Mobile Foundry

User Interface Reference

Table of Contents

[Introduction 3](#__RefHeading___Toc1534_2019578025)

[Default Configuration 3](#__RefHeading___Toc1536_2019578025)

[Screens 4](#__RefHeading___Toc1538_2019578025)

[AboutScreen 4](#__RefHeading___Toc1540_2019578025)

[Configuration 4](#__RefHeading___Toc1542_2019578025)

[AccountDetailsScreen 5](#__RefHeading___Toc1544_2019578025)

[Configuration 6](#__RefHeading___Toc1546_2019578025)

[AddRuleScreen 7](#__RefHeading___Toc1548_2019578025)

[Configuration 8](#__RefHeading___Toc1550_2019578025)

[AllDevicesScreen 8](#__RefHeading___Toc1552_2019578025)

[Configuration 10](#__RefHeading___Toc1554_2019578025)

[AppSettingsScreen 10](#__RefHeading___Toc1556_2019578025)

[Configuration 11](#__RefHeading___Toc1558_2019578025)

[AutoDeviceScreen 12](#__RefHeading___Toc1560_2019578025)

[AylaBLEDetailsScreen 12](#__RefHeading___Toc1562_2019578025)

[ChooseConfigScreen 12](#__RefHeading___Toc1564_2019578025)

[DeviceDetailsScreen 13](#__RefHeading___Toc1566_2019578025)

[DeviceRulesScreen 13](#__RefHeading___Toc1568_2019578025)

[DeviceScreen 14](#__RefHeading___Toc1570_2019578025)

[DeviceSettingsScreen 14](#__RefHeading___Toc1572_2019578025)

[Configuration 15](#__RefHeading___Toc1574_2019578025)

[DeviceSwipeContainer 16](#__RefHeading___Toc1576_2019578025)

[HTMLViewScreen 16](#__RefHeading___Toc1578_2019578025)

[Configuration 16](#__RefHeading___Toc1580_2019578025)

[MenuScreen 17](#__RefHeading___Toc1582_2019578025)

[Configuration 18](#__RefHeading___Toc1584_2019578025)

[NoDevicesScreen 18](#__RefHeading___Toc1586_2019578025)

[Configuration 18](#__RefHeading___Toc1588_2019578025)

[ScheduleActionScreen 19](#__RefHeading___Toc1590_2019578025)

[ScheduleListScreen 20](#__RefHeading___Toc1592_2019578025)

[ScheduleScreen 20](#__RefHeading___Toc1594_2019578025)

[SetupWizardScreen 21](#__RefHeading___Toc1596_2019578025)

[Configuration 22](#__RefHeading___Toc1598_2019578025)

[ShareDevicesScreen 23](#__RefHeading___Toc1600_2019578025)

[ShareListScreen 24](#__RefHeading___Toc1602_2019578025)

[SignInScreen 25](#__RefHeading___Toc1604_2019578025)

[Configuration 26](#__RefHeading___Toc1606_2019578025)

[Controls 26](#__RefHeading___Toc1608_2019578025)

[ActionPicker 27](#__RefHeading___Toc1610_2019578025)

[Configuration 27](#__RefHeading___Toc1612_2019578025)

[EVBControl 28](#__RefHeading___Toc1614_2019578025)

[OnOffControl 30](#__RefHeading___Toc1616_2019578025)

[Configuration 31](#__RefHeading___Toc1618_2019578025)

[TempSetpointControl 32](#__RefHeading___Toc1620_2019578025)

[Configuration 32](#__RefHeading___Toc1622_2019578025)

# Introduction

This document is a part of the Mobile Foundry documentation relating to the available user interface elements available to developers as part of Mobile Foundry. Readers of this document should already be familiar with the Mobile Foundry architecture and configuration from other documents included with the library:

*Mobile Foundry Architecture*

*Mobile Foundry Configuration Specification*

*Mobile Foundry Getting Started Guide*

This document serves as a reference for mobile application developers using Mobile Foundry as the foundation for their applications. Included are the descriptions for each Screen and Control that ships with Mobile Foundry along with details on the customization and configuration of each element.

# Default Configuration

The **userExperience** section of the Mobile Foundry configuration file is the root of all user interface and experience configuration. One of the elements within the **userExperience** section is the **defaultConfig**, which contains the default values for all screens and controls.

Screens and controls may individually override values in their own configuration if desired, similar to cascading style sheets used for the presentation of web content.

The default configuration should contain at least the following elements, although any element may be supplied in the configuration and will be discovered by any Screen or Control that recognizes it:

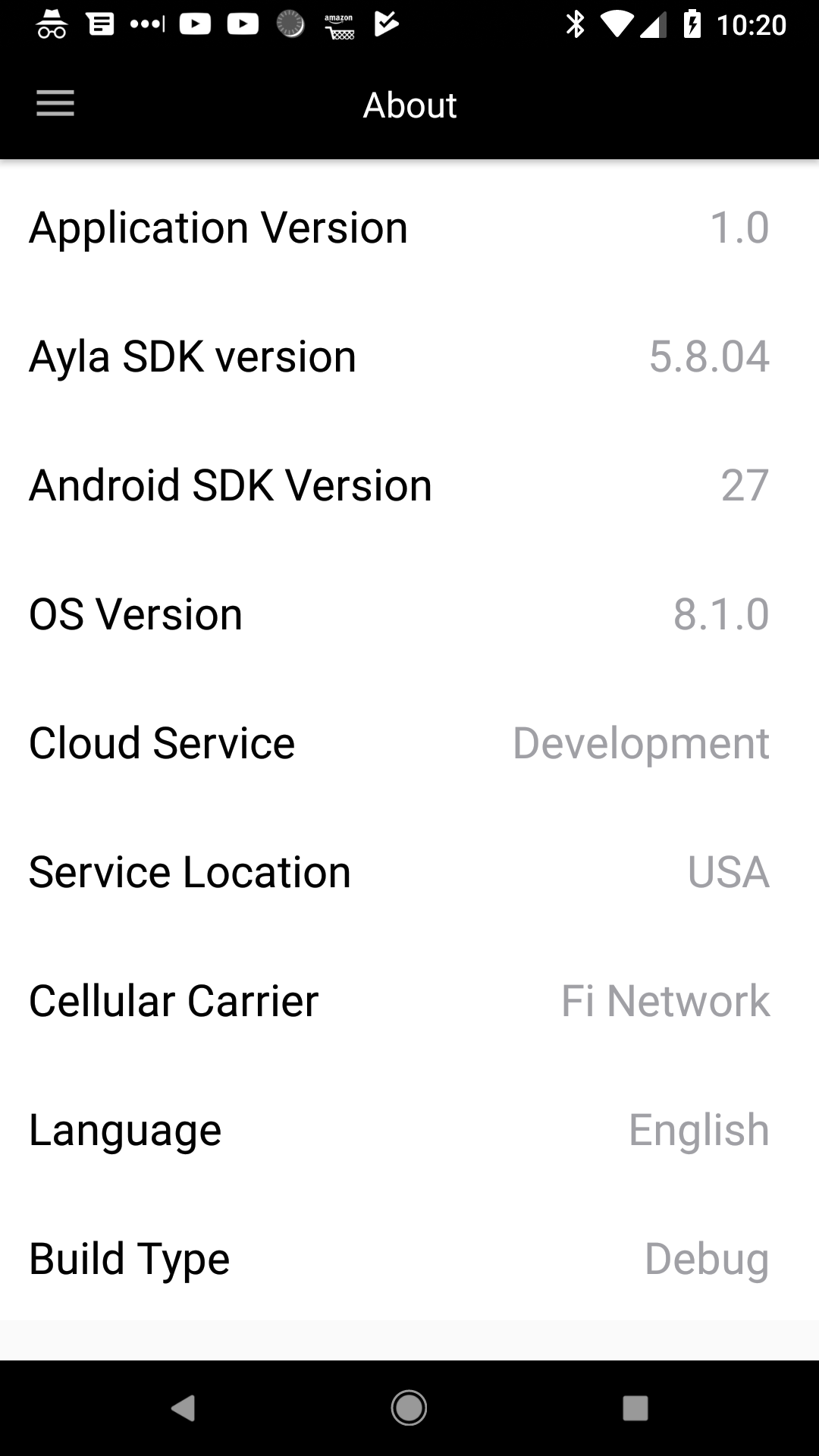
|  |  |
| --- | --- |
| windowBackgroundColor | Default color for window backgrounds |
| windowBackgroundImage | Background image for a window. No image is used if not set. |
| textColor | Default color for text |
| textBackgroundColor | Default color for text backgrounds. Defaults to transparent. |
| listBackgroundColor | Background color for list views |
| listTextColor | Text color for list view windows. If not defined, defaults to textColor. |
| listTextBackgroundColor | Text color for the background of text in list views. If not defined, defaults to transparent. |
| listDividerColor | Color for the divider between list items. Defaults to transparent. |
| editTextColor | Color for editable text fields. If not defined, defaults to textColor |
| editTextBackgroundColor | Background color for editable text fields. If not defined, defaults to textBackgroundColor. |
| toolbarColor | Color for the toolbar at the top of the application |
| toolbarTextColor | Color for text shown in the toolbar |
| buttonTextColor | Text color for buttons |
| buttonBackgroundColor | Background color for buttons |
| buttonBackgroundImage | Background image for buttons, overrides buttonBackgroundColor |
| accentColor | Color used for highlighting areas of interest |
| wantsTitleBar | Boolean, if true, show the title bar. If false, do not. |
| disableDrawerMenu | Boolean, if true, will not use a drawer menu for naviagation |
| allowOfflineModification | Boolean, if true, will allow devices that are offline to be updated |

# Screens

This section lists the various Screens that are supplied with Mobile Foundry along with Screen-specific configuration information.

## AboutScreen

The AboutScreen may be shown to provide information about the application:



### Configuration

{

"class": "AboutScreen",

"name": "about\_screen",

"title": "about\_screen",

"icon": "about\_screen\_icon",

"extras": {

"includedItems": {

"app\_version": true,

"ayla\_sdk\_version": true,

"mobile\_sdk\_version": true,

"os\_version": true,

"ayla\_service": true,

"ayla\_service\_location": true,

"network\_operator": true,

"language": true,

"build\_type": true

},

"labelColor":"#020202",

"valueColor":"#A0A0A5",

"separatorColor":"#00000000",

"listBackgroundColor":"#FFFFFF"

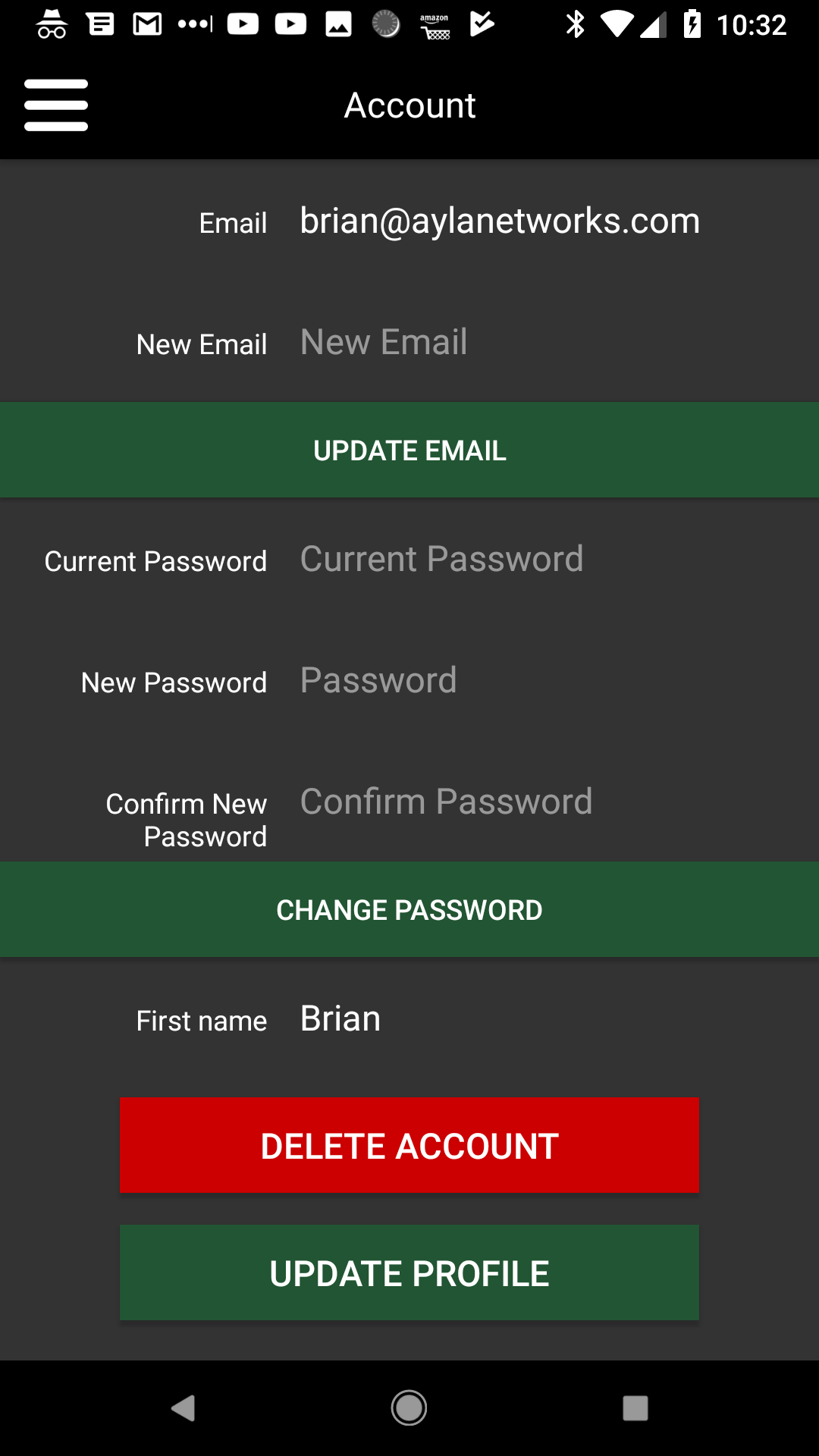
}

}

Setting any of the **includedItems** to false will remove that item from the displayed list.

## AccountDetailsScreen

The AccountDetailsScreen provides the user with editable account information. This screen may be used to sign up a new user, or for an existing user to edit their account details.



### Configuration

Two different configurations are shown below. The first configuration sets up the screen called “edit\_account” with the mode “editAccount”, while the second sets up the screen called “sign\_up” with the mode “newAccount”.

The first screen will launch filled out with the user’s account information and offer to update. The second will launch empty and prompt to create a new account.

{

"class": "AccountDetailsScreen",

"name": "edit\_account",

"title": "edit\_account",

"icon": "ic\_account",

"extras": {"mode": "editAccount"}

},

{

"class": "AccountDetailsScreen",

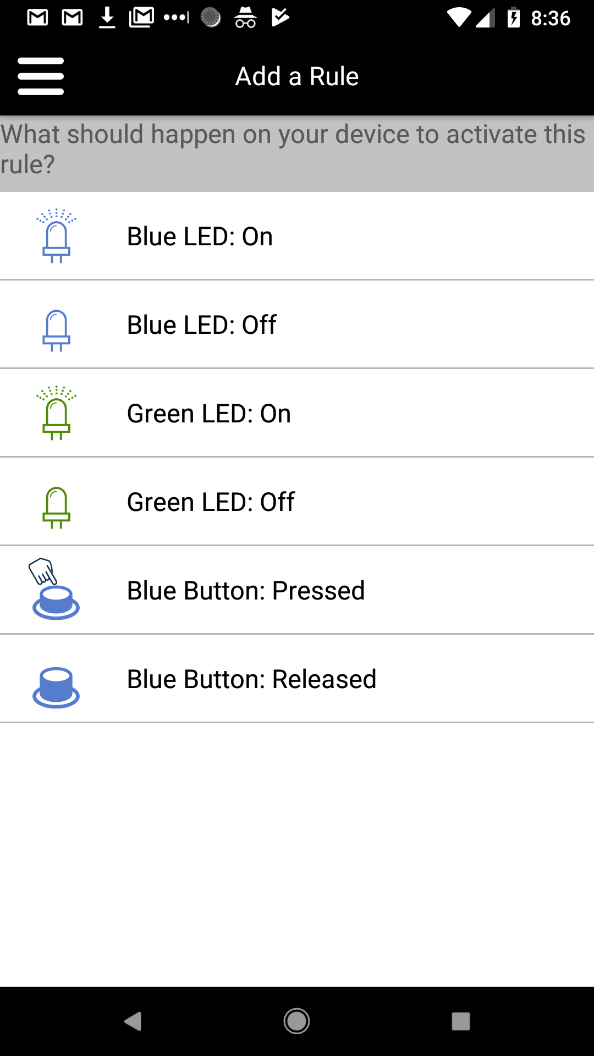
"name": "sign\_up",

"title": "sign\_up",

"extras": {"mode": "newAccount", "disableDrawerMenu": true}

}

## AddRuleScreen

The AddRuleScreen provides an interface for the user to add a rule to a device. The screen should be launched from a screen that has a device context, such as the DeviceDetails screen or any screen that shows information about a device.

### Configuration

This screen has no special configuration.

{

"class": "AddRuleScreen",

"name": "add\_rule",

"title": "add\_rule",

"icon": "add\_rule\_icon"

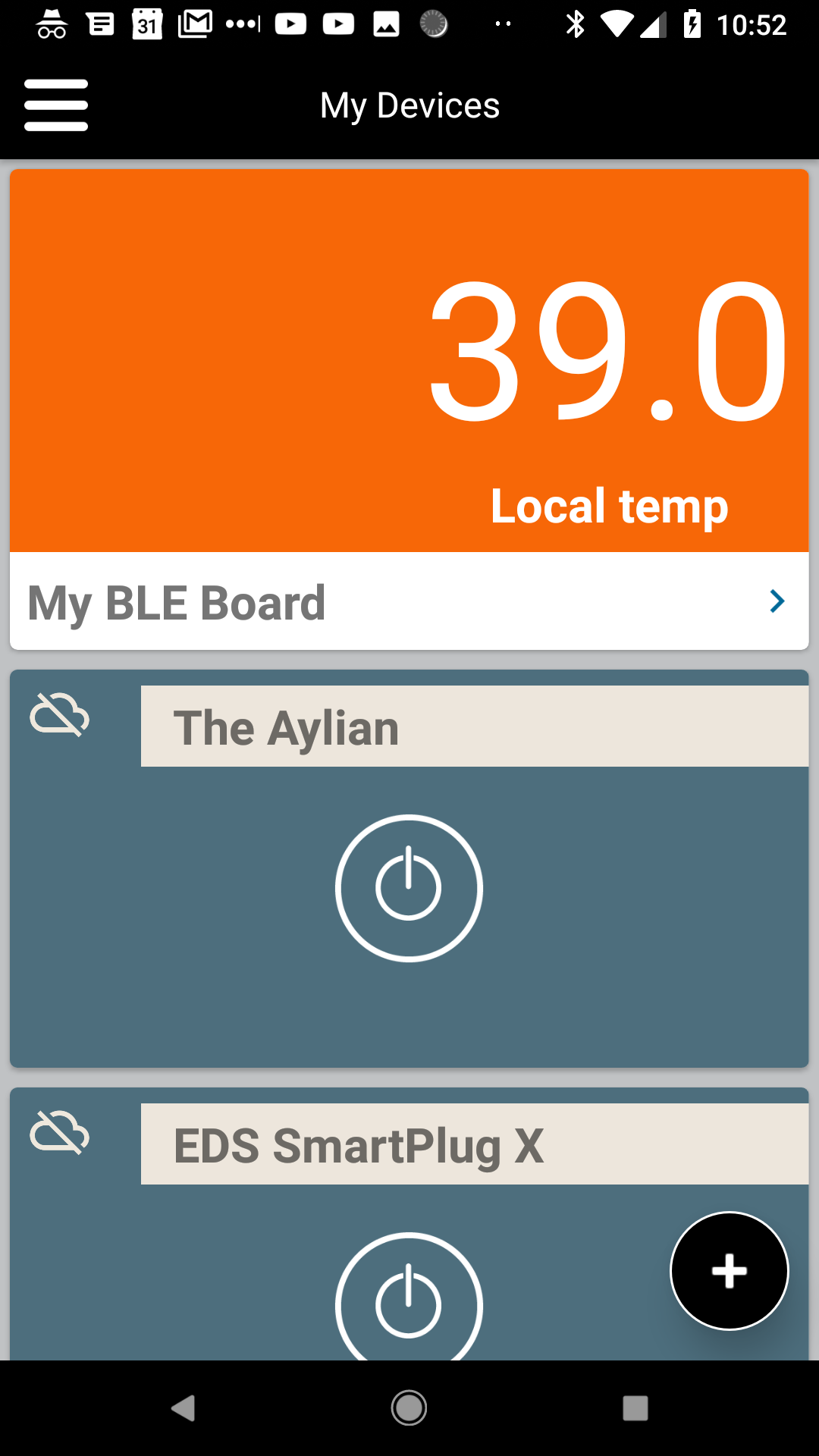
}

## AllDevicesScreen

The AllDevicesScreen is used as the **homeScreen** in many configurations. It provides a list of devices registered to the user’s account, displaying each device in the list by using it’s **deviceControl**, if configured to have one.

The screen has an optional configuration field that may be set to link this Screen with another Screen to help the user register new devices. If set, the AllDevicesScreen will display an add button (+) which will launch the specified Screen when tapped.

The example configuration file provided with Mobile Foundry uses this to launch the SetupWizardScreen.

The screenshot below shows the AllDevicesScreen with a BLE Demo Board and two OnOffDeviceControls, as well as the add button that will launch the SetupWizardScreen:

### Configuration

The configuration for the AllDevicesScreen includes an optional reference to an addDeviceScreen which will be launched upon tapping the add button, if present.

It also contains a Boolean value that if set to true, will include gateways in the list. If false, gateways will not be displayed.

{

"class": "AllDevicesScreen",

"name": "device\_list",

"icon": "ic\_device\_list",

"title": "device\_list",

"extras": {

"addDeviceScreen": "setup\_wizard",

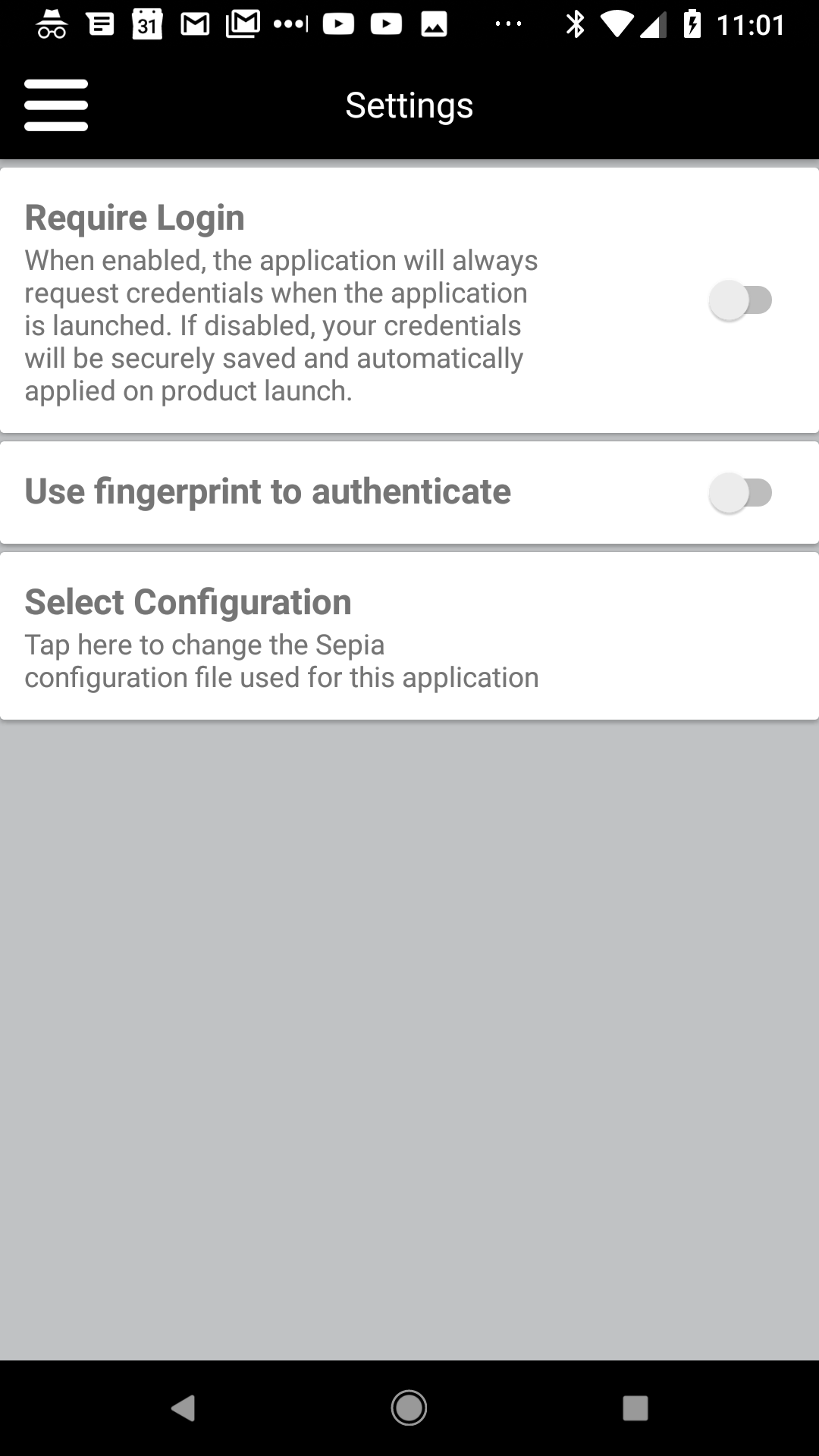
"includeGateways": true

}

},

## AppSettingsScreen

The AppSettingsScreen presents the user with configurable application-wide settings. The settings are defined in the configuration file along with their default values. Changes made via this screen will persist between application launches, and may be obtained anywhere in the application using the *SepiaAppSettings* class.



### Configuration

The AppSettingsScreen uses values from the **applicationSettings** section of the configuration, which lives under the **userExperience** section. The application settings exist whether or not the AppSettingsScreen is used, though the user will not be able to change any of the values.

The applicationSettings section contains an array of setting objects, each of which contains the following fields:

|  |  |
| --- | --- |
| name | Name of the setting |
| description | Description of the setting |
| type | Boolean | String | Integer | Decimal | MultipleChoice | LaunchScreen |
| defaultValue | Default value for the setting |

"applicationSettings": [

{

"name": "require\_login",

"description": "require\_login\_description",

"type": "Boolean",

"defaultValue": false

},

{

"name": "use\_fingerprint\_auth",

"type": "Boolean",

"defaultValue": false

},

{

"name": "choose\_config",

"description": "choose\_config\_description",

"type": "LaunchScreen",

"defaultValue": "choose\_config"

},

{

"name": "temperatureUnits",

"type": "MultipleChoice",

"choices": [ "F", "C"],

"defaultValue": "C"

}

]

The setting type may be a standard datatype such as Integer, String, Boolean or Decimal, or may be a set of choices (string values) presented to the user, or the setting may instead launch a screen to gather the required information. In the latter case, the specified *defaultValue* is the name of the Screen to launch when selected by the user.

## AutoDeviceScreen

The AutoDeviceScreen is generally used as a tool during development. It contains a DeviceControl for the device as well as a list of the managed properties for the device with PropertyControls present.

This screen is useful to test and display the device and property controls during development and has no special configuration.

## AylaBLEDetailsScreen

The AylaBLEDetailsScreen is the designated *detailScreen* for the Ayla BLE Demo Board device.

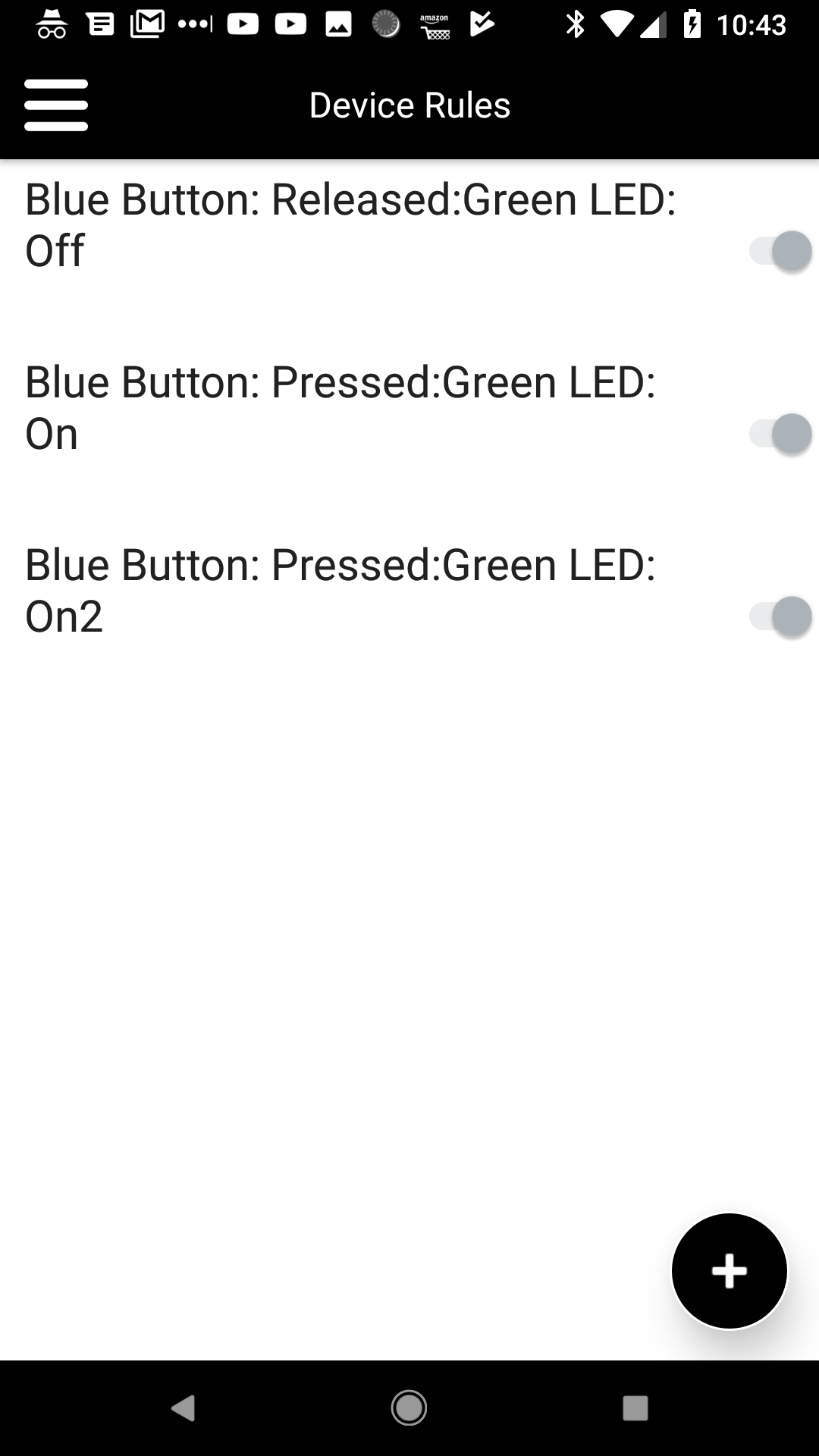
## ChooseConfigScreen

The ChooseConfigScreen is used during development to switch between configurations. It will present a list of all configurations found and re-start the application when a new configuration is selected. This may be used to A/B test between different configurations, and should not be included in a publicly-facing application.

## DeviceDetailsScreen

The DeviceDetailsScreen is a generic screen used by most devices in the default configuration. It displays the device name, which may be tapped and renamed by the user, as well as a list of details about the device including DSN, IP address, product class, etc.

## DeviceRulesScreen

The DeviceRulesScreen displays a list of rules for a device as well as a switch allowing the user to enable or disable each rule.

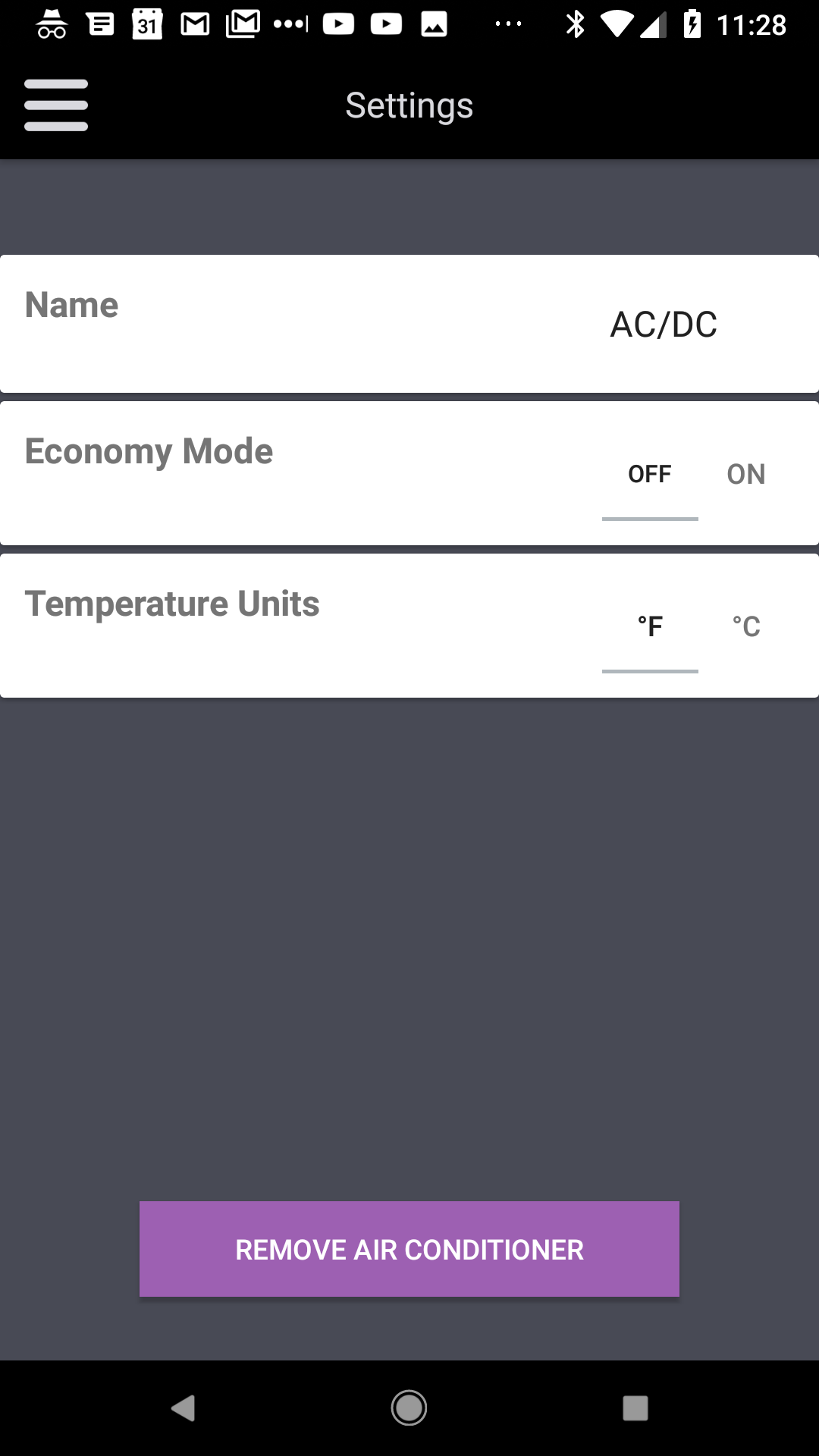
There are no special configuration elements for this screen.

## DeviceScreen

The base class for screens dealing with devices, the DeviceScreen will automatically set up its internal device object when launched with a DSN in its configuration, and contains methods to rename and unregister the device.

The Screen handles the “unregister” menu command and will prompt the user to unregister the device.

## DeviceSettingsScreen

Similar to the ApplicationSettingsScreen, the DeviceSettingsScreen allows the user to set device-specific configuration options. These settings are tied to property values from the device’s configuration. The properties to use as settings are defined in the screen’s configuration, and the screen may display a button to unregister the device as well.

### Configuration

The Screen’s configuration as shown in the screenshot above specifies that an unregister button should be shown (“remove air conditioner”) and that the t\_eco and t\_temptype properties should be used as configuration options for this device. When the user changes a setting, the associated property is changed on the device.

{

"class": "DeviceSettingsScreen",

"name": "ac\_settings",

"title": "ac\_settings",

"extras": {

"unregisterButton": true,

"propertySettings": [

"t\_eco",

"t\_temptype"

]

}

}

The device’s configuration defines those properties as:

{

"name": "t\_eco",

"notify": true,

"schedule": true,

"roles": ["ecoMode"],

"actions": [

{

"name": "eco\_off",

"value": 0

},

{

"name": "eco\_on",

"value": 1

}

]

},

{

"name": "t\_temptype",

"notify": false,

"schedule": false,

"roles": ["temperatureUnits"],

"actions": [

{

"name": "F",

"value": 0

},

{

"name": "C",

"value": 1

}

]

}

## DeviceSwipeContainer

The DeviceSwipeContainer is an alternative to the AllDevicesScreen. Rather than showing the devices in a list, the DeviceSwipeContainer displays the detailScreen of each registered device as a set of screens that may be swiped left-to-right to change between them.

No additional configuration is required for this screen.

## HTMLViewScreen

The HTMLViewScreen may be used to show HTML content, either locally from a resource or remotely via a URL. The screen may be presented as an application-contained WebView, or may launch the page in an external browser.

### Configuration

|  |  |
| --- | --- |
| htmlResource | ID for the resource containing the HTML contents |
| htmlUrl | URL for the external HTML resource |
| launchInBrowser | If true, will launch an external browser to view the content. Otherwise content will be shown in an internal WebView. |

External browser pointed to https://developer.aylanetworks.com

{

"class": "HTMLViewScreen",

"name": "getting\_started",

"title": "getting\_started",

"icon": "user\_manual\_icon",

"extras": {"htmlResource": null,"htmlUrl": "https://developer.aylanetworks.com", "launchInBrowser": true}

}

External browser pointed to static content from application resources

{

"class": "HTMLViewScreen",

"name": "user\_manual",

"title": "user\_manual",

"icon": "user\_manual\_icon",

"extras": {"htmlResource": "user\_manual\_content","htmlUrl": "", "launchInBrowser": true}

}

Internal WebView displaying https://www.aylanetworks.com/privacy-policy:

{

"class": "HTMLViewScreen",

"name": "privacy\_policy",

"title": "privacy\_policy",

"icon": "privacy\_policy\_icon",

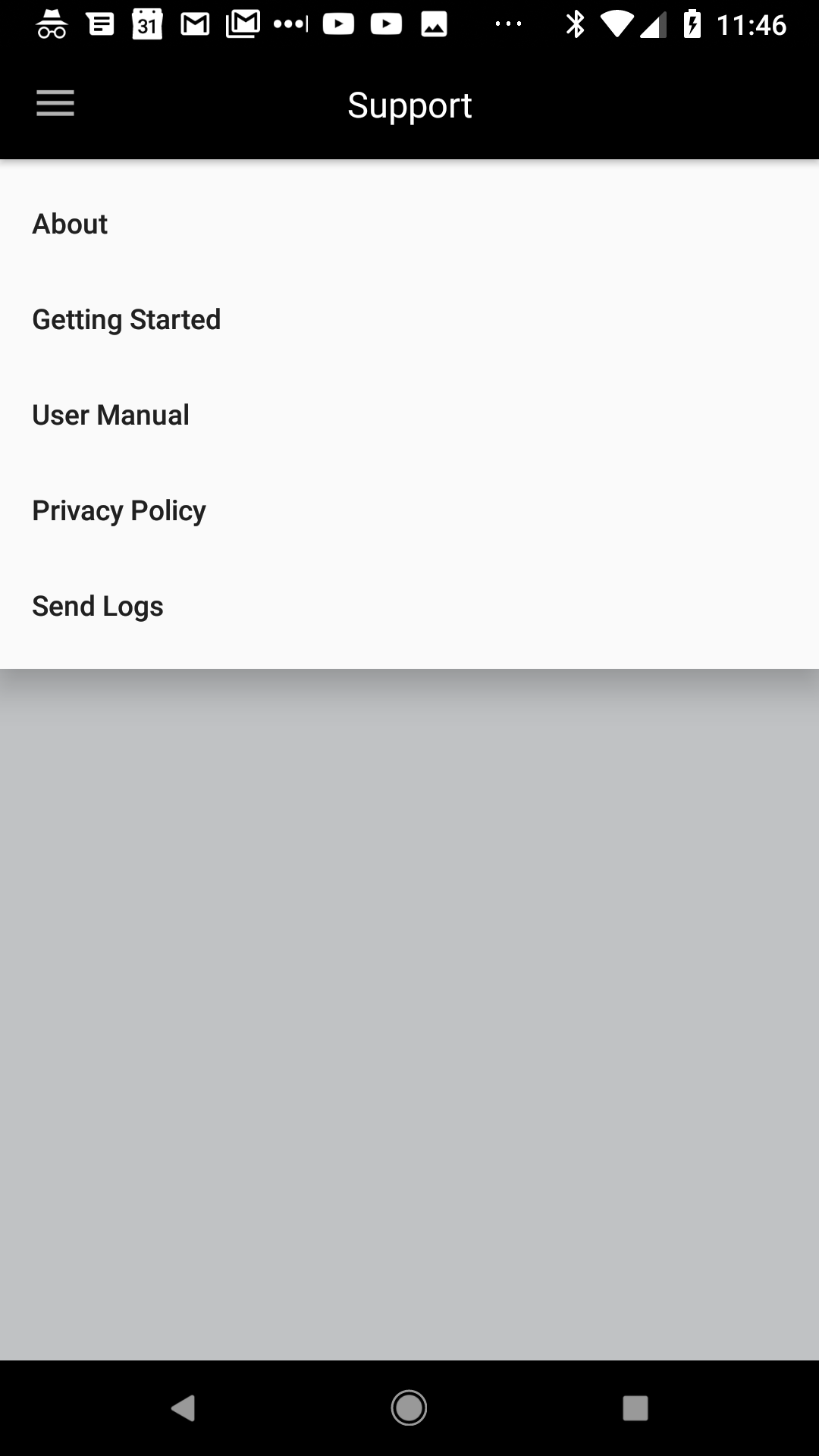
"extras": {"htmlResource": "privacy\_policy\_content","htmlUrl": "https://www.aylanetworks.com/privacy-policy",

"launchInBrowser": true}

}

## MenuScreen

This screen displays the contents of a menu from the configuration. Rather than displaying a context menu, developers may instead launch a MenuScreen to display the contents of a menu for the user to select from. This is useful to allow easy creation of drill-down menus from the configuration file alone.



### Configuration

The MenuScreen configuration for the example above points to the support\_screen\_menu:

{

"class": "MenuScreen",

"name": "support",

"icon": "ic\_support",

"title": "support\_screen\_title",

"extras": {"menu": "support\_screen\_menu"}

}

The support\_screen\_menu is defined in the configuration as:

{

"name": "support\_screen\_menu",

"items": [

"about\_screen",

"getting\_started",

"user\_manual",

"privacy\_policy",

"send\_logs"

]

}

## NoDevicesScreen

The NoDevicesScreen is the default screen shown if the user has signed in and no devices were found registered to the account.

This screen may be replaced with any screen by changing the **noDevicesScreen** value in the **userExperience** section of the configuration. The default screen simply shows a message and an optional button linking to a Screen, most often the *SetupWizardScreen* to discover and register new devices.

### Configuration

{

"class": "NoDevicesScreen",

"name": "no\_devices",

"title": "no\_devices",

"extras": {

"message": "no\_ac\_devices\_message",

"buttonText": "add\_ac\_button\_text",

"buttonTarget": "setup\_wizard"

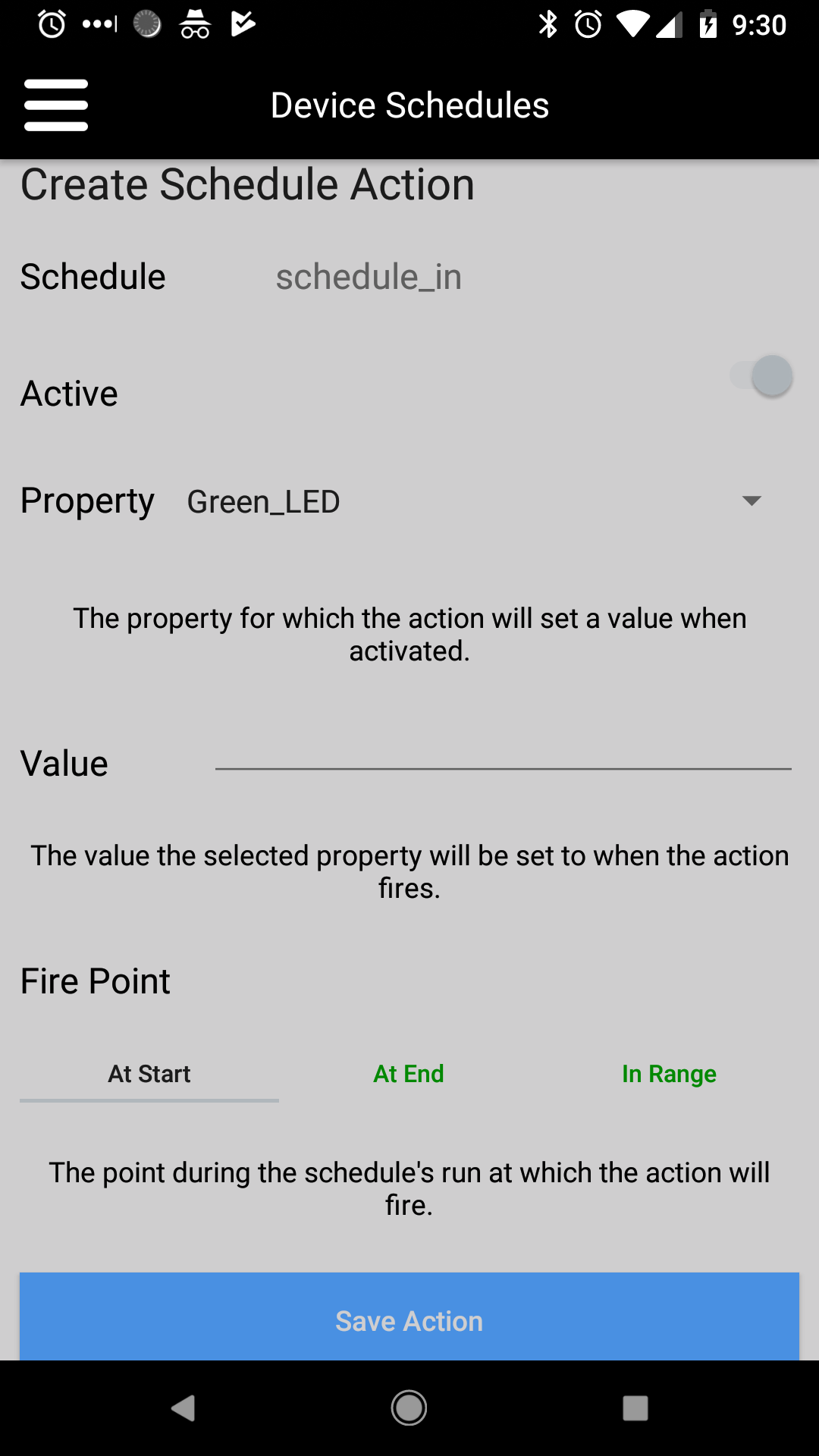
}

}

|  |  |
| --- | --- |
| message | String resource for a message displayed to the user |
| buttonText | String resource for text displayed on the button |
| buttonTarget | Screen name to launch when the button is tapped |

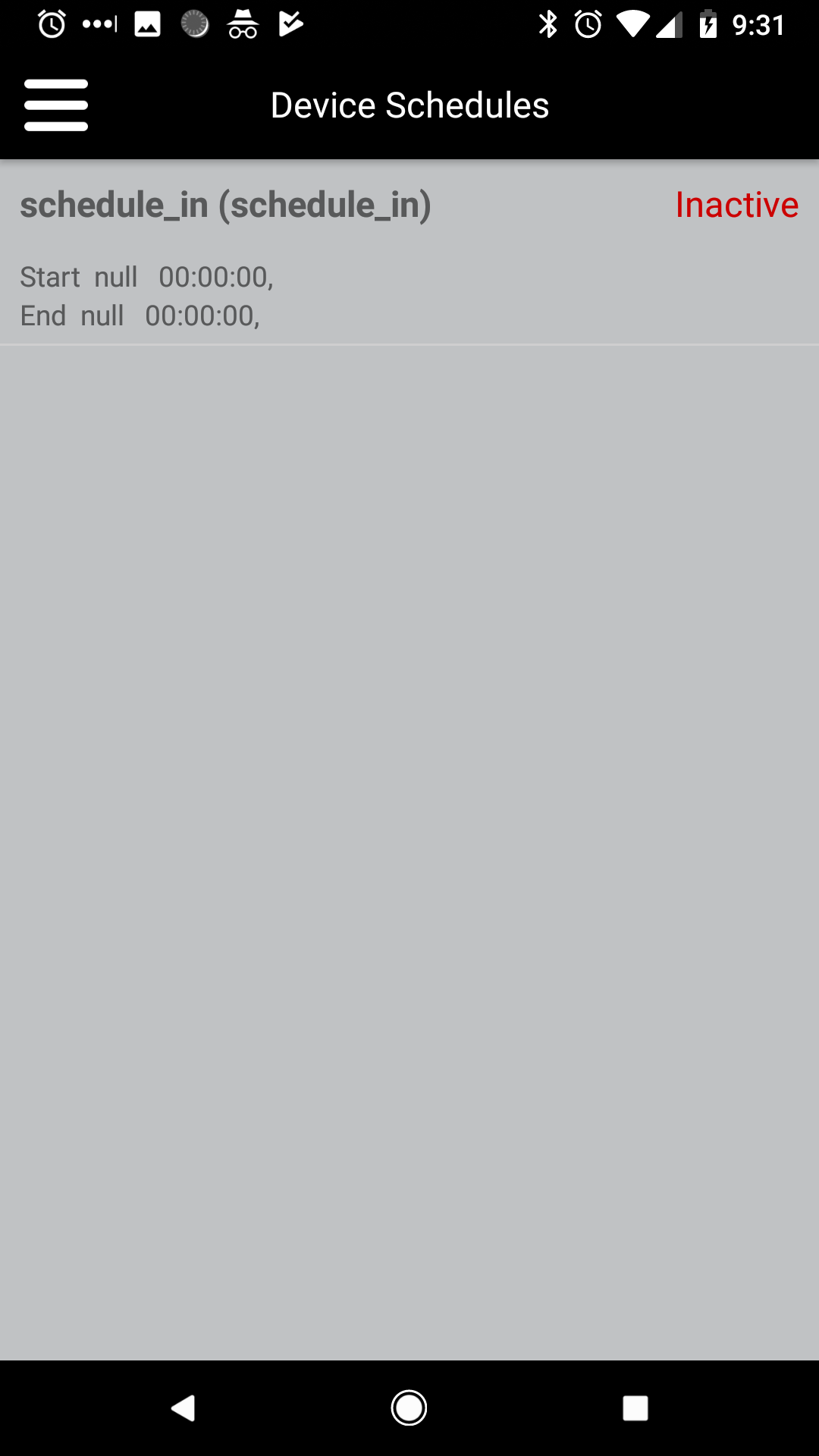
## ScheduleActionScreen

The ScheduleActionScreen is usually launched by the ScheduleListScreen to create or edit actions. The actions are defined in the **managedProperty** section of the device’s configuration.



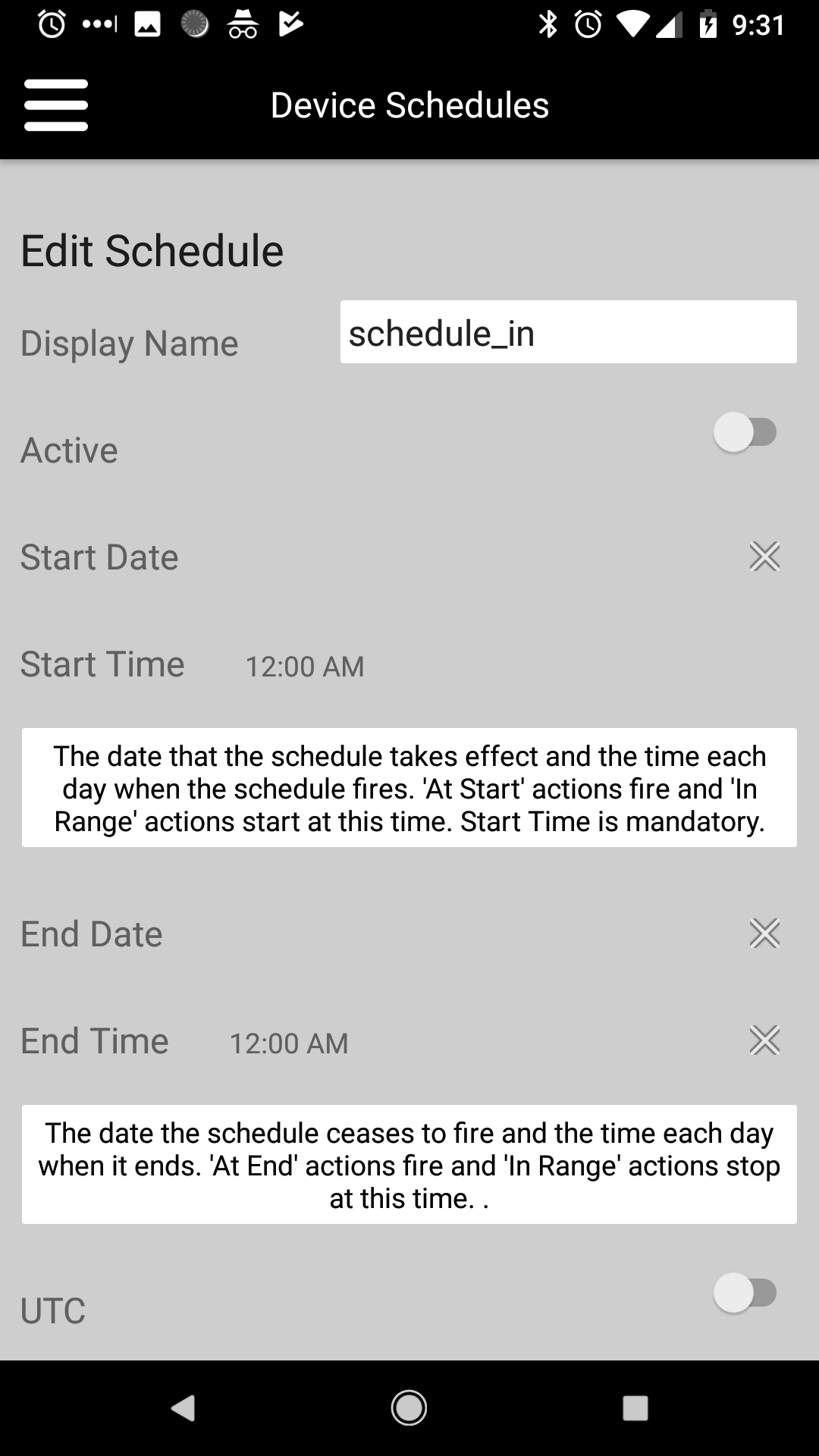
## ScheduleListScreen

The ScheduleListScreen displays a list of schedules for a device.



## ScheduleScreen

The ScheduleScreen provides an interface to edit a schedule’s settings. It contains a link to the ScheduleActionScreen to configure actions.



## SetupWizardScreen

The SetupWizardScreen provides the user with step-by-step guidance through the device onboarding process. The onboarding flow can be summarized as:

1. Determine the type of device the user wants to set up

2. Find the device in a scan. This includes scanning for WiFi access points, local BLE devices or requesting a gateway to discover nodes. The type of scan performed will depend on the type of the selected device.

3. Connect to the device and obtain additional information about it

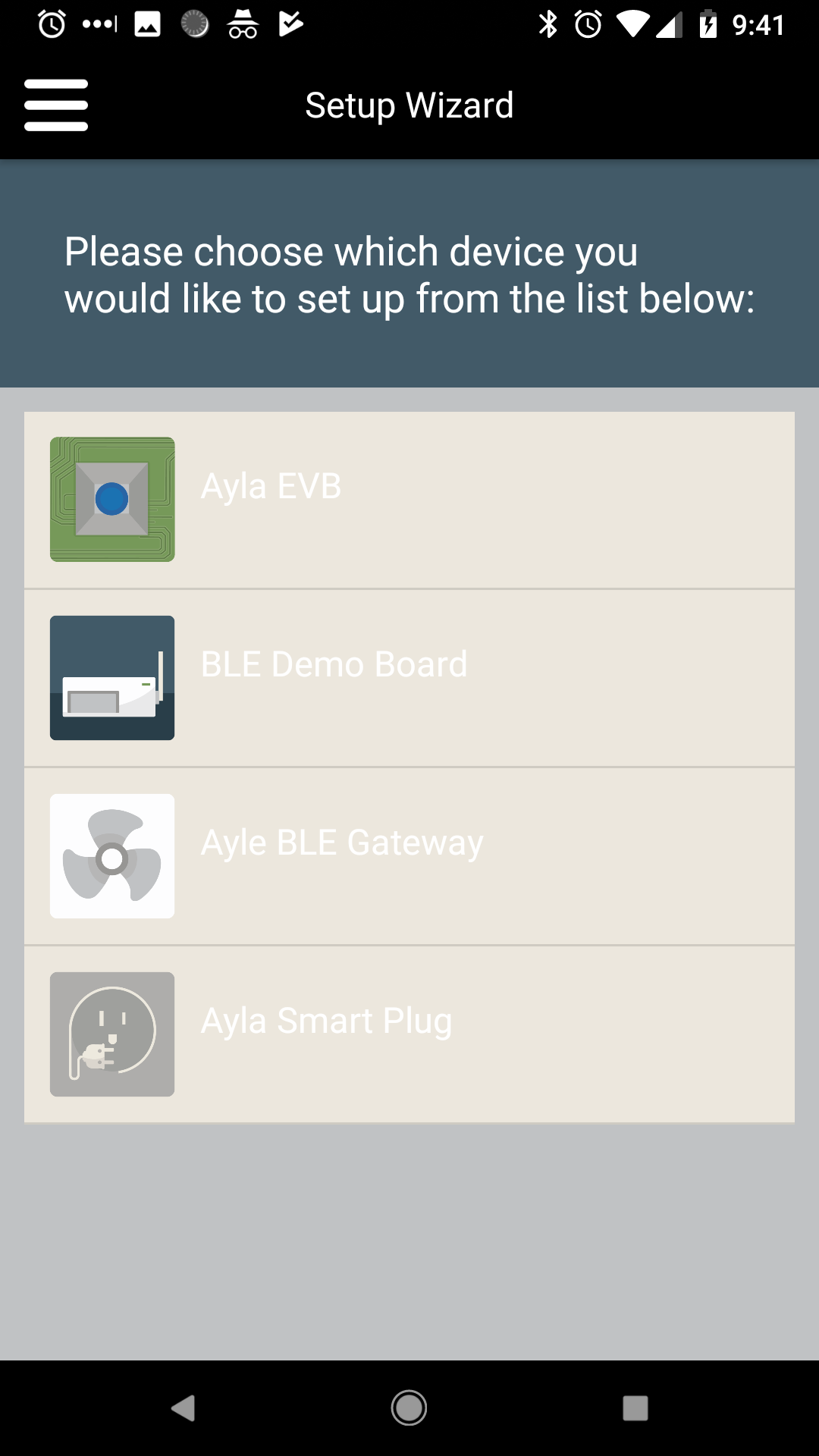
4. Prompt the user for WiFi credentials or a passkey, if required

5. Connect the device to the local WiFi network (for WiFi devices)

6. Obtain registration information from the user or device, depending on the type of registration

7. Register the device with the Ayla cloud

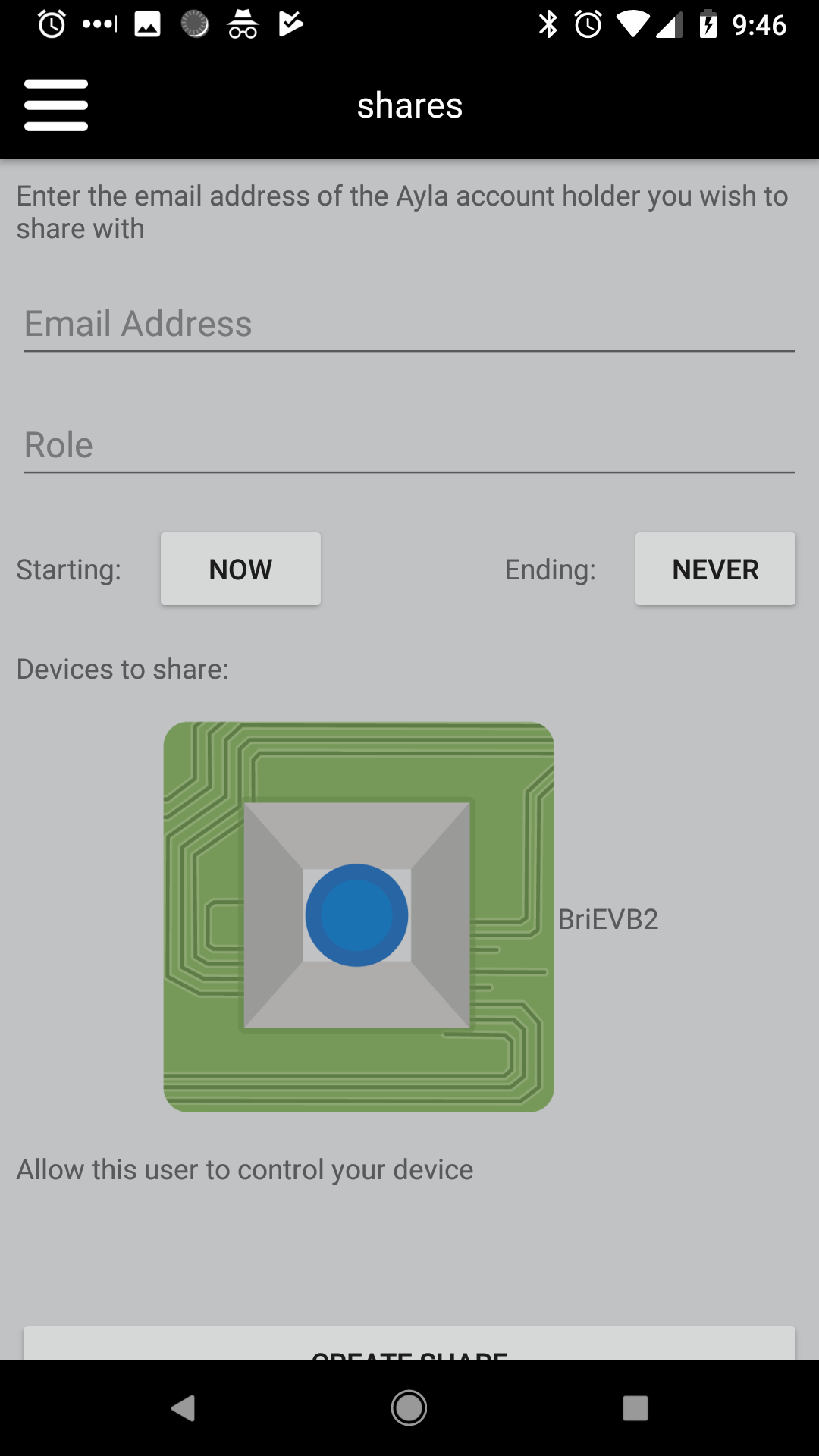
8. Prompt the user to customize the device by naming it, setting its location information, etc.



### Configuration

|  |  |
| --- | --- |
| prerequisiteText | String resource for text displayed to the user as a prerequisite step. This is optional, and if not set no text will be displayed. For example, “Ensure your device is powered on and the green light is blinking” |
| prerequisiteHelp | String resource for text displayed to the user as “help” for the prerequisite. If not present, no help will be offered for the prerequisite steps. For example, “Unplug your device and wait 5 seconds. Then plug it in again and re-try.” |
| mode | If set to “register”, will skip steps up to registration. This is useful to register devices that are already on the network and do not need WiFi configuration. |

## ShareDevicesScreen

This screen provides an interface for the user to share a device with another person.

## ShareListScreen

The ShareListScreen shows the set of devices shared with others or shared by others to the account holder.

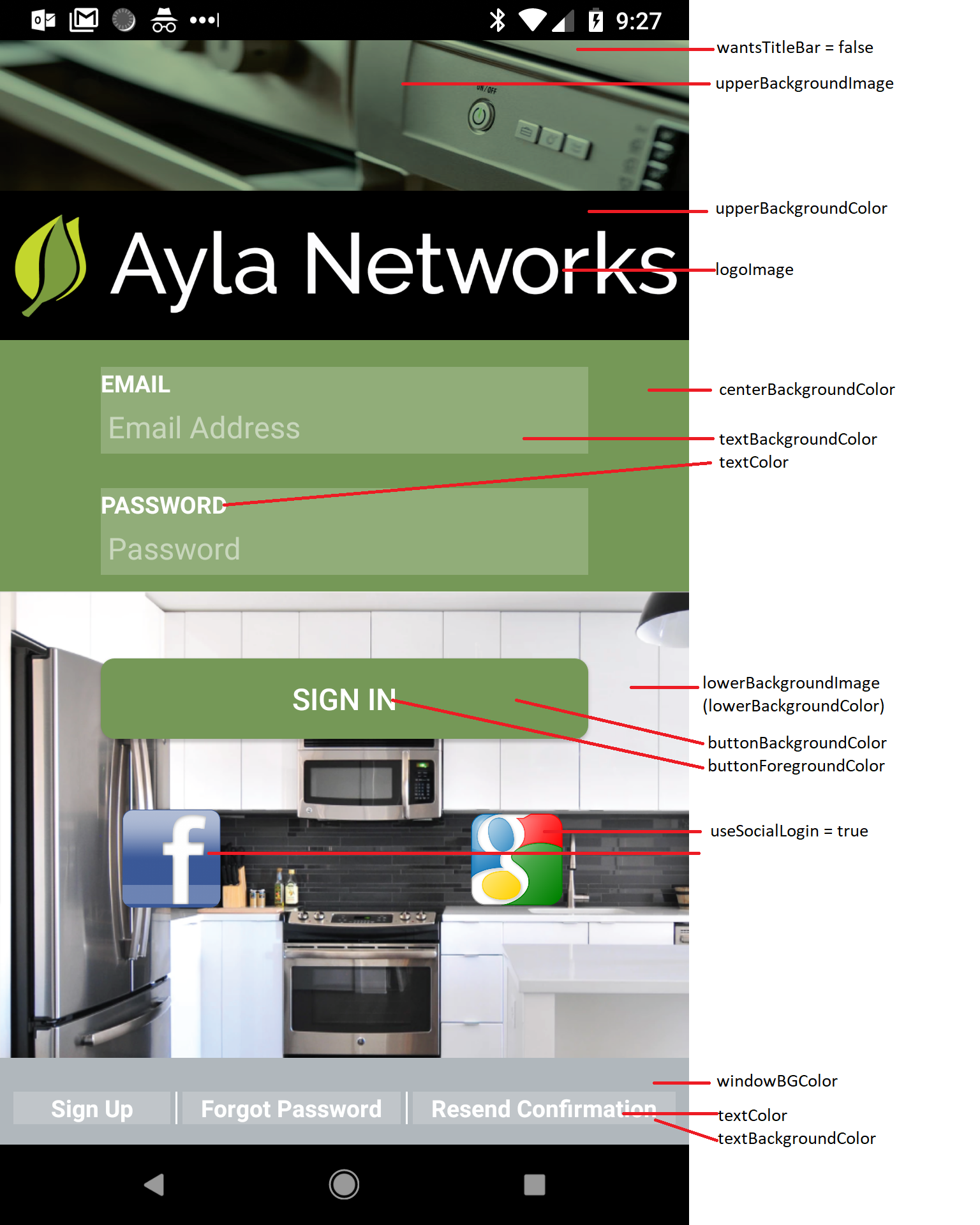
## 

## SignInScreen

The SignInScreen is the first screen shown to the user. It provides fields for the user to enter in an email address and password, as well as icons to sign in using Google or Facebook authentication.

Additionally, the screen contains links to assist users with the sign-in or sign-up process or reset a forgotten password.

If privacy settings have been configured for the application, this screen will prompt the user for consent and store the result in user settings. The text displayed to the user can be configured within the screen’s “extras” as described below.



### Configuration

|  |  |
| --- | --- |
| useSocialLogin | If true, will offer Google and Facebook sign-in options. Otherwise these will not be shown. |
| logoImage | Image ID for the logo shown at the top of the screen |
| upperBackgroundImage | Image ID for the upper image |
| upperBackgroundColor | Color for the upper image and logo backgrounds (may be hidden by images) |
| lowerBackgroundImage | Image ID for the lower image |
| lowerBackgroundColor | Color for the lower image backgrounds (may be hidden by images) |
| centerBackgroundColor | Color for the background between upper and lower images |
| requestDataGrantsAlertTitle | String reference for the title of the message box asking for user consent for data sharing |
| requestDataGrantsAlertMessage | String reference for the message displayed to users asking for consent to share data |

# Controls

This section lists the various Controls that are supplied with Mobile Foundry along with Control-specific configuration information.

## ActionPicker

The ActionPicker control is used to select an action for a particular property in a pop-up menu. When the ActionPicker control is associated with a Property, the Actions for that property are presented as options in a pop-up list when tapped. The current selection is displayed when the pop-up is not active. Selecting an item from the list will set the property to the value associated with the selected action.

### Configuration

The ActionPicker control requires at a minimum that the associated property is defined. Additional configuration options include the text color and background color for the items.

{

"name": "ac\_mode\_picker",

"class": "ActionPicker",

"extras": {

"property": "t\_work\_mode",

"textColor": "#ffffff",

"backgroundColor": "#00000000"

}

},

The contents of the control are determined by the Actions defined for the selected property. The example above used a property with actions set up as follows:

"name": "t\_work\_mode",

"roles": ["modeControl"],

"notify": true,

"schedule": true,

"actions": [

{

"name": "mode\_fan",

"icon": "mode\_fan",

"value": 0

},

{

"name": "mode\_heat",

"icon": "mode\_heat",

"value": 1

},

{

"name": "mode\_cool",

"icon": "mode\_cool",

"value": 2

},

{

"name": "mode\_dehumidify",

"icon": "mode\_dehumidify",

"value": 3

},

{

"name": "mode\_auto",

"icon": "mode\_auto",

"value": 4

}

]

},

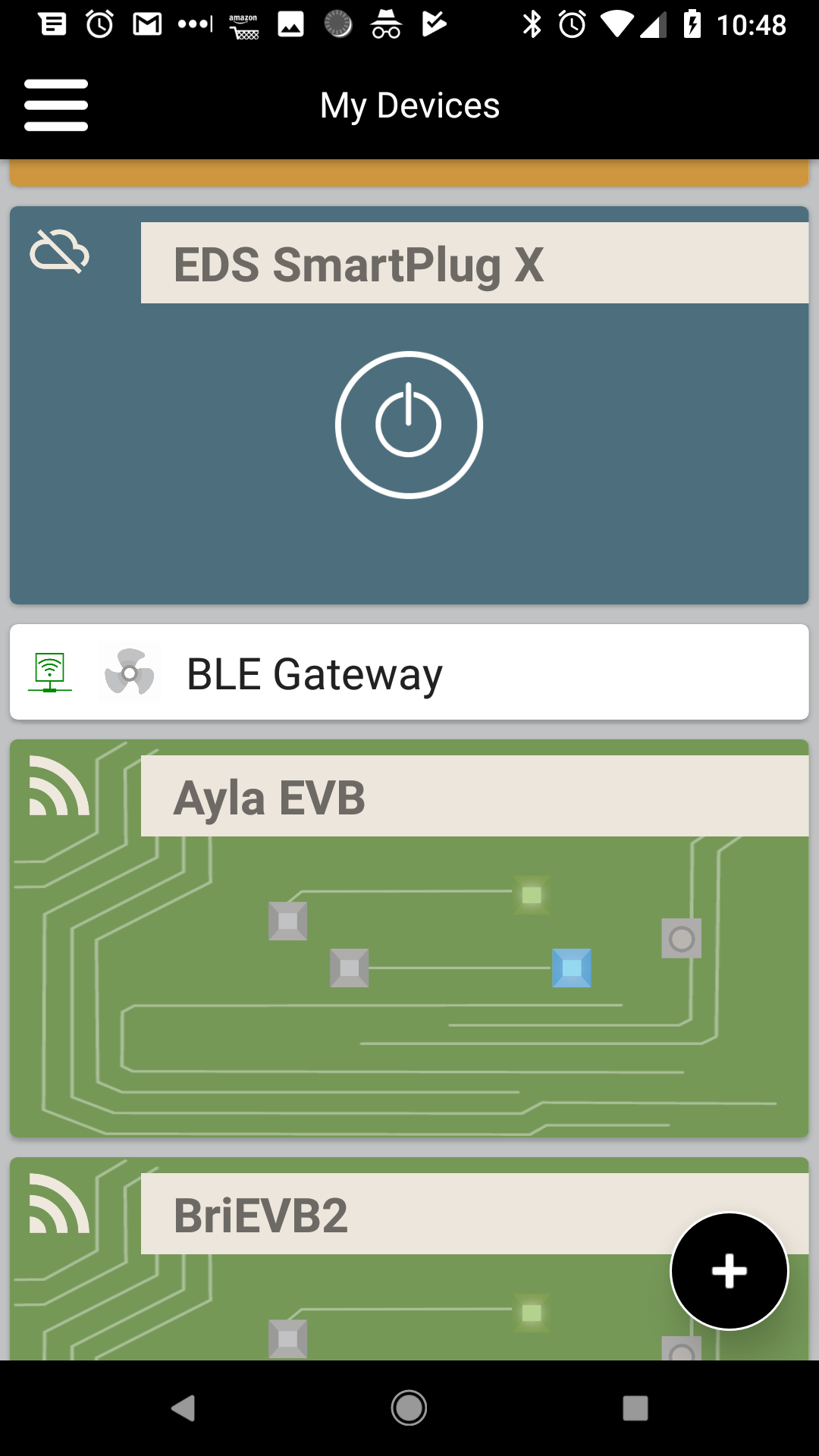
## EVBControl

The EVBControl is an example of a DeviceControl created for Ayla’s development evaluation board. This control is displayed within the AllDevicesScreen. Each device may have its own control that will be displayed in the AllDevicesScreen to allow immediate interaction with all devices in the list.

## 

## OnOffControl

The OnOffControl is used frequently throughout Mobile Foundry applications. Tied to a property, the control displays an image for the property’s state and toggles that state when tapped. The property and on / off images are configurable.



The above screenshot shows several on / off controls. The “EDS SmartPlug X” cell is an instance of an “OnOffDeviceControl” and contains an on / off control as the large button. The two AylaEVBControl cells each contain three OnOffControls: one for each of the Blue and Green LEDs, and a third set as read-only to show the state of the blue button.

### Configuration

The OnOffControls used in the above example are defined as follows:

"controls": [

{

"name": "led\_light\_green",

"class": "OnOffControl",

"extras": {

"onImage": "green\_led\_on",

"offImage": "green\_led\_off"

}

},

{

"name": "led\_light\_blue",

"class": "OnOffControl",

"extras": {

"onImage": "blue\_led\_on",

"offImage": "blue\_led\_off"

}

},

{

"name": "blue\_button",

"class": "OnOffControl",

"extras": {

"onImage": "blue\_button\_on",

"offImage": "blue\_button\_off",

"readOnly": true

}

},

{

"name": "generic\_switch",

"class": "OnOffControl",

"extras": {

"onImage": "bulb\_on",

"offImage": "bulb\_off"

}

}

|  |  |
| --- | --- |
| onImage | Image displayed when the property is non-zero |
| offImage | Image displayed when the property is zero |
| readOnly | Set to true to disable changing the property value on tap |

## TempSetpointControl

The TempSetpointControl is used to set a temperature, most often used in HVAC units.

The control displays the temperature value in the center and contains arrows either to the top and bottom, or left and right depending on the control orientation.

This control looks for a role called “tempSetpoint” to find the appropriate property to use for the temperature value.

### Configuration

{

"name": "temp\_setpoint\_control\_details",

"class": "TempSetpointControl",

"extras": {

"circleColor": "#00000000",

"arrowColor": "#ffffff",

"horizontal": false,

"hasDots": false

}

}

|  |  |
| --- | --- |
| circleColor | Color of the circle drawn around the current temperature. Set to transparent if it should not be shown |
| arrowColor | Color of the arrow buttons used to change temperature |
| horizontal | If true, will orient the control horizontally. Otherwise the control will be oriented vertically. |
| hasDots | If true, will draw three dots between the temperature and arrows |