Foundry/FastTrack Configuration Specification

metric

Overview

Mobile Foundry is a mobile application development platform that allows developers to create mobile Ayla-connected applications quickly and efficiently. Rather than re-writing the same screens, controls and logic for each application, Mobile Foundry allows developers to choose and configure the screens and controls best suited for the project.

This document describes the Mobile Foundry configuration file format and how Mobile Foundry uses it to build and manage an Ayla-connected mobile application.

Configuration Structure

The Mobile Foundry configuration file is a JSON document containing information specific to the product being designed. The supported device types, application features, styles and behavior are all declared in the Mobile Foundry configuration file.

Resource References

Resources such as strings and images are referenced throughout the Mobile Foundry configuration file. To achieve cross-platform compatibility, it is important that resources are named identically in both iOS and Android projects.

Resources referenced in the configuration file are either used as keys in a string lookup table (in the case of string resources) or filenames without extensions in the case of image resources.

For example, a device might be defined in the configuration with the “name” field set to “outlet_name”, and the “icon” field set to “outlet_icon”.

The string table in each application (Localizable.strings in iOS or strings.xml in Android) would contain the display name for that device (“Smart Outlet”, perhaps) in the string table with the key “outlet_name”.

Each application also would include an image resource named “outlet_icon.png” or “outlet_icon.jpg” or any other supported image format.

This allows Mobile Foundry to properly show the correct name and an appropriate image / icon for each device.

Root

The root element of the Mobile Foundry configuration file contains information about the configuration and application itself. Items in bold are required.

Name	Contents
applicationIdentifier	A unique identifier (string) for the application. In Android applications, this value is used for the application ID. In iOS applications, this value represents the bundle ID for the application
packages	On Android, an array of Java package names to search when looking for Sepia classes
bundles	On iOS, an array of bundle IDs to search
name	Name for the configuration, used only to identify different configurations
version	Version of the configuration. Used only to identify different configuration versions.
devices	Array of device objects. Device objects are used to describe the features and details of Ayla IoT devices controlled by this application
userExperience	Section describing the application flow, supported app features, menu system, colors, etc.
sdkConfig	Configuration information for the Ayla Mobile SDK

sdkConfig

The sdkConfig section contains information required by the Ayla Mobile SDK for operation. This includes the appId / appSecret provided to the app developer by Ayla Networks, as well as information regarding the service type / location the app should connect to.

Name	Contents
appId	Application identifier provided by Ayla Networks
appSecret	Application secret provided by Ayla Networks
debugBuildAppId	Application identifier provided by Ayla Networks for debug builds alone and it is implemented for iOS platform only. This will help in testing APNS Sandbox and Production cert.
debugBuildAppSecret	Application secret provided by Ayla Networks for debug builds alone and it is implemented for iOS platform only. This will help in testing APNS Sandbox and Production cert.
allowMobileDSS	If true, the SDK will use the mobile Device Subscription Service (websockets) to keep the device status up to date. If false, the SDK will instead poll the service to maintain device status
xPlatformId	Used internally as a key into user datum shared between iOS and Android implementations
serviceType	Development | Staging | Field (Ayla service type)

serviceLocation	USA | China | Europe (Ayla server locations)
allowOfflineUse	If true, will allow sign-in while the Ayla service is not reachable. Device status may not be accurate in this mode. while archiving AylaDevice and AylaProperty must support NSSecureCoding to unarchive data from iOS 11.0 and above, if AylaDevice and AylaProperty is subclassed make sure to support NSSecureCoding.
defaultNetworkTimeoutMs	Default timeout for network operations in ms.
consoleLogLevel	Level of messages printed to the console: Verbose | Debug | Info | Warning | Error | None
FileLogLevel	Level of messages written to log files: Verbose | Debug | Info | Warning | Error | None
metricSampleRatio	It's sample ratio of metrics applied only for CloudLatency metrics. Other metrics are bypassed with as Cloud Latency metric's are more than expected count. type is of Int and default value is set to **10** in SDK

Privacy

The Privacy section contains settings that pertain to user acceptance of privacy policies, such as granting permission to send logs or metrics to the cloud service.

Name	Contents
UserDataGrants	Array of strings indicating what grants the user agrees to when accepting to share information. Current values are: **metricsService** – grant allowing logs and metrics to be sent to the service **uncaughtException** – grant allowing uncaught exceptions to be logged and sent to the service

Devices

The Devices section contains an array of Device objects. A Device object contains information that allows Mobile Foundry to make intelligent decisions about how devices are displayed and interacted with. Each device supported by the application must have an entry in the Device array.

The Device entries contain the following fields:

Name	Contents
class	Name of the class that should be created to represent this device in the system. Classes must inherit from AylaDevice, and must implement SepiaDeviceInterface.
icon	Resource name for an icon / image to be displayed when referring to this device (image resource)
name	Resource name for the name of the device (string resource)
model	Model of this device, used by Mobile Foundry to identify device types
oemModel	OEM model of this device, used by Mobile Foundry to identify device types
detailScreen	Screen reference indicating the screen that should be shown when the user wants to interact with this device directly

offlineScreen	Screen reference indicating the screen that should be shown when user would want to interact device directly but device is offline.
registrationType	Ayla registration type. Must be one of: Same-LAN, Button-Push, AP-Mode, Display, Dsn, Node or Local. See the Ayla documentation on device registration for more information regarding the specific registration modes.
scheduleScreen	Screen reference indication the screen that should be shown when the user wants to create or edit schedules for this device. If this field is null or does not exist, schedules will not be made available for this device.
ssidRegex	Regular expression used to locate a device before it has joined the user's WiFi network. This expression should be tailored to match the SSID broadcast by the device when in AP mode.
setup	Contains instruction screen configurations for device setup and registration (on-boarding).
managedProperties	Array of objects describing the properties of the device that should be used by Mobile Foundry for user interaction. These objects are described in more detail in the next section.
deviceControl	A reference to a Device Control that should be used to represent the device in its entirety
features	Features are reared to wifi selection screen where developer can configure what feature end user can utile to select network. Ex. "features": ["discoveredNetworks", "others", "scan"]. *discoveredNetworks* => List all wifi which are discoed by the device *others* => Option to enter hidden network *scan* => Option to allow user to scan wifi said and password from QR code
onBoardingFlow	Specify the flow to onboarding flow a device, default is WiFi AP mode. Supported "wifi" & "ble".
developmentDevice	A tag used to Indicate the device is used for development or is in a pre-release development stage. The device can be hidden or exposed by including the hideDevDevices tag on supporting screens.

Each supported device type must have a single entry in the Devices array, and must be able to be uniquely identified by the model and / or oemModel. When properly configured, each device provided to Mobile Foundry via the Ayla Device Manager will be of the class type specified in the device configuration.

The dynamic nature of device creation in Mobile Foundry allows for user-defined device types to be created and managed by the Ayla SDK and Mobile Foundry systems. Custom devices must inherit at some level from AylaDevice (classes such as AylaDeviceGateway, AylaDeviceNode or AylaLocalDevice all derive from AylaDevice and can therefore be used as well) and support the SepiaDeviceInterface (interface used for UI interactions and other information needed by Mobile Foundry).

Managed Properties

A device on its own cannot do anything useful. Devices have "properties", which can be queried or modified to determine or change the state of a device. A simple wall switch, for example, might have a single property called "on_off", which when set to 1 means the switch is on. An HVAC system on the other hand, might have many properties used to control the current setpoint, fan speed, read the current temperature, etc.

Mobile Foundry allows developers to describe the properties that will be managed by the application in the configuration file. Each property may be named, assigned "actions" that can be performed by the user, or be assigned a "role" known to the application (for example, the "setpoint" property of a thermostat might be given the "tempSetpoint" role, which is used by the HVAC control screen to know which property is used to control the setpoint).

Properties that are not listed in this section will not automatically be kept up-to-date by the application. Before a property that is not managed is used, it should be first fetched from the service by calling the *fetchProperty()* method on the device.

Name	Contents
name	Name of the property as defined in the device template
control	Reference to an Mobile Foundry Property Control that should be used to represent this property
notify	True if this property can be used for notifications
schedule	True if this property can be used for schedules
actions	Array of Action objects that map property values or ranges to textual descriptions and icons (see below)
units	Field used by some properties to indicate the unit type (e.g. “C” or “F” for temperature, “in” or “mm” for length, etc.)
readOnly	True if this property may not be modified by the user

Actions

Actions are objects that provide even more detailed information about a managed property and what it means. They are the link between a property’s value and what it means to the end user. For example, a fan control property might have several values representing speed. Each of these values would be represented in the Mobile Foundry configuration as an Action, so when presented with a control to set the speed of the fan, the user would see the action names and icons rather than raw property values:

Value	Text	Icon
0	“Off”	Fan off icon
1	“Silent	Fan silent icon
2	“Low”	Fan medium icon
3	“Medium”	Fan high icon
4	“High”	Fan high icon

```
{
        "name": "t_fan_speed",
        "roles": ["fanControl"],
        "notify": true,
        "actions": [
                {
                        "name": "fan_off",
                        "icon": "fan_off_icon",
                        "value": 0
                },
                {
                        "name": "fan_silent",
                        "icon": "fan_silent",
                        "value": 1
                },
                {
                        "name": "fan_low",
                        "icon": "fan_low",
                        "value": 2
                },
                {
                        "name": "fan_medium",
                        "icon": "fan_medium",
                        "value": 3
                },
                {
                        "name": "fan_high",
                        "icon": "fan_high",
                        "value": 4
                }
        ]
}
```

When configured to use the ActionPicker, the user will be presented with a menu composed of the fan speed names and associated icons. This allows the application to set the appropriate property value based on the user selection.

Setup

setup contains instruction screen configuartions.

Name	Contents
setupScreen	Name of the screen to start on-boarding process.
setupInstructions	Array of instructions contains instruction message and descriptiveImage.
setupConfirmScreen	Instruction confirmation screen before connecting to device. Developer can configure screen title, instruction label and descriptiveImage.
setupConfirmHelpMessage	Instruction confirmation screen help screen label text configuration

setupDeviceConnectingScreen	Setup screen when device connecting to device.
setupWifiSelectionScreen	Setup wifi selection screen configuration, a title and instruction message can be configured.
setupWifiSelectionScreenHelpScreenLabel	Setup wifi selection screen help screen label text configuration
setupConnectivityHelpScreen	Setup wifi selection Help screen configuration, developer can configure descriptive image and multiple institution to render on screen.
setupProgressScreen	Setup progress screen when device is connecting and registering to cloud. All tasks are listed with showing progress of each task.
setupSuccessfullScreen	Setup Successful screen configuration, developer can configure screen title, descriptive image and instruction message to display to end user.
setupParams	Setup params contains onboarding device connection timeouts.
offlineHelpScreen	Offline Help screen is optional. Screen reference indicating the screen that should be shown when user would want to interact device directly but device is offline.
resetScreen	Reset screen is optional for Foundry where as for FastTrack app it's mandatory to mention. if reset screen is empty, then behavior would be to display error message and take user back to home screen. else if reset is mentioned with screen name, then the behavior would be to display error message and take user to given screen. else default would be to display error message and take user to Foundry Reset screen.

```
"setup": {
    "setupScreen": "fasttrack_setup_assist_screen",
    "setupInstructions": [
        {
            "instructionMessage": "SetupScreen.instructionScreen.
message1",
            "descriptiveImage": "setup_device_1"
        },
        {
            "instructionMessage": "SetupScreen.instructionScreen.
message1",
            "descriptiveImage": "setup_device_1"
        },
        {
            "instructionMessage": "SetupScreen.instructionScreen.
message1",
            "descriptiveImage": "setup_device_1"
        }
    ],
    "setupConfirmScreen":
    {
        "title": "SetupScreen.ConfirmDeviceOnScreen.title",
        "instructionMessage": "SetupScreen.ConfirmDeviceOnScreen.
```

```
instruction",
        "descriptiveImage": "setup_device_2"
    },
    "setupConfirmHelpMessage": "SetupScreen.ConfirmDeviceOnScreen.
instructionConfirmation",
    "setupDeviceHelpScreen":
    {
        "instructionMessage": "SetupScreen.deviceInstructionScreen.
instruction",
        "descriptiveImage": "setup_device_3"
    },
    "setupDeviceConnectingScreen":
    {
        "title": "SetupScreen.connectDeviceScreen.title",
        "instructionMessage": "SetupScreen.connectDeviceScreen.
instruction",
        "descriptiveImage": "setup_device_4"

    },
    "setupWifiSelectionScreen":
    {
        "title": "guided_setup_wifi_selection",
        "instructionMessage": "guided_setup_wifi_selection_instructions"
    },
    "setupWifiSelectionScreenHelpScreenLabel":
"guided_setup_wifi_selection.cannotFindNetwork",
    "setupConnectivityHelpScreen": {
        "image": "connectivityHelpImage",
        "instructions": [
          {
            "instructionMessage": "guided_setup_help_screen_message_1"
          },
          {
            "instructionMessage": "guided_setup_help_screen_message_2"
          }
        ]
    },
    "setupProgressScreen":
    {
        "title": "ModalProgressScreen.title",
        "instructionMessage": "ModalaProgressScreen.header",
        "descriptiveImage": "setup_device_6"
    },
    "setupSuccessfullScreen": {
        "title": "RegistrationSuccessFullScreen.title",
        "instructionMessage": "RegistrationSuccessFullScreen.
successMessage",
        "descriptiveImage": "registration_successfull"
    },
    "setupParams": {
```

```
        "scanForDeviceTimeout": 15,
        "connectToNewDeviceTimeout": 20,
        "connectDeviceToServiceTimeout": 60,
        "reconnectToOriginalNetworkTimeout": 60,
        "confirmDeviceConnectedToCloudTimeout": 30
    },
    "offlineHelpScreen": {
        "image": "offlineHelpImage",
        "instructions": [
          {
              "instructionMessage": "DeviceOfflineHelpScreen.
instruction1"
          },
          {
              "instructionMessage": "DeviceOfflineHelpScreen.
instruction2"
          },
          {
              "instructionMessage": "DeviceOfflineHelpScreen.
instruction3"
          }
        ]
    },
    "resetScreen": ""
}
```

User Experience

The User Experience section describes the overall look and feel of the application. Mobile Foundry applications divide the interface into Screens, which are essentially UIViewControllers in iOS and Fragments in Android. Screens may contain Controls, which can be tied to either a Device or a Property of that device.

The User Experience configuration allows developers to add and remove screens, update the menus and change the overall look and feel of the application.

Name	Contents
menus	Array of menu objects, described in a later section
colors	Array of application color definitions
textStyles	Array of application font definitions
defaultConfig	Default configuration options for the application, including colors, title bar style, etc. Described in more detail in a later section
controls	Definitions for UI controls used in the application.
drawer	Configuration for the navigation drawer, if present
screens	Definitions for screens used throughout the application
templates	Email template definitions used for email / sms notification formats
signInScreen	Reference to the screen used to sign in the user

homeScreen	Reference to the screen shown after the user signs in, the “root” screen.
noDevicesScreen	Screen shown when no devices have been added to the account
applicationSettings	Application configuration defaults that may be modified by the user in a Settings screen, for example whether the application should use C or F as temperature units when displaying temperature to the end-user.
singleDeviceDetailsHomeScreen	
themes	Array of Application supported Themes

Screens

A Screen represents a particular navigation destination in Mobile Foundry. When the application is first started, the user is presented with a Screen to sign in. Once signed in, the user is presented with the “home screen”, the springboard for all application navigation. Each device might have its own Screen that is displayed when the user taps on a device from a list. A Screen is used when the user wants to set up a schedule of a device. That Screen might be different for different devices, or it might be the same.

The Screens section in the Mobile Foundry configuration provides the framework with details about each Screen supported in the application.

Mobile Foundry provides a robust set of Screens to handle just about any type of activity the user might want to do. Mobile Foundry also allows for additional Screens to be created by developers and used just like the built-in screens.

The Screen configuration items contain the following fields:

Name	Contents
name	Name for the screen. This name is used elsewhere in the configuration to refer to this screen.
class	The name of the class for the screen, used to create the object dynamically
icon	Optional icon reference used to associate an icon with the screen (e.g. when presented in a list or menu)
title	String reference used when displaying the name of the screen (e.g. when presented in a list or menu)
menu	Menu reference used by the screen. If present, the items from the menu will be populated in the Screen’s system menu
extras	Name / value pairs that are passed to the screen. Each screen may want extra information specific to the screen’s functionality. That is provided in this section.

All screens referenced in the configuration must have an entry in this section.

Screens included with Mobile Foundry/FastTrack

Mobile Foundry ships with several screens that can be used with various devices and applications.

About Screen

This screen provides the “About <Application>” view. The AboutScreen uses the “extras” fields to allow the developer to customize the content and appearance of the screen:

includedItems: A mapping of strings to booleans indicating which items should be displayed in the screen. All items marked as true will be displayed in the about screen:

```
{
        "includedItems": {
                "app_version": true,
                "ayla_app_framework_version": true,
                "ayla_sdk_version": true,
                "sdk_version": true,
                "os_version": true,
                "oem_configuration": true,
                "device_model": true,
                "ayla_service": true,
                "ayla_service_location": true,
                "network_operator": true,
                "language": true
        }
}
```

AccountDetailsScreen

This screen provides edit fields the user can use when creating or editing an account. This screen is used both for the “sign-up” and “edit account” views within the application.

The AccountDetailsScreen uses a “mode” extra to determine whether or not the screen is meant for a new user or to edit an existing account.

The example below shows two different Screen entries using the AccountDetailsScreen. The entry named “edit_account” uses the extras to specify the screen’s mode as “edit”, while the extras field for the “sign_up” version of this screen sets the mode to “signUp” as well as hiding the title bar and disabling the drawer menu while that screen is active:

```
{
        "class": "AccountDetailsScreen",
        "name": "edit_account",
        "title": "edit_account",
        "extras": {
                "mode": "edit"
        }
},
{
        "class": "AccountDetailsScreen",
        "name": "sign_up",
        "title": "sign_up",
        "extras": {
                "mode": "signUp",
                "wantsTitleBar": true,
                "disableDrawerMenu": true
        }
}
```

FastTrack Signup and User profile screen

```
{
    "class": "FTSignUpScreen",
    "name": "sign_up",
    "title": "sign_up",
    "extras": {
        "fields": [ "email", "password"],
        "disableDrawerMenu": true,
        "privacy_policy": "privacy_policy",
        "terms_conditions": "terms_and_conditions"
    }
}
```

FastTrack User profile screen

User profile screen starts from manage account screen which has other user related information like user profile, change password and time zone.

Manage profile section can be configured like below.

```
{
  "class": "FTManageAccount",
  "name": "edit_account",
  "title": "edit_account",
  "icon": "ic_account",
  "menu": "account_menu"
},
{
  "class": "FTUserProfileScreen",
  "name": "ft_user_profile",
  "title": "ManageAccount.userProfile"
},
{
  "class": "FTPasswordScreen",
  "name": "ft_change_password",
  "title": "password",
  "icon": "ic_account"
},
{
  "class": "FTTimeZoneSetting",
  "name": "ft_time_zone",
  "title": "ManageAccount.timezoneSetting",
  "icon": "ic_account"
}
```

DeviceRulesScreen

The DeviceRulesScreen displays the list of rules set up on a particular device, as well as an add button that can be used to add a rule (launches the AddRuleScreen). Each rule displayed in this screen has a switch that allows the rule to be easily enabled or disabled. Long-pressing on a rule prompts the user to delete the rule.

AddRuleScreen

The AddRuleScreen displays a wizard allowing the user to create a new rule for the selected device. The set of available actions for rules are taken from the configuration's managed properties' Action fields.

AllDevicesScreen

The AllDevicesScreen is the default "homeScreen" used in the default Mobile Foundry configuration. This screen displays a list of all devices registered to the user, as well as an "add" button ("+") that launches a screen to help onboard new devices. This screen may be configured by setting the extra, "addDeviceScreen" with the screen name that should be launched when the button is tapped.

When devices are selected (tapped) from this screen, the "device detail" screen for that particular device is launched. Devices may indicate the screen that should be displayed when they are tapped by setting the "detailScreen" field in the device's configuration.

margin param in extras accepts int value to add margin between device's cell, left margin, top margin and right margin. Default Margin is set to 10. If margin is set to 0 than between device cell spacing will be 3px to differentiate devices in device list.

```
{
    "class": "AllDevicesScreen",
    "name": "device_list",
    "icon": "ic_device_list",
    "title": "device_list",
    "extras": {
        "addDeviceScreen": "setup_wizard",
        "includeGateways": true,
        "sortBy": "connectedAt",
        "sortingOrder": "descending",
        "margin": 0
    }
}
```

FT Device Details Screen

FastTrack device details screen will be shown when user click on a device from Device list screen.

FastTrack device details screen has tab bar menu and a property to display. A tab bar can be configured by providing tab bar menu in extras and if tab_menu is not configured then tab bar will not display even with empty list. A property to display is can be configured in extras to display any device property display name and it's value (Currently supported base type are String, Int, Bool, Float & Decimal).

```
{
    "class": "FTDeviceDetails",
    "name": "ft_device_details",
    "title": "ft_device_details",
    "icon": "device_details_icon",
    "extras": {
        "tab_menu": "device_tab_menu",
        "propertyToDisplay": "version"
    }
}
```

AppSettingsScreen

The AppSettingsScreen may be used to provide user-changeable application settings for personal customization. The screen uses the SepiaAppSettings class to manage the settings. The particular settings and their defaults are defined in the “applicationSettings” section of the “userExperience” section.

ChooseConfigScreen

This screen should only be used in demonstrations or while the configuration is still under development. It allows the user to select a different Mobile Foundry configuration file to use with the application. Upon selecting a configuration, the application is re-started using the selected configuration. If the configuration fails to load, the default configuration is instead loaded.

DeviceDetailsScreen

The DeviceDetailsScreen provides basic and generic information about a device. Generally application developers will create a custom screen that allows better control of the device. This screen is launched by default in response to a tap from the AllDevicesScreen for devices that do not have another details screen configured.

DeviceSettingsScreen

The DeviceSettingsScreen can be used to configure settings of a particular device that are managed through its properties. For example, an air conditioner unit might have a display setting for the unit that can be configured in Fahrenheit or Celsius via setting a property to 1 or 0. The DeviceSettingsScreen would be informed of this particular property, and would present a user interface based on the property’s actions allowing the property value to be changed.

The “extras” field of the DeviceSettingsScreen allows for the display of an “unregister” button (to unregister the device) as well as an array of Managed Property names that should be offered as device setting options.

The Actions associated with the Managed Properties are used to determine the valid values for these properties.

In the example below, the “t_eco” property is used to turn on and off “eco mode” on an air conditioner. The “t_eco” property is defined to have two actions, “eco_on” and “eco_off” with the values 1 and 0 respectively. The “t_temptype” is similarly defined to have values for “C” and “F”.

The example below defines a screen called “ac_settings” that allows the user to change the values of t_eco and t_temptype to “on/off” or “C/F”:

```
{
        "class": "DeviceSettingsScreen",
        "name": "ac_settings",
        "title": "ac_settings",
        "extras": {
                "unregisterButton": true,
                "propertySettings": [
                        "t_eco",
                        "t_temptype"
                ]
        }
}
```

DeviceSwipeContainer

The DeviceSwipeContainer is an alternative to the AllDevicesScreen for the root screen. The container provides a left-to-right swipable interface that is populated with the “detail” screens for each device registered to the account. If this screen is used instead of the AllDevicesScreen, the user is presented with the details screen of a single device at a time, with the ability to swipe between devices.

This Screen does not work well with the navigation drawer, as the same swipe gesture is used to swipe “right” as well as open the navigation drawer. While possible to use together, it is recommended that a drawer is not used for this configuration.

EVBDetailsScreen

This is the default screen used to show details for the Ayla EVB. It is an example of how a Screen may be written for a particular device.

HTMLViewScreen

The HTMLViewScreen is used to contain web content. For example, a “help” menu item might be configured to launch a browser with a specific URL

To add a menu item that launches a webview pointing to https://www.aylanetworks.com/privacy-policy, the configuration would include an HTMLViewScreen configured like this:

```
{
        "class": "HTMLViewScreen",
        "name": "privacy_policy",
        "title": "privacy_policy",
        "icon": "privacy_policy_icon",
        "extras": {
                "htmlUrl": "https://www.aylanetworks.com/privacy-
policy",
                "launchInBrowser": true
        }
}
```

and could be launched from the menu simply by adding a menu item named “privacy_policy”.

Configuration options specified in the “extras” field are:

MenuScreen

The MenuScreen presents the contents of a Menu as a full-screen scrolling list. The specific menu to be displayed is indicated with an entry called “menu” in the “extras” field of the screen:

```
{
        "class": "MenuScreen",
        "name": "support",
        "title": "support_screen_title",
        "extras": {
                "menu": "support_screen_menu"
        }
}
```

NoDevicesScreen

This screen is the default screen displayed when no registered devices exist on the account. Rather than displaying an empty list, this screen contains instructions on how to add a new device by tapping the add button.

SetupWizardScreen

This screen takes the user through a step-by-step process to add a new device to their home network and user account. Specific information about the devices such as registration type, SSID regular expression to find the device, etc. is taken from the device configuration section of the Mobile Foundry configuration file.

Some registration flows require additional steps. The messaging of these steps may be provided via string references in the configuration extras for the screen.

SetupWizardScreen extras include:

Name	Contents
prerequesiteText	Instructions for the user that they can confirm with a yes / no response. An example might be, “Is your air conditioner plugged in and powered on?”. This text is displayed to the user at the start of the flow, if present, and yes / no buttons are provided to receive the response.
prerequisiteHelp	Further instructions displayed to the user if the answer to the “prerequisiteText” question was “no”. An example might be, “Plug in and power on your air conditioner by toggling the switch near the back of the unit”.
mode	If set to “register”, the screen will skip the device discovery (WiFi setup) steps and only attempt to register the device rather than set up its network.
advanced_options	Show or hide advance options from top right corner share icon option, default this option is enabled.
hideDevDevices	Hide or show devices tagged as development devices. Devices marked as **developmentDevice** can be hidden on supporting screens.

SignInScreen

The first screen visible to a clean install, the SignInScreen provides an interface that allows the user to sign in to the Ayla service via phone number, email, Google, Facebook or WeChat. It also contains a button to launch the AccountDetailsScreen to create a new account, and links to re-send account confirmation as well as a “forgot password” link.

This screen is not presented if the user has already signed in and can refresh authorization.

FTNotifications Screen

`FTNotifications` screen is a Menu Screen used to display FastTrack Notification Settings. Notification Settings is configured as a drawer menu option where the user can optionally push notifications. Developers can configure the following types of notifications settings in the menu.

- Device On
- Device Off
- Device Online **COMING SOON**
- Device Offline **COMING SOON**

Every notification display title i.e. Notification name along with info on how many device Notifications are enabled in the subtitle. For example ‘All’ means all devices are enabled to receive push notification(s) by default. 'Disabled' means none of devices opted to receive notifications. Otherwise a number will display which reflects the number of devices optioned for push notifications. On a notification tap, the user is taken to the Notification Detail screen.

```
{
    "class": "FTNotifications",
    "name": "ft_notifications",
    "title": "notifications_title",
    "icon": "notification_icon",
    "menu": "notification_menu"
}
```

FTNotificationDetail Screen

The Notification Detail Screen is displayed when a user taps on one of the notification types to configure the push notification settings on a per device basis. Here, the user can select devices to receive push notification(s) based on the type of notification. Developers can configure the push notification title and message using the Notification Detail screen configuration in extras. As shown below, this also works well with localization strings.

Ex. Device on Notification detail screen config

```
{
      "class": "FTNotificationsDetails",
      "name": "ft_notification_device_on",
      "title": "notifications_device_on",
      "extras": {
          "type": "on",
          "push_title": "notification_push_turn_on_title",
          "push_body": "notifications_push_turn_on_message"
      }
  }
```

Controls

In addition to Screens, Mobile Foundry allows Controls to be associated with devices or properties (see the section about Managed Properties and Actions for more details about how these are associated with each other).

Name	Contents
name	Name for the control. This name is used elsewhere in the configuration to refer to this control.
class	Class for the control (Java, Obj-C class name)
extras	Name / value pairs passed to the control to provide additional information. The values here depend on the control.

ActionPicker

The ActionPicker control is backed by a ManagedProperty that has Actions set. The default display of the control is an icon and text field with the selected action. When tapped, the ActionPicker presents the set of Actions for that ManagedProperty to the user and updates the property value when the action is selected.

ActionPicker extras include:

Name	Contents
property	Name of the property this control should represent
menuBackgroundColor	Background color for the pop-up menu
menuTextColor	Text color for the pop-up menu.

AylaConnectivityIndicator

This control displays the connection status of a device (text) and an icon representation.

Name	Contents
onImage	Image shown for the “on” state
offImage	Image shown for the “off” state

readOnly	If true, will not allow modifications to the value

OnOffControl

This control is used to toggle a Boolean property on and off.

Colors

Colors used throughout Sepia are defined in this section. Colors are simply name / value pairs, where the value is a string using HTML color notation (“#rrggbb”). Other sections of the configuration file refer to the colors from this section by name.

```
"colors": [{
                        "name": "ayla_green",
                        "value": "#87cc4b"
                },
                {
                        "name": "primaryColor",
                        "value": "#87cc4b"
                },
                {
                        "name": "accentColor",
                        "value": "#87cc4b"
                },
                {
                        "name": "textColor",
                        "value": "#000000"
                },
                {
                        "name": "list_bg",
                        "value": "#aaffaa"
                },
                {
                        "name": "window_bg",
                        "value": "#ffffff"
                },
                {
                        "name": "evb_button",
                        "value": "#2266cc"
                },
                {
                        "name": "blue_led",
                        "value": "#0000cc"
                },
                {
                        "name": "green_led",
                        "value": "#00cc00"
                },
                {
                        "name": "autoDeviceBG",
                        "value": "#bbbbbb"
```

```
                }
        ]
```

Text Styles

Text Styles or fonts used throughout Sepia are defined in this section. Fonts are Dictionary of name of the font & styleParams. Where font name is name of font which can be configurable and assign to any text in Themes or in default configurations section, styleParams contains font, font style and appearanceStyle. Font is custom font name like Roboto, QuickSand…etc. Font style is of Bold, Regular, Light …etc and appearanceStyle are standard style provided by an iOS or and an Android platform. Below example's shows how to configure fonts in config file under user experience section.

```
{
        "textStyles": [{
                        "name": "screenTitleStyle",
                        "styleParams": {
                                "font": "Roboto",
                                "style": "Bold",
                                "appearanceStyle": "title1"
                        }
                },
                {
                        "name": "screenTitleLight",
                        "styleParams": {
                                "font": "Roboto",
                                "style": "Light",
                                "appearanceStyle": "title2"
                        }
                },
                {
                        "name": "bodyTextLight",
                        "styleParams": {
                                "font": "Roboto",
                                "style": "Light",
                                "appearanceStyle": "body"
                        }
                },
                {
                        "name": "bodyTextBold",
                        "styleParams": {
                                "font": "Roboto",
                                "style": "Bold",
                                "appearanceStyle": "body"
                        }
                }
        ]
}
```

Themes

Themes are the most exiting feature which users expects when using mobile applications with the introduction of iOS 13 light and dark theme. Using Sepia themes developer can now support multi-color mode by configuring colors & fonts in config.json file. With the introduction of Theme, we will not using defaultConfiguration keys and same keys have been moved to themes. Assume in many lower end devices in android or lesser than iOS 13 versions user can't set default color mode like dark/light. In such cases according to customer requirement developer can configure default theme by making defaultTheme true. With the introduction of Fonts/textStyle's and colors theme can play major role in configuring theme based application for Sepia UI elements. Below json structure show how to configure theme in user experience section.

Name	Contents
name	Name of the theme
defaultTheme	The default theme to load: light or dark. If omitted, the theme will be loaded based on the configuration in the OS. If the OS does not specify / support themes, the light theme will be used.
windowBackgroundColor	Color reference for the default window background color
textColor	Color reference to the default text color
toolbarColor	Color reference to the color used for the toolbar (top bar)
toolbarTextColor	Color reference to the text color for the toolbar (top bar)
logoImage	Image reference to the brand/Logo to load on Sign in screen(Account screen in case of Foundry)
brandColor	Color reference to the Customer brand color
buttonBackgroundColor	Color reference to the Buttons Background color
buttonTextColor	Color reference to the Buttons Text Color
segmentedControlColor	Color reference to the Segment control tint color
listBackgroundColor	Color reference for the background color of lists. Defaults to the background window color
listTextColor	Color reference for the title text color of Lists. Default color to the textColor
listTitleFont	Font reference for the title text font of the Lists.
listSubtitleColor	Color reference for the subtitle text color of Lists. Default color to the textColor
listSubtitleFont	Font reference for the subtitle text font of the Lists.
listMenuBackgroundColor	Color reference for the List if it acts as Menu
windowTitleFont	Font reference used for the toolbar (top bar)
windowBackButtonIcon	Icon reference used in screen navigation back button
rectangularButtonTitleFont	Font reference to the Buttons title text
textButtonTitleFont	Font reference to the text Button title text (Used in case of FastTrack)
textButtonOnClickOpacity	Opacity reference to the text Button on click action or highlight color of text button
regularTextFont	Font reference to the text used across the app
textFieldFont	Font reference to the input fields
textFieldTextColor	Color reference for the input fields
validateErrorColor	Color reference for the error to represent in input fields or Labels
regularTextFontCallout	Font reference smaller than **regularTextFont**
regularTextFontBold	Font reference with bold against **regularTextFont**
accountVerificationBoldFont	Font reference used in account confirmation screen (Used in FastTrack)

popOverBackgroundColor	Color reference used in Pop overs background color
regularTextLargeFont	Font reference bigger than **regularTextFont**
drawerScreenNameFont	Font reference used in drawer menu Logged in user name display
drawerScreenListFont	Font reference used in drawer menu list
deleteButtonBackgroundColor	Color reference for Delete button background color say for Delete device/Delete Account...etc (used in fastTrack)
deleteButtonTextColor	Color reference for Delete button text color color
menuHeaderBackgroundColor	Color reference for drawer menu header view background color
disabledColor	Color reference to display disabled input fields background color


```
"themes":[
    {
    "name": "light",
    "defaultTheme": true,
    "windowBackgroundColor": "white",
    "textColor": "black",
    "toolbarColor": "white",
    "toolbarTextColor": "black",
    "logoImage": "ayla_logo",
    "brandColor": "brandColor",
    "buttonBackgroundColor": "brandColor",
    "buttonTextColor": "white",
    "segmentedControlColor": "brandColor",
    "listBackgroundColor": "white",
    "listTextColor": "black",
    "listTitleFont": "regualrTextStyle",
    "listSubtitleColor": "textFieldColor",
    "listSubtitleFont": "regulartextStyleFootnote",
    "listMenuBackgroundColor": "deviceListBgColor",
    "windowTitleFont": "titleFont",
    "windowBackButtonIcon": "backArrow",
    "rectangularButtonTitleFont": "RectangularButtonTitleStyle",
    "textButtonTitleFont": "TextButtonTitleStyle",
    "textButtonOnClickOpacity": 0.3,
    "regularTextFont": "regualrTextStyle",
    "textFieldFont": "TextFieldTitleStyle",
    "textFieldTextColor": "black",
    "validateErrorColor": "red",
    "regularTextFontCallout": "regulartextStyleFootnote",
    "regularTextFontBold": "regualrTextBoldStyle",
    "accountVerificationBoldFont": "regualrTextBodyBoldStyle",
    "popOverBackgroundColor": "deviceListBgColor",
    "regularTextLargeFont": "regulartextStyleTitle2",
    "regularTextLargeBoldFont": "regulartextStyleTitle2Bold",
    "drawerScreenNameFont": "drawerScreenNameFont",
    "drawerScreenListFont": "drawerScreenListFont",
    "deleteButtonBackgroundColor": "deviceListBgColor",
```

```
    "deleteButtonTextColor": "red",
    "drawerHeaderBackgroundColor": "deviceListBgColor",
    "disabledColor": "deviceListBgColor"
    },
    {
    "name": "dark",
    "windowBackgroundColor": "black",
    "textColor": "white",
    "toolbarColor": "black",
    "toolbarTextColor": "white",
    "logoImage": "ayla_logo",
    "brandColor": "brandColor",
    "buttonBackgroundColor": "brandColor",
    "buttonTextColor": "white",
    "segmentedControlColor": "brandColor",
    "listBackgroundColor": "black",
    "listTextColor": "white",
    "listTitleFont": "regualrTextStyle",
    "listSubtitleColor": "textFieldColor",
    "listSubtitleFont": "regulartextStyleFootnote",
    "listMenuBackgroundColor": "black",
    "windowTitleFont": "titleFont",
    "windowBackButtonIcon": "backArrow",
    "rectangularButtonTitleFont": "RectangularButtonTitleStyle",
    "textButtonTitleFont": "TextButtonTitleStyle",
    "textButtonOnClickOpacity": 0.3,
    "regularTextFont": "regualrTextStyle",
    "textFieldFont": "TextFieldTitleStyle",
    "textFieldTextColor": "white",
    "validateErrorColor": "red",
    "regularTextFontCallout": "regulartextStyleFootnote",
    "regularTextFontBold": "regualrTextBoldStyle",
    "accountVerificationBoldFont": "regualrTextBodyBoldStyle",
    "popOverBackgroundColor": "black",
    "regularTextLargeFont": "regulartextStyleTitle2",
    "regularTextLargeBoldFont": "regulartextStyleTitle2Bold",
    "drawerScreenNameFont": "drawerScreenNameFont",
    "drawerScreenListFont": "drawerScreenListFont",
    "deleteButtonBackgroundColor": "deviceListBgColor",
    "deleteButtonTextColor": "red",
    "drawerHeaderBackgroundColor": "black",
    "disabledColor": "clear"
    }
]
```

Templates

Templates are used to format email and SMS messages sent to the user. These messages are used to confirm email addresses or phone numbers, or simply to provide information to the user. Templates are defined in the Ayla cloud, and are given a unique identifier. The Templates section in the Mobile Foundry configuration allows the developer to link template IDs with the application.

Name	Contents
name	Name of the template configuration, user-defined
template_id	String resource identifier for the template ID. As templates are direct communications to the end user, a different template for each supported language is required.
subject	String resource identifier for the email subject
html	String resource identifier for the HTML body of the message (optional, usually unused as the body is generally taken from the template itself)

```
"templates": {
        "on_connection_lost": "on_connection_lost_template",
        "on_connection_restore": "on_connection_restore_template"
}
```

```
// strings.xml contains different template references for English and
Spanish:
strings.xml
<string name=”on_connection_lost”>ayla_on_connection_lost_template_01<
/string>

es/strings.xml
<string name=”on_connection_lost”
>ayla_on_connection_lost_template_01_es</string>
```

Application Settings

Application settings may be defined in the configuration, and are used by the AppSettingsScreen to allow the end-user to override defaults. An example for an HVAC application might have an option to choose between Celsius and Fahrenheit for temperature display.

Name	Contents
name	String reference to the name of the setting
type	Boolean|String|Integer|Decimal|MultipleChoice
defaultValue	Default value for the setting
choices	Array of choices (in the case of MultipleChoice)

```
"applicationSettings": [{
        "name": "temperatureUnits",
        "type": "MultipleChoice",
        "choices": ["F", "C"],
```

```
        "defaultValue": "C"
}]
```

Default Configuration

The defaultConfig section is a set of name / value pairs used to define the default configuration for user interface elements. Any screen or control may override these defaults in their specific configuration sections.

The following names are currently defined in SepiaScreen:

Name	Contents
toolbarColor	Color reference to the color used for the toolbar (top bar)
toolbarTextColor	Color reference to the text color for the toolbar (top bar)
wantsTitleBar	If true, will show the toolbar, otherwise will not show a toolbar
disableDrawerMenu	If true, will not use a drawer menu for navigation
windowBackgroundColor	Color reference for the default window background color
listBackgroundColor	Color reference for the background color of lists. Defaults to the background window color
primaryColor	Color reference to the primary color for the app. used for the title bar as well as other windows
accentColor	Color reference used for highlights, etc.
textColor	Color reference to the default text color
backgroundImage	Image reference to an image to be used as the window background
windowTitleFont	Font reference to all screen title font
windowBackButtonIcon	Back button icon for all window back button
productNameNumOfCharLimit	Char limit for Product name to display in device list screen and rename device screen. (Ex. 32)
sideMenuWidthInPercentage	Percentage width for drawer menu when open state (ex. 85)
enableSwipeToDeleteDevice	Swipe to delete device/ unregister device in device list screen (true /false)
enableDrawerMenuSwipeGesture	Enable/Disable Swipe gesture to open/close drawer menu
hideDevDevices	Show/hide development device when releasing app to App Store. Configure if device is 'developmentDevice' in device config
allowRootedDevice	Default value is considered as **false** and will check for rooted device and alert the user. Set this value **true** if needed to bypass checking for rooted device (true/false). Android refers it as root access where as in IOS it is referred as Jailbreaking.

Extending and Customizing

Mobile Foundry includes many features common to many different types of IoT applications and devices. However, some devices or applications may have needs that are not met by the provided Mobile Foundry screens or controls. Fortunately, Mobile Foundry was designed to allow developers to extend or replace any of the existing screens or controls with their own.

Additional classes must inherit from either SepiaScreen (for Screen classes) or SepiaControl (device or property controls). These classes, once written, may be used and referenced from the main configuration file just like built-in Screens or Controls.

So that the framework can find and instantiate any custom classes, make sure the “packages” and “bundles” arrays in the root of the configuration include all locations where custom classes might be found.