- `Running "bgShell:copyTmpSass" (bgShell) task`
- `Running "bgShell:replaceTemplateMain" (bgShell) task`
- `Running "bgShell:replaceTemplateTemplate" (bgShell) task`
- `Running "bgShell:buildMPin" (bgShell) task`
- `MPin crypto build done.`
- `Running "bgShell:copyResources" (bgShell) task`
- `Running "sass:dist" (sass) task`
- `Done, without errors.`
- Code changes are done "on the fly" as Grunt monitors the files and triggers the Build Process when changes are made. In this scenario Grunt is run prior to making any changes to the code. Grunt is monitoring the files specified in the `watch:` section of the `Gruntfile.js`, and triggers corresponding actions:
- `$ cd ~/dev/frontend/browser`
- `$ grunt`
- `Waiting...OK`
- `>> File "src/sass/main.scss" changed.`
- `Running "sass:dist" (sass) task`
- `Done, without errors.`
  `Completed in 2.579s at Thu Jun 26 2014 16:55:02 GMT+0300 (EEST) - Waiting...`

# PIN Pad API

As already mentioned, the PIN Pad should be integrated into the Customer's Web Site by embedding it into an HTML page. In order to do this, the integrator should take care of the following:

1. Include the PIN Pad's main module, *mpin.js*, by adding into the HTML head a script tag similar to the following
   `<script type="text/javascript" src="{mpin-path}/mpin.js" />`
2. Instantiate and Initialize an `mpin` object. This is done with the following JavaScript code:
   `new mpin( <init-options> );`

The following is an example of an HTML page that does the above:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
                <title>M-Pin demo</title>
                <link
href="/public/css/certivox.css?v=ee4d555e38d8c86110e6130f077bc6e2" rel="stylesheet"
type="text/css" />
                <link
href="/public/css/mpin.min.css?v=b7676587a6d88e2818ef2f1a7d1fdae6" rel="stylesheet"
type="text/css" />
                <!-- Favicons
                ================================================== -->
                <link rel="shortcut icon"
href="/public/images/favicon.ico?v=206feb63bb8d6b052afb1192d57d83a7">
                <!-- Fonts
                ================================================== -->
                <link
href='//fonts.googleapis.com/css?family=Roboto:400,400italic,700,700italic'
rel='stylesheet' type='text/css'>
                <script type="text/javascript"
src="/public/mpin/mpin.js?r=139539313122"></script>
                <script type="text/javascript">
                        new mpin( {
                                targetElement: "pinHolder",
                                clientSettingsURL:
"http://192.168.10.138:8005/rps/clientSettings",
                                mobileAppFullURL:
"http://192.168.10.138:8005/m/index.html",
                                successSetupURL: "/successSetup",
                                successLoginURL: "/protected",
                                onSuccessSetup: function(setupData, onSuccess) {
                                        console.log("Setup PIN successful")
                                        onSuccess()
                                },
                                onSuccessLogin: function(authData) {
                                        console.log("Login successful")
                                        window.location = "/protected"
                        },
                                onVerifySuccess: function(data){
                                },
                                identityCheckRegex: "[0-9a-zA-Z]+"
                        });
                </script>
        </head>
```

```
	<body>
		<div id="header">
			<div class="container">
				<a href="http://certivox.com" target="_blank"
class="logo1"><img src="/public/images/certivox-
logo.jpg?v=0517e63d9febecab7894f22262f21d67" alt="CertiVox Logo" width="170"
height="33" title="CertiVox Logo" style="border-style: none"></a>
				<a href="http://www.certivox.com/m-pin/",
target="_blank" class="logo2"><img alt="M-Pin strong authentication logo"
src="/public/images/m-pin-logo-strong.png?v=3517e8e85a0030f3f4fad11d0e9448c9"
width="152" height="54" title="M-Pin strong authentication logo" style="border-style:
none"></a>
			</div>
			<div class="clear"></div>
		</div>
		<div id="content">
			<div class="container">
				<div class="nav">
					<ul>
						<li><a href="/"
class="active">Home</a></li>
						<li><a
href="http://www.certivox.com/m-pin/", target="_blank">About M-Pin</a></li>
						<li><a
href="http://www.certivox.com/m-pin-download" target="_blank">Download M-Pin</a></li>
						<li><a href="https://certivox.org"
target="_blank">Community</a></li>
					</ul>
					<div class="clear"></div>
				</div>
				<div class="content">
					<h1>Welcome to the M-Pin System Demo</h1>
					<!--action box start-->
					<div class="one column center grey marBot20
marTop20">
						<div id="pinHolder"
style="margin:auto; width:260px;">
							Loading PinPad...
						</div>
					</div>
					<div class="clear"></div>
				</div>
				<!--action box end-->
				<div id="footer">&copy; 2013 CertiVox UK Limited,
All Rights Reserved.</div>
			</div>
		</div>
	</body>
</html>
```

## Initialization Options

As shown in the above example, the PIN Pad (the `mpin` object) can be initialized with some options. Only few of those options are required. Most of them are retrieved from the RPS during the GET /rps/clientSettings request.

In case the same option is specified in the PIN Pad initialization and received from the RPS as well, the client-side options will overwrite the server-side ones.

### targetElement

(Mandatory) The `id` of the HTML element into which the PIN Pad will be embedded

### clientSettingsURL

(Mandatory) This is the URL/endpoint from which the PIN Pad should obtain the Client Settings from the RPS. Those settings might further be overwritten by some client-side settings, listed below.
Example: "http://192.168.10.138:8005/rps/clientSettings"

### requestOTP

(Optional) This is an indicator with valid values of "0" (Default) or "1" that indicates whether One Time Password should be produced as a result of successful authentication. If an OTP should be requested then it is also displayed to the end-user after the authentication. This is used primarily by the SSO Server.

### successSetupURL

(Optional) This is a relative or full URL which, if set, will be loaded after a successful end-user set-up, including setting a PIN code and storing the M-Pin token into the browser local storage.
Example: "/successSetup"

### successLoginURL

(Optional) This is a relative or full URL which, if set, will be loaded after a successful end-user authentication.
Example: "/protected"

### prerollid

(Optional) If this option is set with some end-user identity, then the PIN Pad will open directly into the page for end-user registration, with the given identity pre-loaded.

### signatureURL

(Optional) Endpoint for requesting Relying Party Signature. The Signature is used to get Client Secret Share.
Example: "http://192.168.10.138:8005/rps/signature"

### certivoxURL

(Optional) Base URL for the CertiVox-hosted Services.
Example: "https://m-pinapi-qa-v03.certivox.org/v0.3"

### mpinAuthServerURL

(Optional) M-Pin Server WebSocket or REST endpoint for end-user authentication.
If the option useWebSocket is set to *false* or WebSockets are not supported by the browser or the network, then the PIN Pad will make AJAX requests to *<mpinAuthServerURL>/pass1* and *<mpinAuthServerURL>/pass2* to authenticate an end-user. Otherwise, it will open a WebSocket to *<mpinAuthServerURL>/authenticationToken* and will authenticate the end-user through it.
Example:"wss://192.168.10.138:8002".

## registerURL

(Optional) Endpoint for initiating end-user registration flow.
Example: "http://192.168.10.138:8005/rps/user"

## accessNumberURL

(Optional) Endpoint for Access Number Generation request.
Example: "http://192.168.10.138:8005/rps/accessnumber"

## mobileAppFullURL

This is the full URL from which the Mobile Client might be loaded. If this option is not set the PIN Pad will not display on its start screen the options for Mobile Authentication.
Example: "http://192.168.10.138:8005/m/index.html"

## customHeaders

(Optional) An object including key-value pairs of additional headers to be added to back-end requests. This might be useful in some environments which expect certain headers in order to allow or process the requests.

## identityCheckRegex

(Optional) A string containing regular expression which the format of a new identity will be verified against. The default is a regular expression validating identities with format of e-mail address. If you would like to overwrite this, you may set a different regex.
Example: "[0-9a-zA-Z]+"

## seedValue

(Optional) Ephemeral hex-encoded value, used as a seed for further random number generation.

## appID

(Optional) Customer specific Application ID, received upon Customer/Application registration for the M-Pin Service.

## useWebSocket

(Optional) If set to *false*, the PIN Pad will not make any attempts to use WebSockets for communication with the M-Pin Server during end-user authentication.

## setupDoneURL

(Optional) Endpoint for the setupDone request.
Example: "http://192.168.10.138:8005/rps/setupDone"

## timePermitsURL

(Optional) Endpoint for Customer Time Permit Share request.
Example: "http://192.168.10.138:8005/rps/timePermit"

### RPAVerifyUserURL

(Optional) This option should be used only when the PIN Pad works against the CertiVox-hosted RPS, available mainly for development and demo purposes.
This URL should be the RPA End-user Verification Endpoint to which the RPS should make POST request to verify end-user identity or to start the verification process. When the RPS is hosted by the Customer, this is configured into the RPS configuration file and is send to the PIN Pad with the GET /rps/clientSettings request, but when using the Hosted RPS, this option should be set in the PIN Pad, which passes it to the RPS with the *PUT /rps/user/<mpin-id>* request.

### RPAPermitUserURL

(Optional) This option should be used only when the PIN Pad works against the CertiVox-hosted RPS, available mainly for development and demo purposes.
This URL should be the RPA End-user Permission Endpoint on which the RPS should make a GET request to get permission for authentication for a specific user. When the RPS is hosted by the Customer, this might be configured into the RPS configuration file and is send to the PIN Pad with the GET /rps/clientSettings request, as it is not mandatory. When using the Hosted RPS, this option might be set in the PIN Pad, in which case it will send it to the RPS with the *GET /rps/timePermit/<mpin-id>* request. If this option is not set, the RPS will assume that any user is permitted by the RPA to try and authenticate.

### RPAAuthenticateUserURL

(Optional) This option should be used only when the PIN Pad works against the CertiVox-hosted RPS, available mainly for development and demo purposes.
This URL should be the RPA End-user Authentication Endpoint on which the authentication token should be verified. When the RPS is hosted by the Customer, this is configured into the RPS configuration file and is send to the PIN Pad with the GET /rps/clientSettings request, but when using the Hosted RPS, this option should be set in the PIN Pad, which will make the POST request to this endpoint to verify the end-user authentication, providing the result of the "M-Pin Protocol" as the data.

## Callback Functions

### onSuccessSetup( data, done() )

(Optional) If the option successSetupURL is not set, then this function, if set, will be called with the first argument being JSON structure with the user ID ( `{ "userId": <user-id> }` ), and the second argument is the completion callback to be called when the execution of onSuccessSetup() was completed. When the completion callback is called, the PIN Pad will navigate to the "Setup Completed" page.

### onSuccessLogin( data )

(Optional) If this function is set, it will be called after successful end-user authentication, as the passed data will be the response data of the authentication verification request to the RPA. The format of the data is RPA implementation specific.

### onLoaded()

(Optional) This callback is called without data every time the PIN Pad is navigated to its home screen - either it was loaded, re-loaded or the Home button was pressed.

## onGetPermit( permit )

(Optional) If such a function is set, it is called when the Time Permit Shares were successfully obtained and combined. The argument is a string containing the hex-encoded combined Time Permit.

## onAccountDisabled( mpinId )

(Optional) If this callback is defined, it will be called when an end-user has exceeded the allowed amount of unsuccessful authentication attempts and the account was disabled. The passed argument is the hex-encoded M-Pin ID.

## onUnsupportedBrowser( msg )

(Optional) If this callback is defined, it will be called when the PIN Pad discovers that the browser doesn't support the features, necessary for the PIN Pad operation. Such features include HTML5, Local Storage and others. The argument is a readable message describing the discovered incompatibility.

## onError( msg )

(Optional) If this function is set, it will be called when an error condition occurs and the passed data will be a string containing the error message.

## onGetSecret()

(Optional) If this function is set, it will be called without any arguments to notify that the Client Secret was successfully obtained. This will happen before the end-user enters his/her PIN and the M-Pin token is created and stored in the browser.

## authenticateRequestFormatter( data )

(Optional) If this function is set, it will be called before the PIN Pad executes the authentication verification request to the RPA. The data argument includes the payload that is going to be sent with the request and this data might be modified by the callback.
The modified data should be returned by the function. The format of the original data is:

```
{
        "mpinResponse": {
                "authOTT": <authentication-ott>,
                "version": "0.3",
                "pass": 2
        }
}
```

The *mpinResponse* object is the same object returned by the *PASS2* request to the M-Pin Server.
If authTokenFormatter() is not defined, then the above data is sent "as is" in the  authentication verification request to the RPA.

## accessNumberRequestFormatter( data )

(Optional) If this function is set, it will be called before the PIN Pad executes the POST /rps/accessnumber request to the RPS. The data argument includes the payload that is going to be sent with the request and this data might be modified by the callback.
The modified data should be returned by the function. The format of the original data is:

```
{
        "webOTT": <web-ott>
}
```

**NOTE** that if the original data fields are changed, this might compromise the flow. Therefore it is strongly recommended to only add data fields, and to avoid changes to the original ones.

### registerRequestFormatter( data )

(Optional) If this function is set, it will be called prior to executing the PUT /rps/user request to the RPS. The data argument includes the payload that is going to be sent with the request and this data might be modified by the callback.
The modified data should be returned by the function. The format of the original data is:

```
{
    "userId": <user-id>,
    "mobile": <0|1>,
    ["regOTT": <registration-ott>]
}
```

**NOTE** that if the original data fields are changed, this might compromise the flow. Therefore it is strongly recommended to only add data fields, and to avoid changes to the original ones.

## Language Customization

The PIN Pad allows the texts that appear in the UI to be customized or to be defined for additional languages. This is done through the *customLanguageTexts* and the *language* settings. The default language settings are:

```
{
        en: {
                pinpad_initializing: "Initializing...",
                pinpad_errorTimePermit: "ERROR GETTING PERMIT:",
                home_alt_mobileOptions: "Mobile Options",
                home_button_authenticateMobile: "Authenticate <br/>with your 
Smartphone",
                home_button_authenticateMobile_description: "Get your Mobile Access 
Number to use with your M-Pin Mobile App to securely authenticate yourself to this 
service.",
                home_button_getMobile: "Get <br/>M-Pin Mobile App",
                home_button_getMobile_description: "Install the free M-Pin Mobile App 
on your Smartphone now!  This will enable you to securely authenticate yourself to 
this service.",
                home_button_authenticateBrowser: "Authenticate <br/>with this 
Browser",
                home_button_authenticateBrowser_description: "Enter your M-PIN to 
securely authenticate yourself to this service.",
                home_button_setupBrowser: "Add an <br/>Identity to this Browser",
                home_button_setupBrowser_description: "Add your Identity to this web 
browser to securely authenticate yourself to this service using this machine.",
                mobileGet_header: "GET M-PIN MOBILE APP",
                mobileGet_text1: "Scan this QR Code or open this URL on your
Smartphone:",
                mobileGet_text2: "or open this URL on your mobile:",
                mobileGet_button_back: "Back",
                mobileAuth_header: "AUTHENTICATE WITH YOUR M-PIN",
                mobileAuth_seconds: "seconds",
                mobileAuth_text1: "Your Access Number is:",
                mobileAuth_text2: "Note: Use this number in the next",
                mobileAuth_text3: "with your M-Pin Mobile App.",
                mobileAuth_text4: "Warning: Navigating away from this page will
interrupt the authentication process and you will need to start again to authenticate
successfully.",
                otp_text1: "Your One-Time Password is:",
                otp_text2: "Note: The password is only valid for<br/>{0} seconds
before it expires.", // {0} will be replaced with the max. seconds
                otp_seconds: "Remaining: {0} sec.", // {0} will be replaced with the
remaining seconds
                otp_expired_header: "Your One-Time Password has expired.",
                otp_expired_button_home: "Login again to get a new OTP",
                setup_header: "ADD AN IDENTITY TO THIS DEVICE",
                setup_text1: "Enter your email address:",
                setup_placeholder: "your email address",
                setup_text2: "Your email address will be used as your identity when
M-Pin authenticates you to this service.",
                setup_error_unathorized: "{0} has not been registered in the
system.", // {0} will be replaced with the userID
                setup_error_server: "Cannot process the request. Please try again
later.",
                setup_error_signupexpired: "Your signup request has been expired.
Please try again.",
                setup_button_setup: "Setup M-Pin&trade;",
                setupPin_header: "Create your M-Pin with {0} digits", // {0} will be
replaced with the pin length
                setupPin_initializing: "Initializing...",
                setupPin_pleasewait: "Please wait...",
                setupPin_button_clear: "Clear",
                setupPin_button_done: "Setup<br />Pin",
                setupPin_errorSetupPin: "ERROR SETTING PIN: {0}", // {0} is the
request status code
                setupDone_header: "Congratulations!",
                setupDone_text1: "Your M-Pin identity",
                setupDone_text2: "is setup, now you can login.",
                setupDone_text3: "",
                setupDone_button_go: "Login now",
                setupReady_header: "VERIFY YOUR IDENTITY",
                setupReady_text1: "Your M-Pin identity",
                setupReady_text2: "is ready to setup, now you must verify it.",
                setupReady_text3: "We have just sent you an email, simply click the
link to verify your identity.",
                setupReady_button_go: "Verified your identity? <br/>Setup your M-Pin
now",
                setupReady_button_resend: "Not received the email? <br/>Send it
again",
                setupNotReady_header: "YOU MUST VERIFY <br/>YOUR IDENTITY",
                setupNotReady_text1: "Your identity",
                setupNotReady_text2: "has not been verified.",
                setupNotReady_text3: "You need to click the link in the email we sent
you, and then choose 'Setup M-Pin'.",
                setupNotReady_check_info1: "Checking",
                setupNotReady_check_info2: "Identity not verified!",
                setupNotReady_resend_info1: "Sending email",
                setupNotReady_resend_info2: "Email sent!",
                setupNotReady_resend_error: "Sending email failed!",
                setupNotReady_button_check: "Setup M-Pin",
                setupNotReady_button_resend: "Send the email again",
                setupNotReady_button_back: "Go to the identities list",
                authPin_header: "Enter your M-Pin",
                authPin_button_clear: "Clear",
                authPin_button_login: "Login",
                authPin_pleasewait: "Authenticating...",
                authPin_success: "Success",
                authPin_errorInvalidPin: "INCORRECT PIN!",
                authPin_errorNotAuthorized: "You are not authorized!",
                authPin_errorExpired: "The auth request expired!",
                authPin_errorServer: "Server error!",
                deactivated_header: "SECURITY ALERT",
                deactivated_text1: "has been de-activated and your M-Pin token has
been revoked.",
                deactivated_text2: "To re-activate your identity, click on the blue
button below to register again.",
                deactivated_button_register: "Register again",
                account_button_addnew: "Add a new identity to this list",
                account_button_delete: "Remove this M-Pin Identity from this
browser",
                account_button_reactivate: "Forgot my PIN. Send me a new activation
email.",
                account_button_backToList: "Go back to identity list",
                account_button_cancel: "Cancel and go back",
                account_delete_question: "Are you sure you wish to remove this M-Pin
Identity from this browser?",
                account_delete_button: "Yes, remove this M-Pin Identity",
                account_reactivate_question: "Are you sure you wish to reactivate
this M-Pin Identity?",
                account_reactivate_button: "Yes, reactivate this M-Pin Identity",
                noaccount_header: "No identities have been added to this browser!",
                noaccount_button_add: "Add a new identity",
                home_intro_text: "First let's establish truth to choose the best way
for you to access this service:",
                signin_btn_desktop1: "Sign in with Browser",
                signin_btn_desktop2: "(This is a PERSONAL device which I DO trust)",
                signin_btn_mobile1: "Sign in with Smartphone",
                signin_mobile_btn_text: "Sign in with your Smartphone",
                signin_btn_mobile2: "(This is a PUBLIC device which I DO NOT trust)",
                home_txt_between_btns: "or",
                home_hlp_link: "Not sure which option to choose?",
                mobile_header_txt1: "I",
                mobile_header_donot: "DON'T",
                mobile_header_do: "DO",
                mobile_header_txt3: "trust this computer",
                help_text_1: "Simply choose a memorable <b>[4 digit]</b> PIN to
assign to this identity by pressing the numbers in sequence followed by the 'Setup'
button to setup your PIN for this identity",
                help_ok_btn: "Ok, Got it",
                help_more_btn: "I'm not sure, tell me more",
                help_hub_title: "M-Pin Help Hub",
                help_hub_li1: "What is the difference between signing in with the
browser or with Smartphone?",
                help_hub_li2: "Which is the most secure method to sign in?",
                help_hub_li3: "What details will i need to provide?",
                help_hub_li4: "Who can see my identity?",
                help_hub_button: "Exit Help Hub and return to previous page",
                help_hub_3_p1: "You will simply need to provide an <span
class="mpinPurple">[email address]</span> in order to set up your identity. You will
receive an activation email to complete the set up process.",
                help_hub_3_p2: "You will also need to create a PIN number, this will
be a secret <span class="mpinPurple">[4 digit]</span> code known only to you which you
will use to login to the service.",
                help_hub_return_button: "Return to Help Hub",
                activate_header: "ACTIVATE YOUR IDENTITY",
                activate_text1: "Your M-Pin identity:",
                activate_text2: "is ready to setup.",
                activate_text3: "We have just send you an email, simply click the
link in the email to activate your identity.",
                activate_btn1: "Activated your identity via email? Setup your M-Pin
now",
                activate_btn2: "Not received the activation email? Send it again!",
                settings_title: "IDENTITY OPTIONS",
                landing_button_newuser: "I'm new to M-Pin, get me started",
                mobile_header: "GET THE M-PIN SMARTPHONE APP",
                mobile_footer_btn: "Now, sign in with your Smartphone",
                pinpad_setup_screen_text: "CREATE YOUR M-PIN:<br> CHOOSE 4 DIGIT",
                pinpad_default_message: "ENTER YOUR PIN"
       }
}
```

The default value for the *language* setting is "en".

You can customize the texts in the following way:

```
(function() {
        new mpin( {
                targetElement: "pinHolder",
                clientSettingsURL: "http://192.168.10.138:8005/rps/clientSettings",
                mobileAppFullURL: "http://192.168.10.138:8005/m/index.html",
                ...
                customLanguageTexts: {
                        en: {
                                setup_text1: "Enter your identity:"
                        }
                }
        })
})();
```

Alternatively, one can define texts for alternative language like this:

```
(function() {
        new mpin( {
                targetElement: "pinHolder",
                clientSettingsURL: "http://192.168.10.138:8005/rps/clientSettings",
                mobileAppFullURL: "http://192.168.10.138:8005/m/index.html",
                ...
                customLanguageTexts: {
                        fr: {
                                pinpad_initializing: ...,
                                pinpad_errorTimePermit: ...,
                                home_alt_mobileOptions: ...,
                                ...
                        }
                },
                language: "fr"
        })
})();
```

Custom texts might also be provided within the *GET /clientSettings* response.

## Example: Customizing the Customer's logo

In order to for a developer or an integrator to put a custom logo instead of the "LOGO HERE" logo placeholder, he/she needs to perform the following tasks:

1. Place your logo under the folder `browser/images`. In the same folder you can find the placeholder logo `logo-here-xxxx.svg`.
2. Modify the file `browser/src/sass/templates/_<template-name>.scss` by making the necessary changes in the section `@mixin header-logo()`, which by default is looking similar to the following:

```
@mixin header-logo() {
  margin: 5px 0;
  width: 85px;
  height: 20px;
  float: right;
  background: url('#{$IMAGES}logo-here-grey.svg') no-repeat;
  background-size: auto 100%;
  background-position: right bottom;
}
```

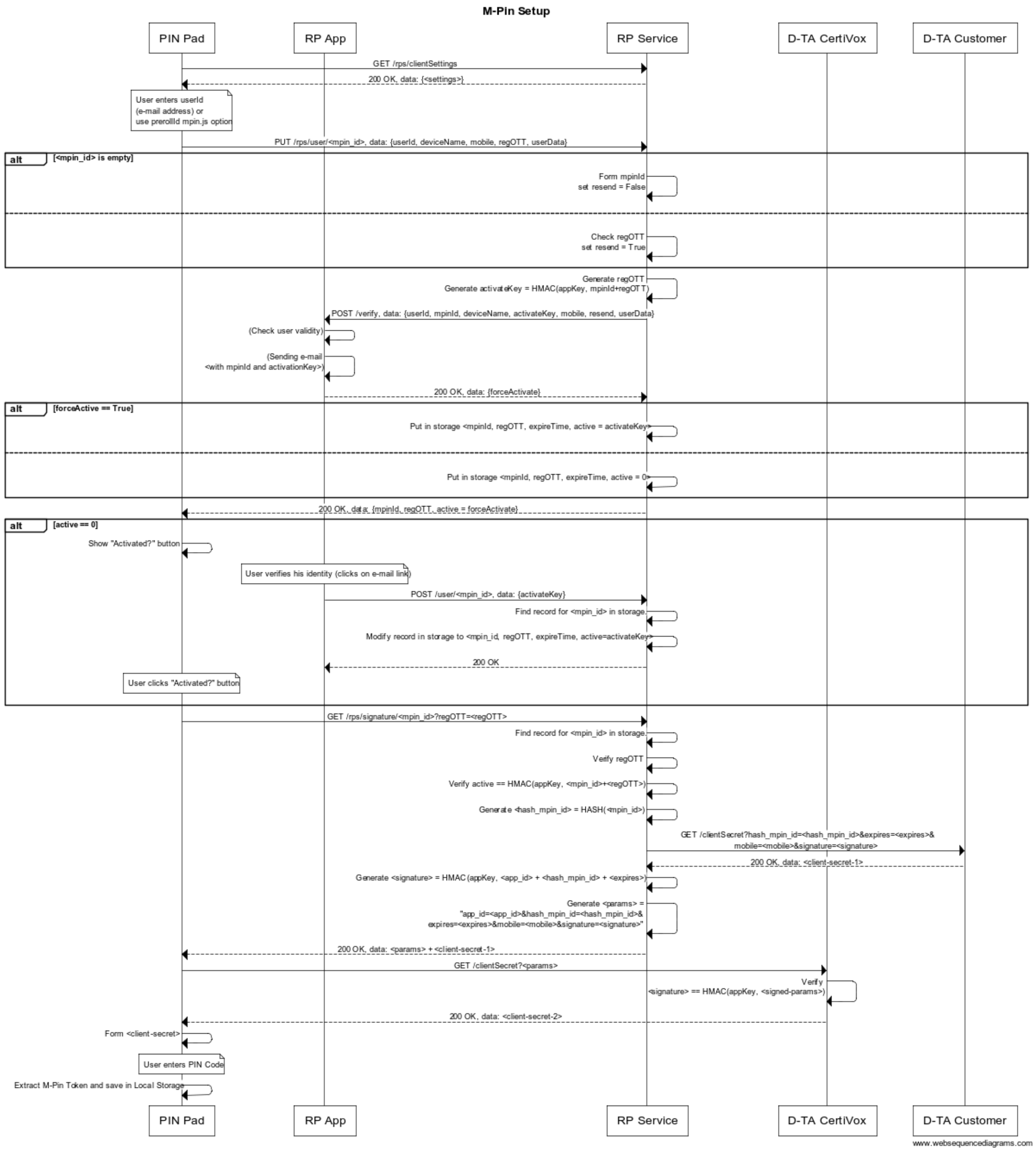
Here the path and name of the new custom logo might be specifies, as well as some visual attributes as height/width or margin might be adjusted.

11. Use Grunt to build the Mobile App and the result will be present under the `build/browser` folder

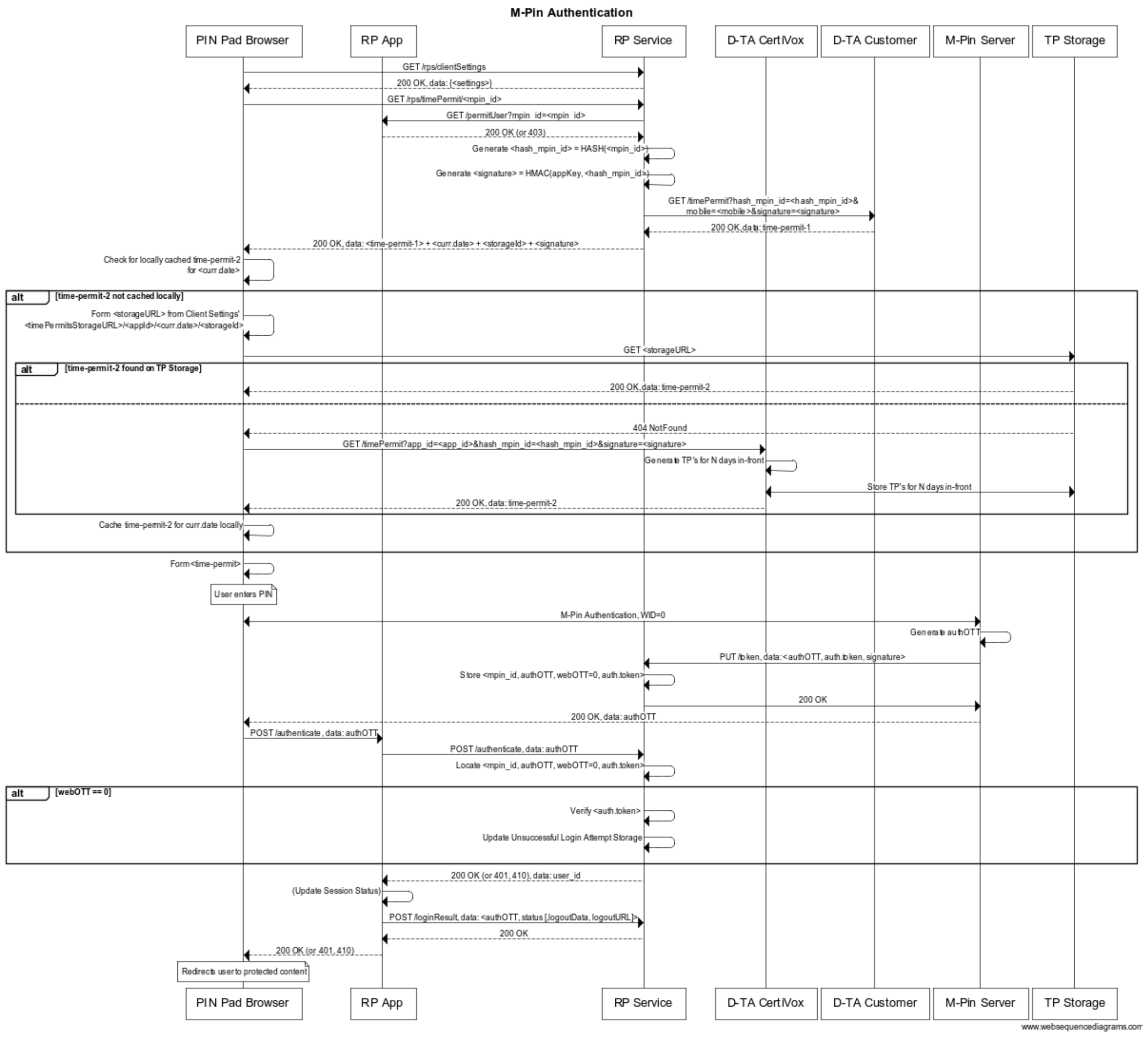
# Workflows

The M-Pin workflows are common to all the components in the solution. For detailed explanation about the calls and the flow see Tech_Spec_-_M-Pin_v3.3.0_Relying_Party_Service.

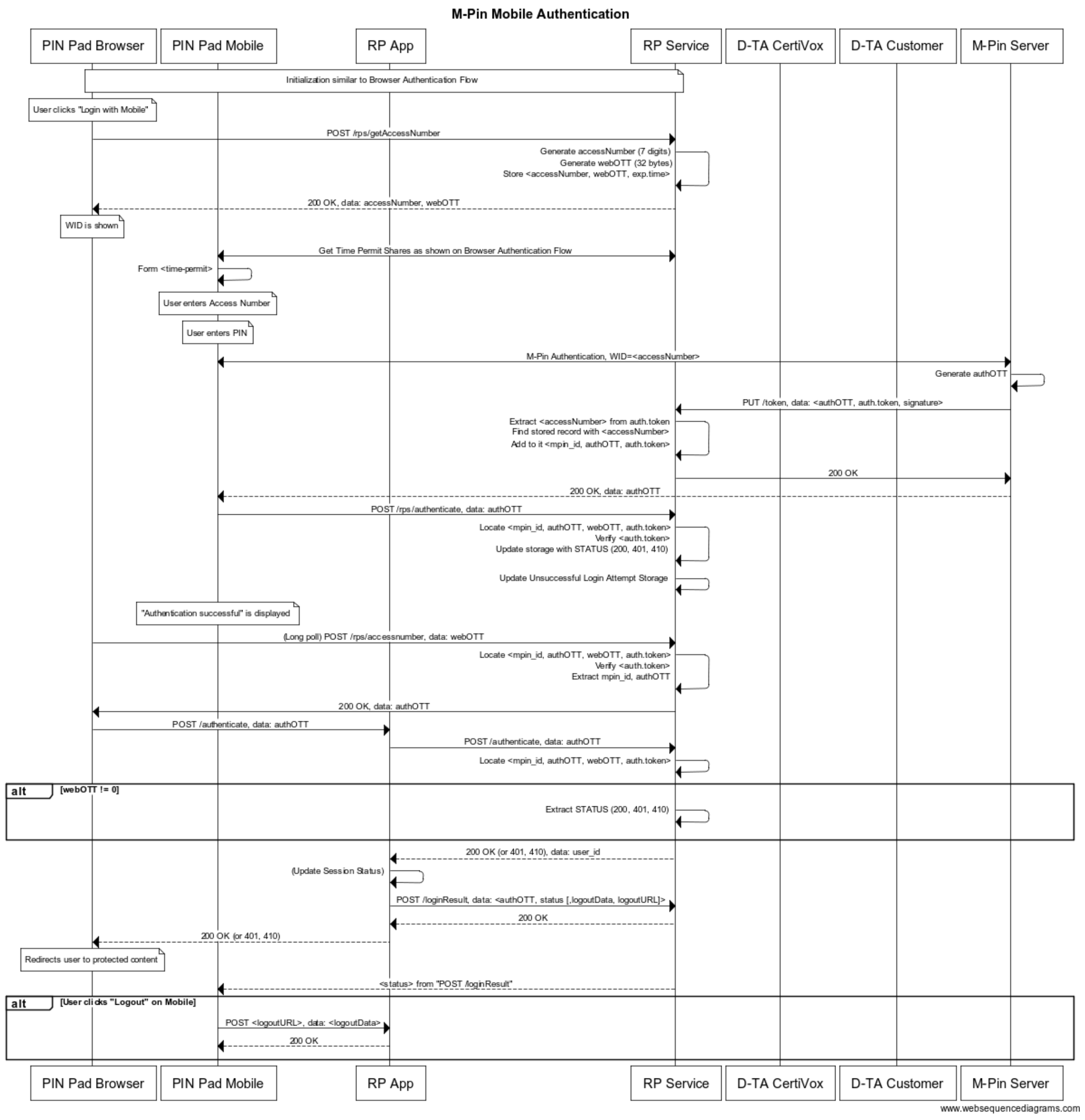
## M-Pin Setup


M-Pin Setup
PIN Pad
RP App
RP Service
D-TA CertiVox
D-TA Customer
GET /rps/clientSettings
200 OK, data: {<settings>}
User enters userId (e-mail address) or use prerollId mpin.js option
PUT /rps/user/<mpin_id>, data: {userId, deviceName, mobile, regOTT, userData}
alt [<mpin_id> is empty]
Form mpinId set resend = False
Check regOTT set resend = True
Generate regOTT Generate activateKey = HMAC(appKey, mpinId+regOTT)
POST /verify, data: {userId, mpinId, deviceName, activateKey, mobile, resend, userData}
(Check user validity)
(Sending e-mail <with mpinId and activationKey>)
200 OK, data: {forceActivate}
alt [forceActive == True]
Put in storage <mpinId, regOTT, expireTime, active = activateKey>
Put in storage <mpinId, regOTT, expireTime, active = 0>
200 OK, data: {mpinId, regOTT, active = forceActivate}
alt [active == 0]
Show "Activated?" button
User verifies his identity (clicks on e-mail link)
POST /user/<mpin_id>, data: {activateKey}
Find record for <mpin_id> in storage.
Modify record in storage to <mpin_id, regOTT, expireTime, active=activateKey>
200 OK
User clicks "Activated?" button
GET /rps/signature/<mpin_id>?regOTT=<regOTT>
Find record for <mpin_id> in storage.
Verify regOTT
Verify active == HMAC(appKey, <mpin_id>+<regOTT>)
Generate <hash_mpin_id> = HASH(<mpin_id>)
GET /clientSecret?hash_mpin_id=<hash_mpin_id>&expires=<expires>& mobile=<mobile>&signature=<signature>
200 OK, data: <client-secret-1>
Generate <signature> = HMAC(appKey, <app_id> + <hash_mpin_id> + <expires>)
Generate <params> = "app_id=<app_id>&hash_mpin_id=<hash_mpin_id>& expires=<expires>&mobile=<mobile>&signature=<signature>"
200 OK, data: <params> + <client-secret-1>
GET /clientSecret?<params>
Verify <signature> == HMAC(appKey, <signed-params>)
200 OK, data: <client-secret-2>
Form <client-secret>
User enters PIN Code
Extract M-Pin Token and save in Local Storage
www.websequencediagrams.com

## M-Pin Authentication


M-Pin Authentication
PIN Pad Browser
RP App
RP Service
D-TA CertiVox
D-TA Customer
M-Pin Server
TP Storage
GET /rps/clientSettings
200 OK, data: {<settings>}
GET /rps/timePermit/<mpin_id>
GET /permitUser?mpin_id=<mpin_id>
200 OK (or 403)
Generate <hash_mpin_id> = HASH(<mpin_id>)
Generate <signature> = HMAC(appKey, <hash_mpin_id>)
GET /timePermit?hash_mpin_id=<hash_mpin_id>&mobile=<mobile>&signature=<signature>
200 OK, data: time-permit-1
200 OK, data: <time-permit-1> + <curr.date> + <storageId> + <signature>
Check for locally cached time-permit-2 for <curr.date>
alt [time-permit-2 not cached locally]
Form <storageURL> from Client Settings' <timePermitsStorageURL>/<appId>/<curr.date>/<storageId>
GET <storageURL>
alt [time-permit-2 found on TP Storage]
200 OK, data: time-permit-2
404 NotFound
GET /timePermit?app_id=<app_id>&hash_mpin_id=<hash_mpin_id>&signature=<signature>
Generate TP's for N days in-front
Store TP's for N days in-front
200 OK, data: time-permit-2
Cache time-permit-2 for curr.date locally
Form <time-permit>
User enters PIN
M-Pin Authentication, WID=0
Generate authOTT
PUT /token, data: <authOTT, auth.token, signature>
Store <mpin_id, authOTT, webOTT=0, auth.token>
200 OK
200 OK, data: authOTT
POST /authenticate, data: authOTT
POST /authenticate, data: authOTT
Locate <mpin_id, authOTT, webOTT=0, auth.token>
alt [webOTT == 0]
Verify <auth.token>
Update Unsuccessful Login Attempt Storage
200 OK (or 401, 410), data: user_id
(Update Session Status)
POST /loginResult, data: <authOTT, status [,logoutData, logoutURL]>
200 OK
200 OK (or 401, 410)
Redirects user to protected content
www.websequencediagrams.com

## M-Pin Mobile Authentication


M-Pin Mobile Authentication
PIN Pad Browser
PIN Pad Mobile
RP App
RP Service
D-TA CertiVox
D-TA Customer
M-Pin Server
Initialization similar to Browser Authentication Flow
User clicks "Login with Mobile"
POST /rps/getAccessNumber
Generate accessNumber (7 digits)
Generate webOTT (32 bytes)
Store <accessNumber, webOTT, exp.time>
200 OK, data: accessNumber, webOTT
WID is shown
Get Time Permit Shares as shown on Browser Authentication Flow
Form <time-permit>
User enters Access Number
User enters PIN
M-Pin Authentication, WID=<accessNumber>
Generate authOTT
PUT /token, data: <authOTT, auth.token, signature>
Extract <accessNumber> from auth.token
Find stored record with <accessNumber>
Add to it <mpin_id, authOTT, auth.token>
200 OK
200 OK, data: authOTT
POST /rps/authenticate, data: authOTT
Locate <mpin_id, authOTT, webOTT, auth.token>
Verify <auth.token>
Update storage with STATUS (200, 401, 410)
Update Unsuccessful Login Attempt Storage
"Authentication successful" is displayed
(Long poll) POST /rps/accessnumber, data: webOTT
Locate <mpin_id, authOTT, webOTT, auth.token>
Verify <auth.token>
Extract mpin_id, authOTT
200 OK, data: authOTT
POST /authenticate, data: authOTT
POST /authenticate, data: authOTT
Locate <mpin_id, authOTT, webOTT, auth.token>
alt [webOTT != 0]
Extract STATUS (200, 401, 410)
200 OK (or 401, 410), data: user_id
(Update Session Status)
POST /loginResult, data: <authOTT, status [,logoutData, logoutURL]>
200 OK
200 OK (or 401, 410)
Redirects user to protected content
<status> from "POST /loginResult"
alt [User clicks "Logout" on Mobile]
POST <logoutURL>, data: <logoutData>
200 OK
PIN Pad Browser
PIN Pad Mobile
RP App
RP Service
D-TA CertiVox
D-TA Customer
M-Pin Server
www.websequencediagrams.com