Tech Spec - M-Pin v3.3.0 Mobile App

Contents

General

This document describes the technical details of the M-Pin v3.3.0 Mobile Client - the Mobile App. 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.

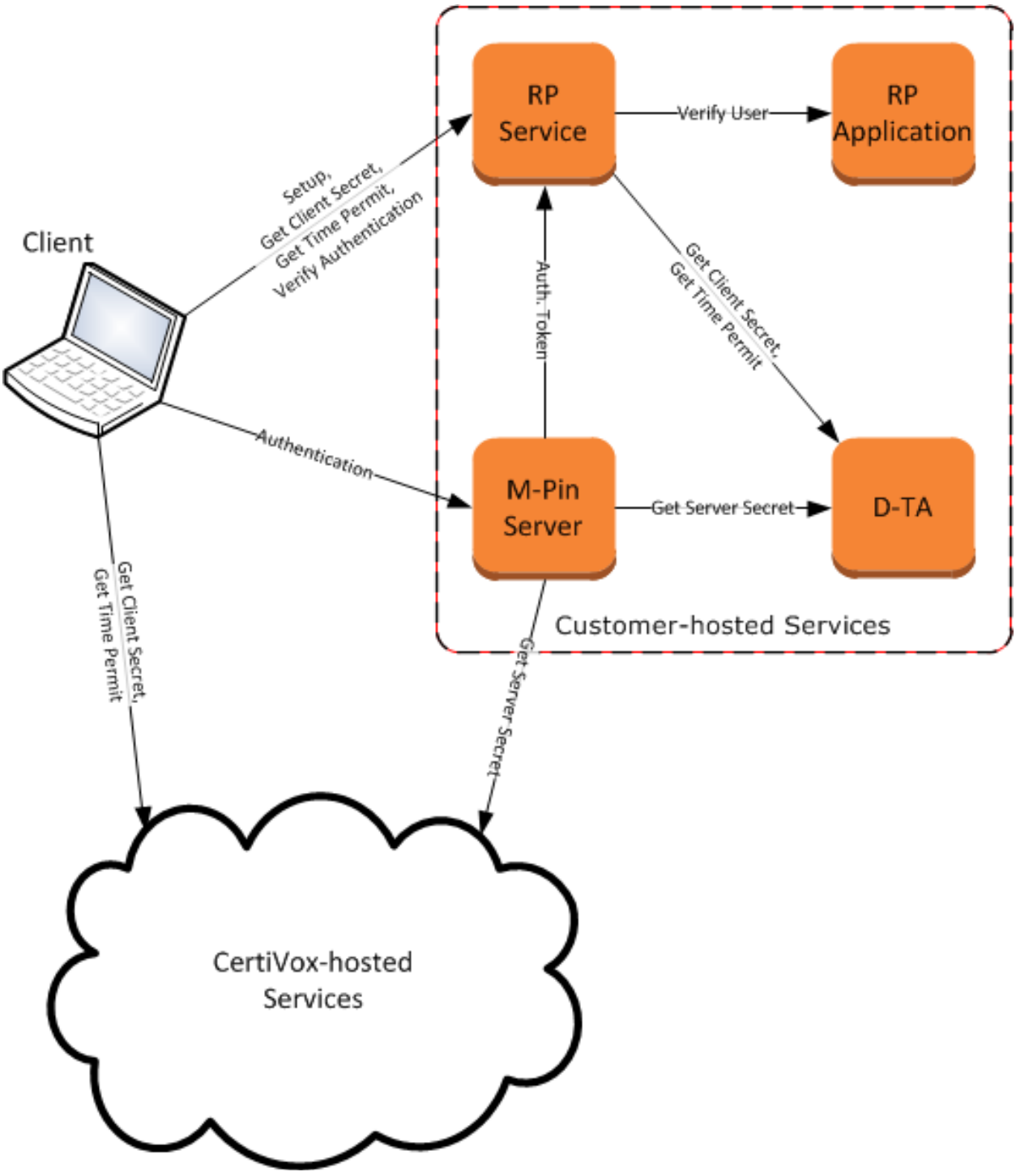
PIN Pad

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


RP Service
RP Application
Verify User
Client
Setup,
Get Client Secret,
Get Time Permit,
Verify Authentication
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

Unlike the Browser PIN Pad, the Mobile App is not intended to be integrated into a Web Application, but just served by this application. The PIN Pad is initialized with the URL address for loading the Mobile App (see Tech_Spec_-_M-Pin_v3.3.0_PIN_Pad for more details). This URL is specific for the Customer's Web Application and the Customer should take care to serve the Mobile App from this URL.

Once loaded, the Mobile App asks the RPS for Client Settings, sending a GET /rps/clientSettings request. The response should include all the settings that the Mobile App needs in order to communicate with the Customer or CertiVox-hosted Services.

NOTE: By default, the GET /rps/clientSettings request is sent to the relative URL /rps/clientSettings, which would work only if the Mobile App is served from the RPA, but might not work otherwise. In this case, the Integrator should modify the Mobile App's settings.json file and set the correct URL.

Other than that, the Mobile App should be a completely autonomous and self-included web application.

Main Files

index.html

This is the HTML page that is initially loaded. It "includes" the main.js module, and has the code to instantiate and initialize the `mpin` object:

```
(function() {
	new mpin("mpinMaster", {
		clientSettingsURL: "@@clientsetting",
		emailCheckRegex: @@emailregex,
		onSuccessLogin: function() {
			alert(" :: on SUCCESS LOGIN ::");
		}
	});
})();
```

This file includes some parameters that are replaced with actual values through the Build Process. In the above sample code two such parameters could be seen - `@@clientsetting` and `@@emailregex`.

main.js

This is the main module of the Mobile Client, which "uses" the other modules. It is included by the index.html page as explained above. This module takes care of rendering the different pages of the Mobile App, i.e. those pages that are specific to the Mobile Client. This file also includes parameters, which are replaced with actual values through the Build Process. Example for such a parameter is `@@templatename`.

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 main.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`.

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.

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. For instance, it replaces all the parameters that are used in some parameterized files, with the actual values, listed in settings.json. This is done in the `replace:` section of the file.

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.

Code Structure

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

The "mobile" Folder

This folder contains some files that are general for the whole application:

- index.html - the starting HTML page as described above.
- Gruntfile.js - the main file that defines the rules for the Grunt operation (see Build Process and Using Grunt)
- settings.json - the file that specifies the actual values to be used for various parameters.

The "mobile/js" Folder

This folder contains the main JavaScript file - main.js.

The "mobile/resources" Folder

This folder contains resources for the visualization of the application. The resources are placed in the `templates` sub-folder, into additional 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 "mobile/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 `mobile/resources` folder so it is recommended that the resources folder name inside `mobile/resources` will correspond to the name of the template in this folder. For more details, see View Templates

The "mobile/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/mobile` sub-folder.

View Templates

The Mobile PIN Pad Application provides many customization options through templates for the HTML pages and CSS style sheets. The templates are generally located in the `mobile/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 `mobile/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.

CSS Templates

The Mobile App makes use of SASS to create the run-time `main.css` from the templates, located under the `mobile/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

```
{
        "clientSettingsURL": "/rps/clientSettings",
        "templateName": "gradient",
        "emailRegex": "/.*/"
}
```

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/mobile` folder and its sub-folders. As part of the build process the next activities are executed:

- All the View template files from the `mobile/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/mobile/css`.
- The `mpin-all.js` (`mpin-all.min.js`) files is generated from the files located in `libs/jslib`, and written into `build/out/mobile/js`.
- 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/mobile` folder.

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 `mobile` 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/mobile
$ grunt build
Running "bgShell:makeDirs" (bgShell) task
Running "bgShell:makeViews" (bgShell) task
Running "bgShell:buildMPinAll" (bgShell) task
MPin crypto build done.
Running "bgShell:copyResources" (bgShell) task
Running "bgShell:copyHandlebarsRuntime" (bgShell) task
Running "bgShell:copySASS" (bgShell) task
Running "replace:dist" (replace) task
Replace index.html → ../build/out/mobile/index.html
Replace mpin.appcache → ../build/out/mobile/mpin.appcache
Replace src/sass/main.scss → ../build/tmp/mobile/sass/main.scss
Replace src/sass/templates/_darkgrey.scss →
../build/tmp/mobile/sass/templates/_darkgrey.scss
Replace src/sass/templates/_flatwhite.scss →
../build/tmp/mobile/sass/templates/_flatwhite.scss
```

- Replace src/sass/templates/_gradient.scss → ../build/tmp/mobile/sass/templates/_gradient.scss
- Replace js/main.js → ../build/out/mobile/js/main.js
- 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/mobile
- $ gruntRunning "watch" task
- Waiting...OK
- >> File "src/sass/main.scss" changed.
- Running "bgShell:makeDirs" (bgShell) task
- Running "bgShell:makeViews" (bgShell) task
- Running "bgShell:buildMPinAll" (bgShell) task
- MPin crypto build done.
- Running "bgShell:copyResources" (bgShell) task
- Running "bgShell:copyHandlebarsRuntime" (bgShell) task
- Running "bgShell:copySASS" (bgShell) task
- Running "replace:dist" (replace) task
- Replace index.html → ../build/out/mobile/index.html
- Replace mpin.appcache → ../build/out/mobile/mpin.appcache
- Replace src/sass/main.scss → ../build/tmp/mobile/sass/main.scss
- Replace src/sass/templates/_darkgrey.scss → ../build/tmp/mobile/sass/templates/_darkgrey.scss
- Replace src/sass/templates/_flatwhite.scss → ../build/tmp/mobile/sass/templates/_flatwhite.scss
- Replace src/sass/templates/_gradient.scss → ../build/tmp/mobile/sass/templates/_gradient.scss
- Replace js/main.js → ../build/out/mobile/js/main.js
- Running "sass:dist" (sass) task
- Done, without errors.
  Completed in 1.634s at Fri Oct 24 2014 17:33:23 GMT+0300 (EEST) - Waiting...

Language Customization

The Mobile App 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 of the mpin object, which is instantiated in the index.html file. The default language settings are:

```
{
        en: {
                pinpad_initializing: "Initializing...",
                pinpad_errorTimePermit: "ERROR GETTING PERMIT:",
                home_alt_mobileOptions: "Mobile Options",
                home_button_authenticateMobile_noTrust: "Sign in with Smartphone <br>
(This is a PUBLIC device which I DO NOT trust)",
                home_button_authenticateMobile_trust: "Sign in with Browser <br>
(This is a PERSONAL device which I DO trust)",
                home_button_authenticateMobile_intro: "First let's establish trust to
choose the best way for you to access this service:",
```

```
                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",
                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_text2: "Your email address will be used as your identity when
M-Pin authenticates you to this service.",
                setup_text3: "your email address",
                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",
                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 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 <br/> 'Setup M-Pin'.",
                setupNotReady_check_info1: "Checking",
                setupNotReady_check_info2: "Identity not verified!",
                setupNotReady_resend_info1: "Sending email",
                setupNotReady_resend_info2: "Email sent!",
                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 M-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",
                pinpad_placeholder_text: "Enter your pin",
                pinpad_placeholder_text2: "Enter your access Number",
                logout_text1: "YOU ARE NOW LOGGED IN",
                logout_button: "Logout",
                home_button_setupMobile: "Add an identity to this browser",
                mobile_splash_text: "INSTALL THE M-PIN MOBILE APP",
                mobile_add_home_ios6: "Tap the <img
src='resources/templates/@@templatename/img/ios6-share.png'/> icon to 'Add to
homescreen'",
                mobile_add_home_ios7: "Tap the <img
src='resources/templates/@@templatename/img/ios7-share.png'/> icon to 'Add to
homescreen'",
                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",
```

```
                logout_btn: "Sign out",
                success_header: "Success",
                success_text: "You are now signed in as:",
                accessdenied_header: "Access Denied",
                accessdenied_text: "Your M-Pin identity",
                accessdenied_text_cont: "has been removed from this device.",
                accessdenied_btn: "Register again"
        }
}
```

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

You can customize the texts in the following way in the index.html file:

```
(function() {
        new mpin("mpinMaster", {
                clientSettingsURL: "@@clientsetting",
                emailCheckRegex: @@emailregex,
                onSuccessLogin: function() {
                        alert(" :: on SUCCESS LOGIN ::");
                }
                ...
                customLanguageTexts: {
                        en: {
                                setup_text1: "Enter your identity:"
                        }
                }
        })
})();
```

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

```
(function() {
        new mpin("mpinMaster", {
                clientSettingsURL: "@@clientsetting",
                emailCheckRegex: @@emailregex,
                onSuccessLogin: function() {
                        alert(" :: on SUCCESS LOGIN ::");
                }
                ...
                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 `mobile/resources/templates/<template-name>/img`, as <*template-name*> is the name of the currently used template. In the same folder you can find the placeholder logo `logo-here-xxxx.svg`.
2. Modify the file `mobile/src/sass/templates/_<template-name>.scss` by making the necessary changes in the section `@mixin custom-logo()`, which by default is looking similar to the following:

```
@mixin custom-logo() {
  width: 106px;
  height: 32px;
  background-image: url('#{$IMAGES}/logo-here-grey.svg');
  background-repeat: no-repeat;
  background-position: right;
  background-size: auto 100%;
  margin: 5px 10px 5px 0px;
  @include box-flex-value(1.0);
}
```

   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.

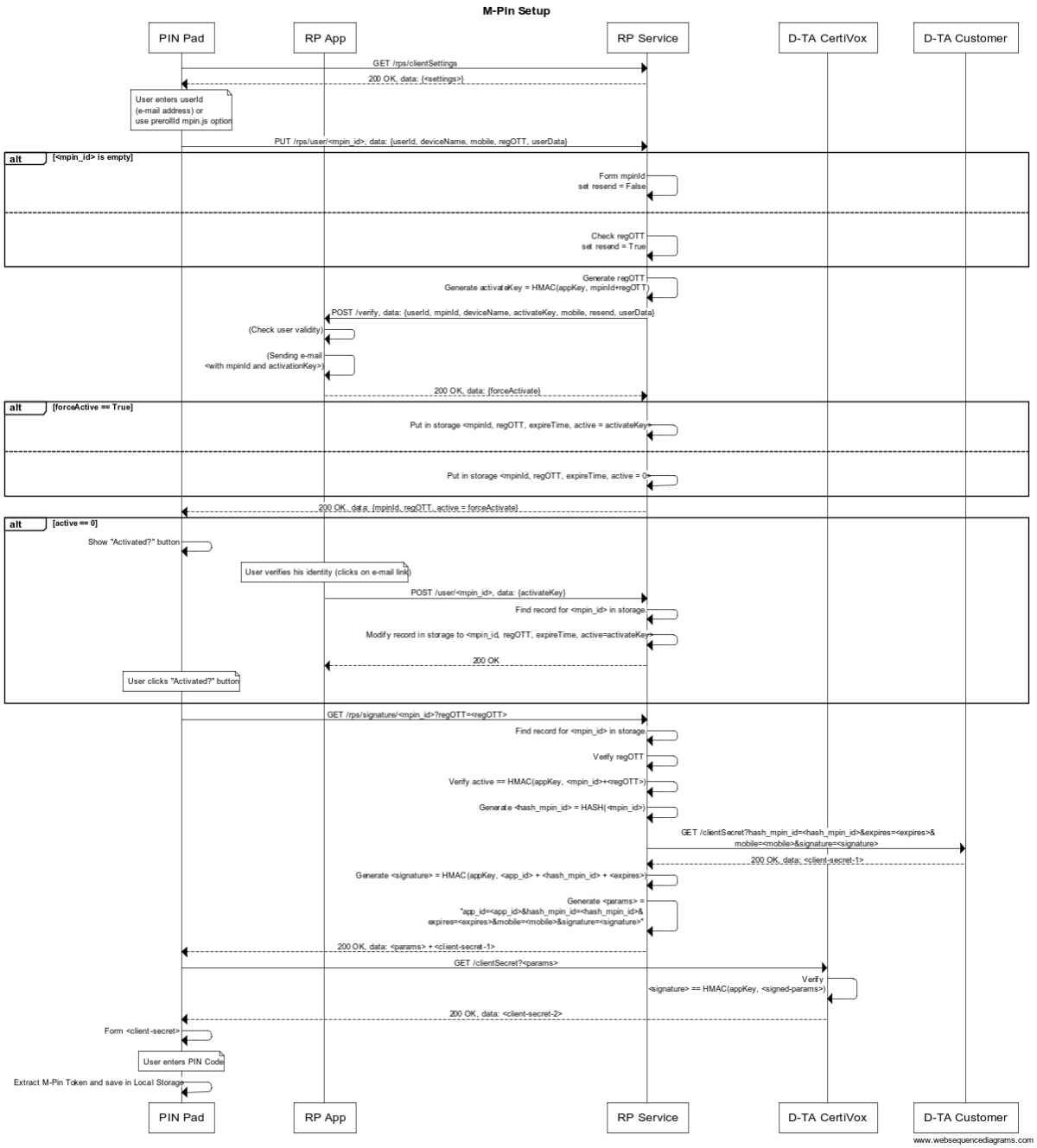
12. Use Grunt to build the Mobile App and the result will be present under the `build/mobile` folder

Workflows

The M-Pin workflows are common to all the components in the solution. Moreover, the Mobile App uses the same flows as the Browser PIN Pad. For detailed explanation about the calls and the flow see M-Pin_PIN_Pad_v3.3.0_Tech_Spec and Tech_Spec_-_M-Pin_v3.3.0_Relying_Party_Service

- **M-Pin Setup**

This flow is the same for the PIN Pad and the Mobile App, so "PIN Pad" in the diagram below could be replaced by "Mobile App".



- **M-Pin Mobile Authentication**