Red Hat Mobile Application Platform 4.3
Product Features

For Red Hat Mobile Application Platform 4.3

Red Hat Customer Content Services

Red Hat Mobile Application Platform 4.3 Product Features

For Red Hat Mobile Application Platform 4.3

Legal Notice

Abstract

This document provides a comprehensive description and usage instructions for the features of Red Hat Mobile Application Platform 4.3.

Table of Contents

CHAPTER 1. PROJECTS, APPS, AND SERVICES

1.1. PROJECTS

1.1.1. Overview

The projects section allows you to create and manage projects. Once a project has been created, multiple Apps can then be associated with those projects. If you have an existing App, the 'Import' feature can be used to import it into the Platform.

Requirements

The user must be a member of one or more teams with the following permissions:

- Domain — Project (View and Edit)

1.1.2. New Project

A Project consists of Apps for different platforms, and a Cloud App that they talk to.

Note

Although a new project generally consists of one Client App and a Cloud App for it to communicate with, multiple Apps can be added to any given project.

1.1.2.1. Client Platforms

All Projects have a template as an initial starting point. The following example describes the makeup of a template:

- A single App that uses the technology Cordova and a Cloud App that uses the technology Node.js.

1.1.3. Import

As well as creating Apps, existing Apps can be imported into the Platform. When importing an App, you have the choice to either create a new project and add the App to it, or to simply add the App to an existing project. There are a number of different App types that can be imported into the Platform. These include Native iOS, Native Android, PhoneGap, Basic Web App, and Advanced Web App. There are also a number of different ways in which an App can be imported to the Platform. These include :

- Cloning an existing Git repo
- Uploading a Zip
- Creating an App with a blank repo, and then pushing the code there
- Not importing source, but integrating RHMAP SDK

Step by step instructions will be provided, informing users of how to successfully import their Apps into the Platform. These steps will differ slightly depending on which import method was selected. For example, only the 'Cloning an existing Git repo' option has instructions on how to clone Git repositories.

1.1.4. Apps, Cloud Apps & Services

The Apps, Cloud Apps & Services section allows for the managing of Client Apps and Cloud Apps within a project. Here, new Client and Cloud Apps can be added to a project. Cloud Apps serve as a gateway between the Client App and the cloud back-end storage and services.

1.1.4.1. Apps

A project can consist of multiple Apps. In general, a project will consist of at least one Client App, along with a Cloud App for it to communicate with. Multiple Apps and Cloud Apps can be added to a project. When adding a Client App, Apps can be created based on a number of Native or Hybrid templates. Multiple Cloud Apps can also be added to a project. As mentioned above, an existing App can also be imported into an existing project.

1.1.4.2. Cloud Code Apps

Cloud Apps allow developers to link their Apps to back-end cloud storage. It allows for extensive user management, where users can manage mBaaS Services such as online data storage, web based email services, push notifications, and more. Client Apps will communicate with Cloud Apps to avail of this functionality. For example, since all functionality created in your cloud can be accessed via a simple API call, you can make use of a cloud cache for performance so less work needs to be done by the device itself.

1.1.4.3. Services

External mBaaS Services can be added to a project in order to provide increased functionality. Services can be added to a project in the same way as a Client App or mBaaS Cloud Instance. Cloud Instances make calls to these services to avail of increased functionality such as PayPal. Only a user with the Service Administrator role can provision mBaaS services. When a Service Administrator is creating a service, they have the option to make the service globally available to all projects. If it is made globally available, users with the Developer role can select it for inclusion in their project.

1.1.5. Add Apps to a Project

You can add as many Apps to a Project as you like - you have three options to choose from:

1.1.5.1. Create a New App

Creating a new App allows you to select template to base your new App on - alternatively, if you have an existing App, you may want to select the "Import Existing App" option

1.1.5.2. Import Existing App

If you have an existing App, you can select this option and we'll guide you through importing and integrating your existing Apps with the Platform. You'll need to select an appropriate App Type and make sure the pre-requisites are met - the App Studio will help guide you through the process.

1.1.5.3. Clone Existing App

If there's an App in this (or another Project) that you'd like to clone, select this option. You'll be asked to give your cloned App a new name. Depending on the source App, you may also be presented with some additional choices in terms of cloning depending on the source App.

Some Apps can have their types changed while being cloned, and other Apps can share their Git repositories with existing Apps. We recommend that Apps have their own dedicated Git repository, but there are some instances where it's desirable to share Git repositories between Apps (for example, where a Basic Web App and a Cordova App share extremely similar source) - in these cases, Apps can share repositories and choose to elect folders within a shared repo to be used for Building, Previewing and Deploying.

1.1.6. Connections

The connections section provides an overview of Binaries built in the selected project, along with running Cloud code. A project contains both Client Apps, and Cloud Apps. A Connection dictates which Cloud Instance a Client App points towards. A connection will point towards a particular Cloud App in a given environment. The Dashboard provides an overview of connections that are present in all environments. From the dashboard, users can see how each Client App is connected to a Cloud Instance. Connections between Client Apps and Cloud Apps can be reconfigured or disabled as often as you like.

1.1.6.1. SDK Config

When a connection between a Client App and a Cloud Instance is created, an SDK configuration file is generated. Selecting the **`SDK Config`** option displays this config file. The contents of this file can be copied and pasted into your Client Apps **`fhconfig.json`** file in order to instruct the Client App to use that particular Connection.

1.1.6.2. Reconfiguring a connection

Connections can be reconfigured to allow a Client App to communicate with a different Cloud Instance. Cloud connections are refreshed by Apps upon startup.

1.1.6.3. Disabling a Connection

This removes the connection between a Client App and a Cloud Instance. This may be necessary if a new App has been added to a project and an update is required. Disabling a connection can cause a Client App to stop functioning. It is possible to send a message to clients informing them of a connection being disabled.

1.1.7. Git QuickStart

The Git QuickStart page shows how to access Client and Cloud code of an App on your local system by using Git. To access your project from your computer, you must first clone the repository. Cloning a repository creates an exact copy of an existing repository. All versions of all files in project are pulled down when a 'git clone' is performed. However before you can clone a repository, you need to upload your SSH Public Key. Once a Public Key has been uploaded, repositories can be cloned at will.

For more information regarding cloning a Git repository, see the Git Documentation on Getting started with Git.

For a more in-depth list of useful Git commands, see Git Reference.

1.1.8. Resources

The Resources section shows resource usage, such as CPU usage, Memory usage & Disk usage, for all Apps in the Project across all environments. If an App is not deployed in a particular environment, no resource usage will be shown for it. The status of deployed Apps can be managed from this screen as well. For example, if a particular App is showing high memory usage, it can be stopped. Resource graphs are based on live data, and as such may take a while to show trends while data is fetched periodically.

1.1.9. Lifecycle Management

A Project typically starts as a single App (for example, a Cordova App) and a Cloud App (for example, Node.js Application). As a Project progresses, it goes through various stages, that is, a Project Lifecycle. Every Project moves at a different pace, and grows & varies in composition over time. The goal of the Lifecycle Management Dashboard is to make this process easier for both the developer(s) and the project manager.

The Dashboard provides a snapshot view of a Project at the current point in time. All Apps are represented by a row on the dashboard. Each Environment corresponds to a column on the dashboard. This grid can give a quick overview of the stage that each part of the Project is in, without having to consult with each team member.

If an App has been built, the most recent build will be shown in the column that represents the Environment it is targeting a Cloud App in. The App binary can be downloaded from here, or rebuilt from a different git branch or tag. The 'Promote' option is a shortcut to rebuild the same code, but connect it to a different Cloud App Environment.

If a Cloud App has been deployed, the most recent deploy details and status will be shown in the corresponding Environment column. The Cloud App code can be deployed (or redeployed) to any available Environment. The 'Promote' option for Cloud Apps is a shortcut to deploy an already deployed git branch or tag to a different Environment.

Note

Any App binary builds done outside the platform will not be reflected on the Lifecycle Management Dashboard.

1.1.10. Forms

The Forms configuration page allows users to both associate Forms and Themes with a Project, and also configure Client Side AppForms settings. Users can manage settings relating to Camera Settings, Submissions, Admin Users, Client Logging, and Cloud Logging.

Submissions associated with a project can also be viewed from this page.

Note

If a user does not have sufficient privileges, they will not be able to view submissions. For more information regarding user roles, see the Administration Documentation.

1.1.11. Reporting

Note

The Reporting data is generated once per day.

The reporting section offers high level graphical representation of analytical information for both Client and Cloud Apps in a given project. Users can access a variety of information, such as how many Device Installs were performed, how many App Startups occurred, and more. Any one of these areas can be further investigated to access analytical information based on Date, Platform, and Location.

In Studio, this information is represented via a number of different graphs. Date specific data is represented by a line graph. This allows users to see a breakdown of App activity for a specific date range. Users can see the volume of activity for any given date in the range. For example, users could see how many Cloud Requests were performed on any given date in a date range.

The donut graph shows a breakdown of activity per platform. For example, the number of installs performed per platform inside a specified date range. Location specific data is represented on a Geographical map. This provides users with a breakdown of App activity based on geographical location. Each graph has a corresponding legend. Hovering the cursor over any of these charts will provide more info on the corresponding value in the legend.

The date range from which the data is gathered can be altered. Predefined ranges display information based on the 'Last 7 Days', 'Last 30 Days', and 'Previous Month', however if none of these are applicable, it is also possible to manually select a start and end date for the Range. This enables users to retrieve App data for a specific date range by simply selecting a data category, and then entering a date range.

1.1.12. Settings

From the Project Settings page, projects can be renamed or deleted.

1.1.12.1. Deleting a project

Once a project is deleted, it can not be undone. All Git repositories, Client Apps, and Cloud Apps will be removed. Any Apps that are associated with the project will no longer function.

1.1.12.2. Renaming a project

Here a project can be renamed.

1.2. APPS

1.2.1. Overview

There are a wide variety of Apps that can be built via the Red Hat Mobile Application Platform (RHMAP), including hybrid, HTML5, or Native Apps. Both Client and Cloud Apps can be added to any project. Apps have access to a number of JavaScript API’s that offer access to features such as geolocation or back-end storage. Form based Apps can now be created via a simple drag and drop interface.

Requirements

The user must be a member of one or more teams with the following permissions:

- Domain — Project (View)
- Client App (View & Edit) — Analytics (View)

Related Resources

- FHC CLI

1.2.2. Details

The Details page is opened by default when the **Apps, Cloud Apps and Services** screen is selected. Here users can access overview information regarding the currently selected App. This information includes:

- App Type: The type of your App.
- App Name: The name of your App. For Client Apps, this is also the name of the App as it will appear on a device.
- App Description: A description of your App.
- App ID: Unique identifier for the App. This can be used to perform actions on this App with FHC.
- App Description: Brief description of the App.
- App API Key: This is used in conjunction with App ID to allow the App to communicate with a chosen Cloud App.
- Source Path (Cordova only): The folder containing all source files of the app.

  Typically this is `www`. If you want to use a different folder when building or deploying your App, it can be changed here. Preceding and trailing slashes are stripped from paths. For example, if your source folder is `/mysource/folder/`, the path you enter here should be `mysource/folder`. Any build processes you have for your App (Grunt minification etc.) should place finalised assets in this folder.
- Git SSH URL: URL to clone repository. If this repository is shared with other Apps, these Apps will be listed here.
- Git Branch: The branch to checkout from the Git repository.

Any of the App details can be updated from this page. In Studio, there is also an App Preview on this screen. Users can choose to preview how the App would appear on either a phone, tablet, or desktop. After one of the devices has been selected in the App preview window, the user can preview how the App would appear on specific device by selecting a specific device from a list of dropdown options.

Apps can also be deleted from the **Apps, Cloud Apps & Services** page.

Note

When an App is deleted, its associated Git Repository will be removed and if it's a Cloud App, it will be undeployed.

1.2.3. Editor

The built in editor allows users to edit code within their Apps. The built in syntax highlighting feature makes it easier to distinguish between syntactical elements. Users can create new files and folders, with saved changes being automatically pushed back to the corresponding Git repository.

After a change has been made, the preview automatically refreshes so you can instantly see your changes take effect. Changes can also be pulled in from a Git Repository by performing a 'git pull'.

The App Preview in the Editor section allows users to view how the current App appears on a number of selectable devices.

1.2.4. Docs

If your App contains a **`README.md`** markdown file in its route, it will be rendered and shown here.

1.2.5. Analytics

Note

The Analytics data is generated once per day.

The Analytics section provides access to a graphical representation of App usage statistics. Data can be examined via a number of categories including Installs, Cloud Requests, StartUps, and Active Devices. This data can be further categorized by date, platform, and location. Graph content is displayed by platform, with the exception of the 'Installs by Location'. Each platform is represented by a different key in the legend. Content in the graphs can be toggled on/off by clicking on the respective platform in the legend.

The date range can be also be customized. Predefined ranges include 'Last 7 days', 'Last 30 days', and 'Previous Month'. Users can also apply their own start and end date to the range.

Note

The Analytics page is not to be confused with the 'Reporting' tab. The Reporting tab shows analytical information on a per project basis. The 'Analytics' page shows displays data on a per App basis.

1.2.6. Push

The Push section allows users to configure push notifications for an App. Learn more about using push notifications in the Push Notifications chapter.

1.2.7. Building

The Build feature enables users to build a set of binaries for a selected platform. When performing a build operation, the Cloud App you want the Binary to connect to must be selected. This can be any of the Cloud Apps associated with the project. Builds can be done for any of the following platforms:

- Android

- iPhone
- iPad
- iOS Universal
- Windows Universal Apps

One of a number of Build Types must also be selected. These include:

- Development Build
- Distribution Build
- Release Build
- Debug Build

Depending on the platform that has been selected, along with the Build Type, a Credential Bundle may be needed to sign the Binary. For more information on Credentials, see the Credentials section below.

Once a build has been successful, its Artifact is added to the Artifact History. The Artifact History provides summary information of the build, including the Platform the binary was built for, the Build Type, and the Credential Bundle used for the build. Selecting an Artifact from the Artifact History gives the option to download the binary again.

Note

In order to be able to perform some build operations, you must have first uploaded the relevant credentials via the Credentials page.

1.2.7.1. Publishing App Binaries to Third-party MAM and MDM providers

RHMAP provides support for uploading app binaries to third-party MAM and MDM providers. You can enable or disable support for individual providers in the *Admin* > *Mobile App Management* > *Third-party MAM/MDM* section.

If at least one MAM or MDM provider is enabled, the *Build* screen of Client Apps shows an *MDM integration* section with options for individual providers.

1.2.7.1.1. Apperian

To have the generated binary uploaded to the Apperian App Catalog after building, follow these steps before the build.

1. Ensure the domain name of your RHMAP instance is added to a whitelist in Apperian's EASE Portal. See Enable a Domain Whitelist for Custom Admin Sites in official Apperian documentation for detailed steps.
2. Ensure that the app binary meets the requirements imposed by the Apperian App Catalog. See Add an Application in the official Apperian documentation for the full list of requirements.

Note

Some of the RHMAP Client App Templates do not fulfill all of the requirements. For example, you might need to configure an icon file for the app in some cases.

3. Enable publishing to Apperian for a particular build.

    a. Select *Push the generated app binary to the Apperian App Catalog*.

    b. Provide your Apperian username and password.

    c. Click *Apperian Login*. After a successful authentication, the *OAuth Token* field will show a token that will be used for communication with the Apperian API.

The app binary resulting from a subsequent build will be uploaded to the Apperian App Catalog.

1.2.8. Exporting

The Export section allows the user to export an App via a number of selected platforms. When exporting an App, the Cloud App you want the Binary to connect to, the Platform, and the Version of the Platform must all be specified. Apps are exported as a zip file. When an export is complete, the export is added to the Artifcat History where it cab be redownloaded at any time.

1.2.9. Credentials

The Credentials section enables users to manage Credential Bundles. New Credential Bundles can be added or imported. A list of existing Credential Bundles can also be viewed.

1.2.9.1. Credential Bundles

In order to be able to perform certain types of build, an App must be signed with a Credentials Bundle. A Credential bundle is a combination of resources, such as certificates, provisioning profiles, and private keys, necessary for performing specific types of builds, be it a development build, distribution build, debug build etc. Depending on both the platform, and the build Type, a combination of different resources will constitute a bundle. For more information regarding the individual resources that make up a Credentials Bundle, see the Components of a Credentials Bundle page.

Note

To be able to develop Apps for iOS, you must have an Apple Developer Account.

The required credentials for each build type for specific platforms can be seen below:

Note

Ensure that the Private Key and its password are stored in a secure location as the Private Key and password are required when configuring a Credential Bundle on RHMAP. When generating a Certificate, ensure that the correct Private Key is used, for example: when working with an iOS Credentials Bundle, use the Development Build's Private Key to generate the Development Build's Developer Certificate.

iOS Credentials Bundle

- Development Build
  - Developer Certificate
  - Private Key
  - Development Provisioning Profile
- Distribution Build
  - Distribution Certificate
  - Private Key
  - Distribution Provisioning Profile
- Release Build
  - Distribution Certificate
  - Private Key
  - Distribution Provisioning Profile

Android Credentials Bundle

- Debug Build
  - None
- Release Build
  - Certificate
  - Private Key

Windows Phone Credentials Bundle

- Debug Build
  - None
- Release Build
  - None

1.2.9.1.1. Create New Bundle

Here a new Credentials Bundle can be created and later used to sign binaries during a build.

Depending on the platform choosen, different credentials may need to be added to the bundle. For example, when creating an Android Credentials Bundle, a Private Key, and Certificate are required, whereas when creating an iOS Credentials Bundle, a Private Key, Certificate, and Provisioning Profile are required.

1.2.9.1.2. Migrate Existing Dev Resources

This migrates existing credentials from the old App Studio, and adds them to the Credentials Bundles list.

Once a new bundle is created, or imported via migration, it will appear in the Credentials Bundle List.

1.2.10. Integrate

The Integrate section allows users to import an existing Client App into a RHMAP project. In order to import an app, you must clone the repo and integrate the app with the RHMAP SDK.

1. Clone a Git Repo

   Cloning a repository creates an exact copy of an existing repository. All versions of all files in project are pulled down when a 'git clone' is performed. However before you can clone a repository, you need to upload your SSH Public Key. Once a Public Key has been uploaded, repositories can be cloned at will.

   For more information regarding cloning a Git repository, see the Git Documentation on Getting started with Git.

   For a more in-depth list of useful Git commands, see Git Reference.

2. Download SDK

   In order to get your app to function, you must integrate it with the RHMAP SDK. An on-screen link is provided to download the most up to date RHMAP JavaScript SDK.

3. Copy the JavaScript file to your Project

   After the RHMAP SDK has been downloaded, it can be referenced by moving it into the Project you created.

4. Copy Initialisation Snippet

   You must create an **`fhconfig.json`** file that contains the id of the app, the app key, the host domain, and the id of the project to be associated with.

5. Git Commit and Push

   After the SDK has been successfully integrated with the app, the changes can be committed and pushed. The app will now be successfully associated with the project.

1.3. CLOUD APPS

1.3.1. Overview

A *Cloud App* is a server-side application that runs on an *MBaaS* - our cloud execution environment. Client Apps communicate with Cloud Apps in order to access back-end functionality such as data

storage, authentication, caching etc.

Requirements

The user must be a member of one or more teams with the following permissions:

- Domain — Project (View)
- Cloud App (View & Edit) — Analytics (View)

Related Resources

- Cloud API Reference - List of Cloud API calls
- FHC CLI
- Platform API - Cloud Alerts API
- Platform API - Cloud Events API

1.3.2. Details

The Details section provides access to overview information regarding the Cloud App. This section allows users to edit and manage this data.

An App can be in one of two states, either Running or Stopped. The state of the App can be controlled by the *Start App*, *Stop App*, and *Restart App* buttons. If an App is Running, on device builds of that app will function as expected. However if the app is Stopped, on-device users will not be able to use the app if it relies on Cloud functionality.

There is also a list of general information such as the App ID, App API Key, name, etc. This information can be edited and updated from this section.

- App Id: Unique identifier for the App.
- Name: The name of the Cloud Instance
- Description: Brief description of the Cloud Instance
- App API Key: This is used in conjunction with the App Id to allow the App to communicate with the Cloud
- Git URL: URL to clone repository
- Git Branch: The branch to checkout from the Git repository

1.3.3. Editor

The built in Editor can be used to edit code within the Cloud App. The built in syntax highlighting feature makes it easier to distinguish between syntactical elements. Users can create new files and folders, with saved changes being automatically pushed back to the corresponding Git repository.

After a change has been made, the preview automatically refreshes so you can instantly see your changes take effect. Changes can also be pulled in from a Git Repository by performing a **`git pull`**.

The App Preview in the Editor section allows users to view how the current App appears on a number of selectable devices.

1.3.4. Environment Variables

You can define environment variables using the Studio and those variables can be accessed in the cloud code. The Studio will show a list of current defined environment variables and current deployed environment variables for each cloud environment. You can also view the list of environment variables defined by the system.

1.3.4.1. Environment Variables List

You can find the environment variables option in the left hand menu. The initial view shows a list of all environment variables for this Cloud Instance.

1.3.4.1.1. App Environment Variables

This section lists the variables defined by you. These variables may not have been pushed to the cloud yet so the values here may be different from the values that are currently active. Any inconsistency between the defined values and deployed values will be highlighted in red. You can click on the value field to update it.

1.3.4.1.2. Deployed User Environment Variables

This section lists the current deployed environment variables created by you. Again, any inconsistencies will be highlighted. However, those values can not be updated directly.

1.3.4.1.3. Deployed System Environment Variables

This section lists the current deployed environment variables created by the platform. Those values are for your reference only and you should not change the values of those variables.

1.3.4.2. Create Environment Variables

You can create a new environment variable by clicking on the **Add Variable** button. The name of the environment variable should follow these rules:

- Should NOT begin with **`FH`**_ or **`MONIT`**_
- Should NOT be one of
  - **`PATH`**
  - **`HOME`**
  - **`PWD`**
  - **`USER`**
  - **`NODE_PATH`**
  - **`LD_LIBRARY_PATH`**

1.3.4.3. Unset/Delete Environment Variables

In the current environment variables list, when you check any of the variables, you will get the option to unset/delete the variable.

- **Unsetting** a variable removes it from the currently selected environment.
- **Deleting** a variable removes it from all environments.

You can unset/delete multiple variables at the same time.

1.3.4.4. Push Environment Variables

After finishing updating the values of the variables, you need to push the updated variables to the cloud by clicking on the "Push Environment Variables" button. This will also cause the app to be restarted.

1.3.4.5. Referencing Environment Variables In Code

An environment variable can be referenced in cloud code directly using process.env.variable name. For example

```
// PIN_ENABLED is the environmental variable created in the Studio.
var pin_enabled = process.env.PIN_ENABLED  || true  // defaults to true
if no value defined
```

If you wish to obtain the value of the variable from the client side, the best method is to export a function on the cloud side returning the value of the variable and calling the function from the client side. For example

```
// On the cloud side using node.js
exports.getEnvVariable = function(params, cb){
  var pin_enabled = process.env.PIN_ENABLED  || true

  return cb(null, {
    enabled : pin_enabled
  });
};

// On the client side
$fh.act({
  act : 'getEnvVariable',
  req : {}
  }, function(res){
   // run this in the event of success.
}, function(err){
   // run this in the event of failure.
});
```

1.3.5. Data Browser

The Data Browser section of the App Studio allows a developer to

- Graphically and interactively view the data associated with their app,
- View, create and delete, import and export collections,
- Modify data in a collection.

For more information, see the Data Browser Guide.

1.3.6. Deploy

The Deploy section allows a developer to:

- Deploy the cloud code of the current App and view a log of the deploy
- Choose (if more than 1 available) which environment to deploy the cloud code to
- Choose (if available) which git 'branch' or 'tag' to deploy
- Choose (if available) which Runtime & Version to run the cloud code with
- Perform a clean deploy, where the entire deploy directory (including dependencies) is removed and redeployed to from scratch
- Choose whether the app should be deployed automatically on future Git pushes to a selected branch

The Deploy process for Node.js applications typically does the following:

- Retrieves the cloud code at the specified branch or tag reference
- Gathers all environment variables values (both internal and custom), and any other config (for example, runtime) for the chosen environment
- Pushes the cloud code, environment variables values and any other config to the chosen environment
- Cloud code is extracted/copied to the relevant deploy folder
- Environment variables are configured in the app startup script
- Any other config sent along is acted upon for example, configured the correct runtime in the startup script
- Dependencies are resolved and installed (from package.json)
- The App is started using the startup script

The steps above will differ depending on the options chosen, the type of App and the presence & content of certain files.

In addition, developers may also configure the Platform to deploy cloud code to OpenShift Online. For more information, see the Staging Cloud Apps to RedHat OpenShift Online PaaS guide.

1.3.7. Stats

There are 2 types of stats available in the Studio: App Endpoint stats and Custom Stats. These are found in the 'Stats' section for any Cloud Apps.

There are some common UI features in the charts that represent this stats data. The data can be downloaded in different formats (for example, csv). The date range can be altered using the scrollbar and sliders below each chart. Each line in a chart can be toggled on/off by clicking that item in the legend.

1.3.7.1. App Endpoints

App Endpoint stats are available for all Apps without any additional setup required. Any endpoints defined in **`main.js`** will begin to produce stats as soon as the endpoint is called. A combined chart for 'All Endpoints' will also become available when stats are produced.

For each endpoint that is producing stats, the following summary information is available:

- Requests per minute - Number of requests in the last minute (calculated value based on smaller intervals)
- Average Request Time - Average time per request in the last interval
- Average Concurrent Requests - Average number of concurrent requests being processed in the last interval (calculated based on requests per interval and average request timer per interval)

The default chart view shows the following data series:

- Requests per second - Number of requests processed by the App per second (based on last interval)
- Average Concurrent Requests - Average number of concurrent requests being processed in the last interval (calculated based on requests per interval and average request timer per interval)
- Average Request Time (90th percentile) - Average duration of the requests in the last interval, ignoring outliers

The other graph items, Longest Request Time and Shortest Request Time, can be shown, by clicking on their names in the legend above the graph. The graph below show those stats included, and the Requests per second and Average Concurrent Requests, deselected.

1.3.7.2. Custom Stats

Custom stats are defined by the developer using the $fh.stats API.

1.3.7.2.1. Counters

Counters can be viewed on a line graph, showing the counter value for each interval.

Note

Counters are reset after each interval

1.3.7.2.2. Timers

Timers can also be viewed on a line graph. Each timer value is the average value (90th percentile) of that timer for that interval. The upper and lower values in each interval are also graphed on separate lines. These value give an overview of how the timer varies around the average time.

1.3.8. Notifications and Events

An app will generate events when certain actions are invoked against it. Those events will be recorded by the platform and presented to the developers through the App Studio and FHC. For example, when an app is deployed by an developer, the platform will record when the deployment happened, who invoked it and what are the results.

Each event has a category, a severity level and a name. The following events are generated by the platform at the moment:

Event Name	Event Category	Event Severity	Description
CREATE_REQUESTED	APP_STATE	INFO	App creation is requested
CREATED	APP_STATE	INFO	App is created
CREATE_FAILED	APP_STATE	ERROR	App creation failed
DEPLOY_REQUESTED	APP_STATE	INFO	App deploy is requested
DEPLOYED	APP_STATE	INFO	App is deployed
DEPLOY_FAILED	APP_STATE	ERROR	App deploy failed
START_REQUESTED	APP_STATE	INFO	App start is requested
START_SUCCESSFUL	APP_STATE	INFO	App started
START_FAILED	APP_STATE	ERROR	App start failed
SUSPEND_SUCCESSFUL	APP_STATE	INFO	After one week of inactivity due to zero REST API calls between the App and the Platform, the App is stopped. Next REST API call automatically starts the App.
STOP_REQUESTED	APP_STATE	INFO	App stop is requested

Event Name	Event Category	Event Severity	Description
STOP_SUCCESSFUL	APP_STATE	INFO	App is stopped
STOP_FAILED	APP_STATE	ERROR	App stop failed
CRASHED	APP_STATE	ERROR	Uncaught exception is thrown from the app and causes it to stop. It will be restarted automatically by the system monitoring process.
KILLED_RESTARTED	APP_STATE	ERROR	App is stopped then started
TERMINATED	APP_STATE	FATAL	App is terminated by the system due to too many restarts within a short time (currently 10 restarts in 20 seconds). It will not be restarted by the system monitoring process.
DELETE_REQUESTED	APP_STATE	INFO	App deletion is requested
DELETED	APP_STATE	INFO	App deleted
DELETE_FAILED	APP_STATE	ERROR	App deletion failed
CHANGE_REQUESTED	APP_ENVIRONMENT	INFO	Environment variables change is requested
CHANGE_SUCCESSFUL	APP_ENVIRONMENT	INFO	Environment variables changed

Event Name	Event Category	Event Severity	Description
CHANGE_FAILED	APP_ENVIRONMENT	ERROR	Environment variables change failed

1.3.8.1. System Events View

You can view all the events generated by the platform in the "Notifications" section in the Cloud App view in the Studio. You can also use the filters to search for events. More information of an event is available when you click on an event entry.

1.3.8.2. Alerts & Email Notifications

Email notifications can be sent to developers when certain cloud events occur. This is done through alerts. An alert is to define in what condition an email notification should be triggered and what email addresses should be used.

1.3.8.2.1. Alerts View

In the alerts view, there are two tables. The first table will list all the alerts that are created for this app, and the second table will show all the cloud events that matches the selected alerts' settings. More information of an alert is available when click on the entries in the first table.

1.3.8.2.2. Creating Alerts

When click on the "Create An Alert" button, you will be presented with the alert creation view. In this view, you can specify the name of the alert, what events the alert should be monitoring on and the email address to receive the alert.

When specify the events, you can use any of the three criteria (event categories, severities and names), or any combination of them. You don’t need to select values in all these fields. Once an alert is created, if an event matching the criteria occurred, an email notification will be sent to the address specified in the alert.

1.3.8.2.3. Notifications View

The platform will keep records of all the email notifications that have been sent and you can view them in the Studio as well. Click on the table entry will show the full details of the notification.

1.3.9. Logs

Users can access current and archived App logs created in each App environment.

1.3.10. Endpoint Security

The endpoint security feature allows you to decide the level of security you wish to apply to your App’s endpoints.

1.3.10.1. Managing Endpoint Security

You can find the endpoint security option in the left hand menu for a Cloud Instance. There are a couple of ways to configure security.

1.3.10.1.1. App Security

App security defines a default level of security that will be applied across all of your endpoints. The default is HTTPS. This means all requests to your App's endpoints will be sent over HTTPS. The App API Key options means that HTTPS will still be used, but that in order to access an endpoint the App API Key must be sent and must match the key created with your App. You can find this key under the details menu option for your App. This key is sent by default from the RHMAP SDKs. Enabling this option will mean your endpoints are only accessible when a correct App API Key is sent. If is not present, a 401 (Not Authorised) http response will be sent and your endpoint will not be called.

1.3.10.1.2. Individual Endpoint Security

You can change the security applied to an App's endpoints at an individual level also. When you update the list, your Cloud Instance will be restart.

Note

If you have a running App and have not staged it for sometime, that you may need to re-stage the App once in order to pick up the latest version of the software.

1.3.10.2. The Audit Log

In order to allow you to transparently see what has been done to your App, whenever a change is made to your security setting, an entry is created in the audit log. You can view this log by clicking the Audit Log tab at the top of the Endpoint Security screen. The audit log shows you a clear log of what has happened with your App's security settings. The filters allow you to get a smaller selection of the audit log.

1.3.10.3. CORS Support

Cross-origin resource sharing (CORS) is a mechanism that allows JavaScript on a web page to make XMLHttpRequests to another domain, not the domain the JavaScript originated from. For more on CORS, see CORS on Wikipedia. RHMAP automatically enables CORS for all cloud requests.

You can restrict this access to a domain of your choice as follows:

- Under the 'Cloud Management' section of the Studio, click on 'Environment Variables'
- Add a new Environment Variable called **`CORS_WHITELIST`**
- Set its value to be the domain you wish to restrict access to
- Hit the 'Push Environment Variables' button to apply the change to the currently selected environment

1.4. MBAAS SERVICES

1.4.1. Overview

A Service is a running Cloud App used by projects to integrate with back-end systems for example, an Oracle integration service. Services are added to one or more projects to make them available to Apps in that project.

Requirements

In order to be able to provision an Cloud Service, the Service Administrator Role must be assigned to an account.

In order to be able to create an instance of a Cloud Service, users must have either a Developer Admin Role or Developer Role assigned to their account, in order to be able to create and manage projects. Services can then be added to a project in the same way an App or Cloud Instance is added.

Related Resources

- Invoking a service via an API call
- Local Development with Services

1.4.2. Provisioning an MBaaS Service

Service Administrators can provision new MBaaS microservices that allow developers access to various back-end systems. Users can choose from a number of available services. After a service has been provisioned, it is then available to be used within Projects on the Platform. By default, when a service is created, it is marked as private. In order for the service to be available to all projects on a domain, the 'Global Service' option must be enabled. This declares the service as being publically available.

1.4.3. Details

Here you can access general overview information related to a specific service.

1.4.3.1. Service Details

The Service Details section provides you with overview information.

- Service ID: This is the unique identifier for a particular service. It is used to reference a particular service through FHC.
- Service API Key: This is used in conjunction with App Id to allow the App to communicate with the Cloud.
- Git URL: URL to clone repository.

1.4.3.2. Service Settings

The Service Settings section allows users to edit details related to the service.

- Name: The name of the service.
- Projects: A list of projects that have access to the service.

- Global Service: Selecting this option makes the service globally available to all projects on the domain. If this setting is enabled, developers will be able to select the service for use within their projects.
- Delete Service: Deletes a service.

1.4.4. Docs

The *Docs* section contains an API documentation viewer, which clearly shows important aspects of the service's API for each of its endpoints, such as:

- expected request headers and an example of a request body,
- an example of a response for the sample request,
- a complete snippet of a **`$fh.service`** call to invoke the method on the given service,
- a form for invocation of the service directly from the documentation viewer.

All of this is automatically generated if the root directory of the service contains a **`README.md`** file adhering to the API Blueprint specification. To learn more about the specification, take a look at the API Blueprint Tutorial or refer to the Language Specification for a full reference.

1.4.5. Editor

The built in Editor allows users to edit cloud code related to their service instance. The built in syntax highlighting feature makes it easier to distinguish between syntactical elements. Users can create new files and folders, with saved changes being automatically pushed back to the corresponding Git repository.

After a change has been made, the preview automatically refreshes so you can instantly see your changes take effect. Changes can also be pulled in from a Git Repository by performing a 'git pull'.

The App Preview in the Editor section allows users to view how the current App appears on a number of selectable devices.

1.4.6. Environment Variables

You can define environment variables using the Studio and those variables can be accessed in the cloud code. The Studio will show a list of current defined environment variables and current deployed environment variables for each cloud environment. You can also view the list of environment variables defined by the system.

1.4.6.1. App Environment Variables

This section lists the variables defined by you. These variables may not yet have been pushed to the cloud so the values here maybe different from the values that are currently active. Any inconsistency between the defined values and deployed values will be highlighted in red color. You can click on the value field to update it.

1.4.6.2. Deployed User Environment Variables

This section lists the current deployed environment variables created by you. Again, any inconsistencies will be highlighted. However, those values can not be updated directly.

1.4.6.3. Deployed System Environment Variables

This section lists the current deployed environment variables created by the platform. Those values are for your reference only and you should not change the values of those variables.

1.4.6.4. Create Environment Variables

You can create a new environment variable by clicking on the "Add Variable" button. The name of the environment variable should follow these rules:

- Should NOT begin with **`FH_`** or **`MONIT_`**
- Should NOT be one of
  - **`PATH`**
  - **`HOME`**
  - **`PWD`**
  - **`USER`**
  - **`NODE_PATH`**
  - **`LD_LIBRARY_PATH`**

1.4.6.5. Unset/Delete Environment Variables

In the current environment variables list, when you check any of the variables, you will get the option to unset/delete the variable.

- **Unsetting** a variable removes it from the currently selected environment.
- **Deleting** a variable removes it from all environments.

You can unset/delete multiple variables at the same time.

1.4.6.6. Push Environment Variables

After finishing updating the values of the variables, you need to push the updated variables to the cloud by clicking on the "Push Environment Variables" button. This will also cause the app to be restarted.

1.4.7. Data Browser

The Data Browser section of the App Studio allows a developer to

- Graphically and interactively view the data associated with their app,
- View, create and delete, import and export collections,
- Modify data in a collection.

For more information, see the Data Browser Guide.

1.4.7.1. Enabling the Data Browser

Selecting the Data Browser tab in the App Studio will present you with one of three screens.

- The Data Browser, if you have already enabled the Data Browser.
- A screen asking you to enable the Data Browser.
- A screen asking you to migrate your data to enable to Data Browser.

Clicking on the enable button will perform the following steps:

- Your app is stopped.
- A new database is created specifically for your app.
- An environmental variable is set for your app containing the raw connection string to the mongodb.

If your app's data needs to be migrated, you will see a screen with a migrate button. This button will do the following:

- Your app is stopped. This is to ensure no more data can be written during the migrate.
- A new database is created specifically for your app.
- Your existing data is migrated from the old database to the new one.
- Your data is validated in the new database.
- An environmental variable is set for your app containing the raw connection string to the app's MongoDB instance.
- Finally, if everything has succeeded, your old data is removed.

Note

You may also need to update the contents of your cloud/application.js and your cloud/package.json. If this is the case you will be informed on the migrate screen.

After all data migration steps have completed, you will be asked to re-stage your app.

1.4.7.2. Using the data browser

1.4.7.2.1. Viewing/Adding Collections

The collections associated with an app can be viewed by selecting the Data Browser tab in the Cloud Management section of the Studio.

This screen has two controls located at the top of the collection list, with buttons for:

- Adding a collection.
- Refreshing the list of collections.

Clicking on the button to add a collection prompts you to enter the collection name. Click on the Create button to create the collection.

1.4.7.2.2. Viewing Data In A Collection

To view the data stored in a collection simply click on one of the collections listed in the Data Browser. This view shows the data associated with the collection. At the top of the screen are the main listing functions:

- Switch Collection. Selecting this option presents you with a list of collections for the app. Click on a collection to list the data in that collection.
- Add an entry to the collection.

1.4.7.2.2.1. Sorting Data

To sort the data by a specific field, simply click in the field name at the top of the list. Sorting will alternate between ascending and descending order.

1.4.7.2.2.2. Filtering Data

To filter the displayed data, select the "Filter" button at the top of the Data Browser screen. Clicking this button displays the filtering options. These options allow you to filter the displayed data by one or more fields.

1.4.7.2.3. Editing Data

Editing data in the Data Browser can be done using either the Inline or Advanced Editor

- The Inline Editor is used to edit simple data in a collection (for example, changing the text in a single field).
- The Advanced Editor is used to edit more complex data types. This can be done using an interactive Dynamic Editor or a Raw JSON editor.

1.4.7.2.3.1. Editing Using the Inline Editor

To edit an entry using the Inline Editor, select the Edit option to the right of a data entry and select Edit Inline. The option will turn to a green tick and black arrow icons. When a field is too complex to edit in the Inline Editor, the "Advanced Editor Only" text is shown. This field is editable only in the Advanced Editor.

When finished updating the entry, select the green tick button to commit the changes to the data or the black arrow button to cancel any changes made.

1.4.7.2.3.2. Editing Using the Advanced Editor

The advanced editor is used to edit more complex data types (for example, where a field is composed of multiple nested fields).

To open the advanced editor, select the Edit option to the right of a data entry and select Advanced Editor. The advanced editor has two modes

- A Dynamic Editor to interactively add/edit fields.
- A Raw JSON Editor to directly edit the data in JSON format.

1.4.7.2.3.3. Editing Using the Dynamic Editor

The Dynamic Editor is an interactive editor for JSON data. It presents a structured view of each field to allow adding/editing complex data types. The actions menu provides all the functionality needed to manage complex fields for the entry:

- Type: The type option changes the data type of the field to an array, JSON object or string. It is also possible to set the field to auto, where the data type is automatically selected from the data entered.
- Sort: The sort option sorts the sub-fields of a complex type in ascending or descending order.
- Append: The append option adds a field after the object selected.
- Insert: The insert option inserts a field before the object selected.
- Duplicate: The duplicate option copies the object selected and appends it to the end of the selected object.
- Remove: The remove option deletes the field from the entry.

1.4.7.2.3.4. Editing Using the Raw JSON Editor

The Raw Editor allows for editing the JSON representation of the data. It is important to ensure that the data entered is in valid JSON format. The JSON data can be displayed in either formatted or compact form.

1.4.8. Deploy

The Deploy section allows users to manage the deployment of their current mBaaS code. Deployment targets can be created and selected, and deploys can be staged to either the live or development environments. As the Cloud Instance is being deployed, a deployment log is displayed, allowing users to closely examine all details of the deploy.

1.4.9. Stats

There are 2 types of stats available in the Studio: App Endpoint stats and Custom Stats. These are found in the 'Stats' section for any Cloud Instances.

There are some common UI features in the charts that represent this stats data. The data can be downloaded in different formats (for example, csv). The date range can be altered using the scrollbar and sliders below each chart. Each line in a chart can be toggled on/off by clicking that item in the legend.

1.4.9.1. App Endpoints

App Endpoint stats are available for all Apps without any additional setup required. Any endpoints defined in 'main.js' will begin to produce stats as soon as the endpoint is called. A combined chart for 'All Endpoints' will also become available when stats are produced.

For each endpoint that is producing stats, the following summary information is available:

- Requests per minute - Number of requests in the last minute (calculated value based on smaller intervals)
- Average Request Time - Average time per request in the last interval
- Average Concurrent Requests - Average number of concurrent requests being processed in the last interval (calculated based on requests per interval and average request timer per interval)

The default chart view shows the following data series:

- Requests per second - Number of requests processed by the App per second (based on last interval)
- Average Concurrent Requests - Average number of concurrent requests being processed in the last interval (calculated based on requests per interval and average request timer per interval)
- Average Request Time (90th percentile) - Average duration of the requests in the last interval, ignoring outliers

The other graph items, Longest Request Time and Shortest Request Time, can be shown, by clicking on their names in the legend above the graph. The graph below show those stats included, and the Requests per second and Average Concurrent Requests, deselected.

1.4.9.2. Custom Stats

Custom stats are defined by the developer using the $fh.stats API.

1.4.9.2.1. Counters

Counters can be viewed on a line graph, showing the counter value for each interval.

Note

Counters are reset after each interval

1.4.9.2.2. Timers

Timers can also be viewed on a line graph. Each timer value is the average value (90th percentile) of that timer for that interval. The upper and lower values in each interval are also graphed on separate lines. These value give an overview of how the timer varies around the average time.

1.4.10. Notifications and Events

An app will generate events when certain actions are invoked against it. Those events will be recorded by the platform and presented to the developers through the App Studio and FHC. For example, when an app is deployed by an developer, the platform will record when the deployment happened, who invoked it and what are the results.

Each event has a category, a severity level and a name. The following events are generated by the platform at the moment:

Event Name	Event Category	Event Severity	Description
CREATE_REQUESTED	APP_STATE	INFO	App creation is requested

Event Name	Event Category	Event Severity	Description
CREATED	APP_STATE	INFO	App is created
CREATE_FAILED	APP_STATE	ERROR	App creation failed
DEPLOY_REQUESTED	APP_STATE	INFO	App deploy is requested
DEPLOYED	APP_STATE	INFO	App is deployed
DEPLOY_FAILED	APP_STATE	ERROR	App deploy failed
START_REQUESTED	APP_STATE	INFO	App start is requested
START_SUCCESSFUL	APP_STATE	INFO	App started
START_FAILED	APP_STATE	ERROR	App start failed
SUSPEND_SUCCESSFUL	APP_STATE	INFO	After one week of inactivity due to zero REST API calls between the App and the Platform, the App is stopped. Next REST API call automatically starts the App.
STOP_REQUESTED	APP_STATE	INFO	App stop is requested
STOP_SUCCESSFUL	APP_STATE	INFO	App is stopped
STOP_FAILED	APP_STATE	ERROR	App stop failed

Event Name	Event Category	Event Severity	Description
CRASHED	APP_STATE	ERROR	Uncaught exception is thrown from the app and causes it to stop. It will be restarted automatically by the system monitoring process.
KILLED_RESTARTED	APP_STATE	ERROR	App is stopped then started
TERMINATED	APP_STATE	FATAL	App is terminated by the system due to too many restarts within a short time (currently 10 restarts in 20 seconds). It will not be restarted by the system monitoring process.
DELETE_REQUESTED	APP_STATE	INFO	App deletion is requested
DELETED	APP_STATE	INFO	App deleted
DELETE_FAILED	APP_STATE	ERROR	App deletion failed
CHANGE_REQUESTED	APP_ENVIRONMENT	INFO	Environment variables change is requested
CHANGE_SUCCESSFUL	APP_ENVIRONMENT	INFO	Environment variables changed
CHANGE_FAILED	APP_ENVIRONMENT	ERROR	Environment variables change failed

1.4.10.1. System Events View

You can view all the events generated by the platform in the "Notifications" section in the Cloud App view in the Studio. You can also use the filters to search for events. More information of an event is available when you click on an event entry.

1.4.10.2. Alerts & Email Notifications

Email notifications can be sent to developers when certain cloud events occur. This is done through alerts. An alert is to define in what condition an email notification should be triggered and what email addresses should be used.

1.4.10.2.1. Alerts View

In the alerts view, there are two tables. The first table will list all the alerts that are created for this app, and the second table will show all the cloud events that matches the selected alerts' settings. More information of an alert is available when click on the entries in the first table.

1.4.10.2.2. Creating Alerts

When click on the "Create An Alert" button, you will be presented with the alert creation view. In this view, you can specify the name of the alert, what events the alert should be monitoring on and the email address to receive the alert.

When specify the events, you can use any of the three criteria (event categories, severities and names), or any combination of them. You don’t need to select values in all these fields. Once an alert is created, if an event matching the criteria occurred, an email notification will be sent to the address specified in the alert.

1.4.10.2.3. Notifications View

The platform will keep records of all the email notifications that have been sent and you can view them in the Studio as well. Click on the table entry will show the full details of the notification.

1.4.11. Logs

Users can access current and archived service logs created in each App environment.

1.4.12. Endpoint Security

The endpoint security feature allows you to decide the level of security you wish to apply to your App’s endpoints.

1.4.12.1. Managing Endpoint Security

You can find the endpoint security option in the left hand menu for a Cloud Instance. There are a couple of ways to configure security.

1.4.12.1.1. App Security

App security defines a default level of security that will be applied across all of your endpoints. The default is HTTPS. This means all requests to your App’s endpoints will be sent over HTTPS. The App API Key options means that HTTPS will still be used, but that in order to access an endpoint the App API Key must be sent and must match the key created with your App. You can find this key

under the details menu option for your App. This key is sent by default from the RHMAP SDKs. Enabling this option will mean your endpoints are only accessible when a correct App API Key is sent. If is not present, a 401 (Not Authorised) http response will be sent and your endpoint will not be called.

1.4.12.1.2. Individual Endpoint Security

You can change the security applied to an App's endpoints at an individual level also. When you update the list, your Cloud Instance will be restart.

Note

If you have a running App and have not staged it for sometime, that you may need to re-stage the App once in order to pick up the latest version of the software.

1.4.12.2. The Audit Log

In order to allow you to transparently see what has been done to your App, whenever a change is made to your security setting, an entry is created in the audit log. You can view this log by clicking the Audit Log tab at the top of the Endpoint Security screen. The audit log shows you a clear log of what has happened with your App's security settings. The filters allow you to get a smaller selection of the audit log.

1.4.12.3. CORS Support

Cross-origin resource sharing (CORS) is a mechanism that allows JavaScript on a web page to make XMLHttpRequests to another domain, not the domain the JavaScript originated from. For more on CORS, see CORS on Wikipedia. RHMAP automatically enables CORS for all cloud requests.

You can restrict this access to a domain of your choice as follows:

- Under the 'Cloud Management' section of the Studio, click on 'Environment Variables'
- Add a new Environment Variable called **`CORS_WHITELIST`**
- Set its value to be the domain you wish to restrict access to
- Hit the 'Push Environment Variables' button in order for the changes to apply

CHAPTER 2. MOBILE BACKEND

2.1. MBAAS TARGETS AND ENVIRONMENTS

2.1.1. Overview

An MBaaS (Mobile Backend as a Service) is a runtime space for Cloud Apps and Cloud Services. Each MBaaS provides a Node.js runtime, a NoSQL database, a caching server, and all the Cloud APIs.

To support application lifecycle management, there's an additional layer of isolation provided for Cloud Apps and Cloud Services, called *Environments*. You can configure an environment for every life-cycle stage of your project, such as *Development*, *Testing*, and *Production*. Cloud Apps, Cloud Services and Forms are always deployed in a particular Environment, and can be easily promoted from one Environment to another.

One MBaaS can host one or more Environments. For example, you could host Development and Testing environment together on a single MBaaS, but have a separate MBaaS dedicated for a Production Environment. Even when multiple Environments are hosted on a single MBaaS they only share system resources but still remain isolated on most levels — separate Node.js runtime, database, caching server, and also a separate host name.

2.1.2. Creating an MBaaS Target

Navigate to the *Admin* > *MBaaS Targets* section.

There are three types of MBaaS targets: *FeedHenry*, *OpenShift 2* and *OpenShift 3*. Each type requires its own specific configuration as described below.

Note

An MBaaS target type cannot be changed after its creation.

2.1.2.1. FeedHenry MBaaS Target

- **MBaaS ID:** A unique ID for internal reference.

  This parameter cannot be changed after the MBaaS target creation and cannot contain any special characters or uppercase letters.

- **MBaaS Label:** A custom label used within the Studio.
- **MBaaS URL:** The host name of the MBaaS target.
- **Username:** The user name to be used when logging into the instance.
- **Password:** The password to be used when logging into the instance.
- **Service Key:** A key used to identify the system with the FeedHenry MBaaS.

2.1.2.2. OpenShift 2 MBaaS Target

- **MBaaS ID:** A unique ID for internal reference.

This parameter cannot be changed after the MBaaS target creation and cannot contain any special characters or uppercase letters.

- **MBaaS Label:** A custom label used within the Studio.
- **MBaaS URL:** The host name of the public OpenShift instance (for example **`openshift.redhat.com`**).
- **Username:** OpenShift 2 user name.
- **Password:** OpenShift 2 password.
- **fh-mbaas Host:** Host name of a running fh-mbaas service.

Note

If you plan to use Forms in your apps, you need to set the *fh-mbaas Host* field. This requires a few additional steps to deploy an fh-mbaas service on your OpenShift instance. See Enabling Forms support in OpenShift Targets.

2.1.2.3. OpenShift 3 MBaaS Target

- **MBaaS ID:** A unique ID for internal reference.

  This parameter cannot be changed after the MBaaS target creation and cannot contain any special characters or uppercase letters.
- **MBaaS Label:** A custom label used within the Studio.
- **OpenShift Master URL:** URL where the OpenShift Master API is available.
- **Username:** OpenShift 3 user name.
- **Password:** OpenShift 3 password.
- **OpenShift Router DNS:** OpenShift 3 Router wildcard DNS entry, for example **`*.local.feedhenry.io`**.
- **Automatic MBaaS Installation**: The automatically installed MBaaS is not optimized for resiliency and stability required in production environments. For production use, follow the manual installation steps.
- **MBaaS Service Key**: Value of the **`FHMBAAS_KEY`** environment variable in the fh-mbaas service.
- **MBaaS URL:** Exposed route where fh-mbaas is running in OpenShift.

After you set up an MBaaS target, you also **must** add at least one Environment. Navigate to the *Admin* > *Environments* section to add an Environment.

2.1.3. Creating an Environment

Navigate to the *Admin* > *Environments* section.

First, create a new environment using the *Create Environment* button. You need to set the following parameters:

- **Environment ID:** A unique ID for internal reference.

This parameter cannot be changed after the environment creation and cannot contain any special characters or uppercase letters.

- **Environment Label:** A custom label used within the Studio.
- **MBaaS Target:** The MBaaS Target to be used by the Environment. An Environment can only use one MBaaS target.

Cloud Apps, Cloud Services and Forms can now be deployed to this new Environment.

2.2. CLOUD RESOURCES

2.2.1. Overview

In the Platform, when an app is staged or deployed, it will be running inside a container, also known as a Cloud Environment. Environment resource usage at both the container level and the app level can be managed and viewed here.

Requirements

The user must be a member of one or more teams with the following permissions:

- Domain — Cloud Resources (View & Edit)
- Domain — Projects (View)

Related Resources

- FHC CLI
- Platform API - EndPoint Security API

2.2.2. Summary View

Once the page is loaded, the resource summary view is presented. You can see what cloud environments are available in your domain and current overall usage of CPU, memory and disk of an environment. Each summary view can be expanded by clicking on it.

2.2.3. Resources

You can view the detailed usage of memory, CPU and disk resources in this view. The "Dashboard" view contains an overall view of all three resources and each resource’s tab contains both the environment and per-app usage.

2.2.4. Apps

You can see all the apps that are currently running inside the environment, see their status, resource usages and manage them as well. Note you will be able to control embed apps (deployed through the build embed app wizard) here as well.

2.2.5. Cache

You can see the current cache usage of the environment, flush the cache and reset maximum cache size.

2.3. PUSH NOTIFICATIONS

2.3.1. Overview

Push notifications are a mechanism which lets server-side applications initiate communication with mobile clients - to notify the user about availability of new content on the server, or even to send a short message. This is commonly used, for example, in messaging and social network applications, or any other situation where the user might wish to be informed at all times, even when not handling the mobile device.

Red Hat Mobile Application Platform (RHMAP) has integrated support for push notifications. Through a single unified API you can deliver notifications to all client devices, even across multiple different networks simultaneously with a single function call. Push notification support in RHMAP is based on the open-source project AeroGear UnifiedPush Server (UPS).

To start using push notifications now, see the Push Notifications Client API and cloud API documentation, or try the step-by-step guide Developing An Application Using Push Notifications.

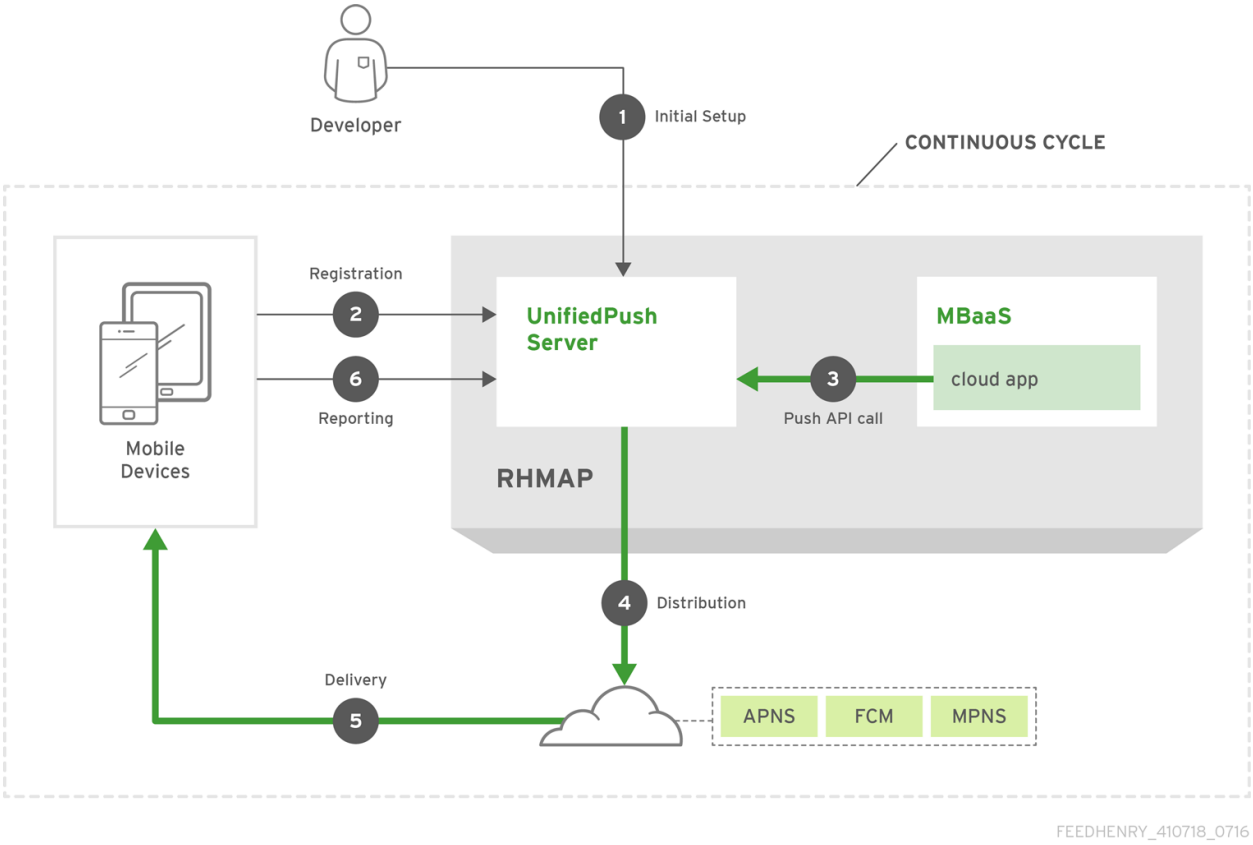
2.3.2. Supported Platforms

Support for push notifications is available for all supported mobile platforms through their corresponding push notification networks:

- **Android** — Firebase Cloud Messaging (FCM) (previously Google Cloud Messaging - GCM)
- **iOS** — Apple Push Notification Service (APNS)
- **Windows**
  - Microsoft Push Notification Service (MPNS) for native Windows Phone 8 apps and for Cordova apps on Windows
  - Windows Push Notification Service (WNS) for native Windows 8.1 and Windows 10 Universal Apps

2.3.3. How It Works

The push notification server inside RHMAP acts as a broker that distributes notifications from your Cloud App to individual push networks, which subsequently try to deliver messages to registered mobile devices.



1. **Initial setup**

   - Enable push notification support, separately for each Client App
   - Enable all targeted *variants* for push notification networks, such as FCM, APNS or MPNS and provide credentials for access to the networks, such as certificates and private keys. See Obtaining credentials for push networks for more details on setting up individual networks. For Cordova apps, create a variant for each targeted platform, just like for native Client Apps.

2. **Registration**
   Upon start-up, the Client App registers with the push server by calling **`$fh.push`** and thus becomes ready to receive notifications. During registration, identification and preferences of the client are sent to the server.

3. **Push API call**
   A Cloud App sends a push notification by calling **`$fh.push`**. The API lets you target all, or only specific Client Apps in your project.

4. **Distribution**
   The call to **`$fh.push`** is handled by the integrated push server which distributes the notification to all corresponding push networks, depending on the variants enabled in targeted Client Apps.

5. **Delivery**
   Individual push notification networks handle the actual delivery to mobile devices.

6. **Reporting**
   Metadata is reported back to RHMAP, including the information that the notification was delivered and whether it was opened by the end user.

Note

The fact that push notifications pass through 3rd party networks has some implications on the design of applications using them:

- **Delivery:** the delivery to the target mobile devices can't be guaranteed by RHMAP, since most push networks do not guarantee delivery.
- **Security:** it is recommended never to send any sensitive or confidential information as part of a push notification. See Security Considerations for more information.

2.3.4. Sending notifications

The simplest use case is delivering a mass notification to all devices, as demonstrated in the following example:

```
$fh.push(
  {alert: "Hello from {ProductShortName} to all devices!"},
  {broadcast: true},
  function (err, res) {
    if (err) {
      console.log(err.toString());
    } else {
      console.log("status : " + res.status);
    }
  }
);
```

The Platform is capable of several modes of delivery: all devices, a filtered set of devices, or a single device. You can also choose which Client Apps in your project receive push notifications.

2.3.4.1. Client App filtering

In the cloud-side **`$fh.push`** call, you need to explicitly choose one of two modes of notification delivery:

- **broadcast** - notification delivered to all Client Apps in the project
- **selected client apps** - only chosen Client Apps, identified by their App IDs, will receive the notification

See the cloud Push API documentation for examples and full documentation.

2.3.4.2. Recipient filtering

When registering with the server, Client Apps report their identification and preferences to enable filtering of notifications. You can choose to deliver push notifications to all registered devices, or only to a specific subset, filtered by the available criteria:

- **alias** - user identification, such as a user name or an e-mail representing a single person (with possibly multiple devices). Intended to enable a *unicast* communication model.
- **categories** - additional filtering criterion, enabling a *publish-subscribe* communication model

- **device type** - automatically set by the client device; a short identifier of the mobile device hardware
- **variant ID** - automatically configured by the platform; can be used for addressing only chosen push networks

See the client Push API documentation for examples and full documentation.

2.3.5. Security Considerations

2.3.5.1. Notification contents

As noted before, the payload of the push notification is delivered to third-party push notification network providers, like Google or Apple. It is highly recommended not to send any sensitive personal or confidential information belonging to an individual (for example, a social security number, financial account or transaction information), or any information where the individual may have a reasonable expectation of secure transmission, as part of any push notification.

If sensitive content needs to be delivered to the device, the recommended solution is only to send a notification informing the user about availability of new content on the server. Such content can then be securely transmitted to the device through a separate trusted channel once the user reacts to the notification.

2.3.5.2. Variant secrets

A configuration file containing the **variant ID** and **variant secret** is distributed in Client App binaries to mobile devices. These credentials authorize the Client App to register with the push server. These keys should be kept secure and should not be distributed through any other channels than through the Client App binary.

In case the variant secret key is ever compromised, it can be invalidated and replaced with a new one. You can perform the key renewal in Studio, by opening a Client App's Push section, Variants tab and clicking on *Renew Variant Secret*.

> **Warning**
>
> Note that after renewing the variant secret key, existing installations can no longer use push notifications and the new key has to be redistributed through a new version of the Client App binary.

2.3.6. Obtaining Credentials for Push Networks

For the built-in UnifiedPush Server (UPS) to be able to access the push notification networks of individual providers - Google, Apple and Microsoft - it needs to supply credentials for authentication with the provider's push server. The required credentials and the ways to obtain them are very different for each provider. Follow the guide for your target platform and save or make note of the required credentials, which you'll need to supply to RHMAP when setting up push network variants for your app.

2.3.6.1. Android

For step-by-step setup instructions, follow the Obtaining FCM Credentials Guide

Required credentials:

- Server API key
- Project number

2.3.6.2. iOS

For step-by-step setup instructions, follow the APNs Push Notifications with AeroGear's UnifiedPush Server Guide

Required credentials:

- Client SSL Certificate
- Certificate passphrase

The guide asks you to provide a *Bundle ID* - a unique identifier of an iOS app. If you created your iOS app in Xcode and imported it into RHMAP, you most likely already have a bundle ID set up. See About Bundle IDs in official Apple documentation for more information on bundle ID and how to set it in Xcode.

If you created your iOS app in RHMAP, your app got a default bundle ID assigned, which needs to be changed to a custom value. The bundle ID is set in the property list file of your project, located at **`<name-of-your-project>/<name-of-your-project>-Info.plist`**, under the **`CFBundleIdentifier`** key.

For Cordova apps, the bundle ID can be set in Studio. In the Config tab of your app, set the Bundle Identifier field for your build target - iOS, iPhone, or iPad.

> **Warning**
>
> The generated Client SSL Certificate for push notifications is bound to a particular bundle ID. Therefore a change in bundle ID would require re-generation of the certificate.

2.3.6.3. Windows

MPNS

No setup is required.

For step-by-step setup instructions, follow the WNS Push Notifications with AeroGear's UnifiedPush Server Guide

Required credentials:

- Package SID
- Client secret

2.3.7. Analytics

Statistics about push notifications sent to a Client App can be found in the Studio. Navigate to a Client App's *Push* section and then to the *Analytics* tab. The available statistics include:

- number of push messages sent to the integrated push server for dispatching to registered devices,
- number of notifications dispatched via push networks to registered devices,
- number of delivered notifications opened by a user,
- average open rate,
- number of currently registered devices.

The Push Analytics screen also provides charts for:

- open rate,
- breakdown by push networks.

2.3.8. Building Cordova apps with Push Notification Support

Support for push notifications in Cordova apps is provided through the **`aerogear-cordova-push`** plugin. When a push-enabled app is built in the RHMAP Build Farm, this plugin is added automatically for the particular build if it's configured in the **`www/config.json`** file. All RHMAP Cordova templates have this plugin configured.

When building the app locally, you need to add the plugin manually using the following command:

```
cordova plugin add aerogear-cordova-push
```

See Using Cordova Plugins for more information.

2.3.9. Community Version Of UnifiedPush Server

The open-source community version of AeroGear UnifiedPush Server can still be used in your apps, by deploying UPS to OpenShift and connecting it to the Platform through the Cloud Service *AeroGear Community Push Connector* which is available in the Platform for this purpose. See Developing a Push Notification Application using AeroGear UnifiedPush Server for a detailed walk-through of the process. However, note that the community version is only available for evaluation purposes and is not officially supported.

CHAPTER 3. DRAG AND DROP APPS

3.1. CREATING FORMS USING THE FORMS BUILDER

Overview

The Forms Builder allows you to manage existing forms and create new forms from a template.

3.1.1. Creating a Form

All forms are created from templates.

When creating a form, you have the option to create an empty form using the *Blank Form* template. There are also pre-populated form templates for different scenarios that may be more convenient to customize.

You can preview a template to see all its fields. Once you select a template, you can enter a form name, and an optional description.

Note

The form preview is for illustration purposes only. To submit forms or save drafts, you need to create a new forms project or associate forms with an existing project.

When the form has been created using the *Create Form* button, you are taken directly to the *Edit Form* screen to customize the form.

3.1.2. Adding Fields to a Form

Forms are created through the inclusion of fields. There are a variety of different fields that can be added, each with a different purpose. Listed below are the fields that can be added to a form, along with a brief description. Fields have a variety of different configuration settings. Depending on the field selected, different field configuration settings will be available.

Fields are split into two categories:

- content fields — take user input (text, number, file, etc.)
- structure fields — auxiliary fields which help you create the form layout

Fields can be added to a form by:

- Drag & Drop — Drag the field type anywhere on the form to place the form. When dragging the field, you will see a dotted outline of where the field will be placed when finished dragging.
- Clicking a field type — Single-clicking on a field will add the field after the currently selected field already in the form.

After fields have been added to a form, they can be repositioned by dragging them to the desired place in the form.

3.1.2.1. Structure Fields

3.1.2.1.1. Section Break

Section Breaks are used to group fields into sections. In the default template, sections are collapsible groups of fields. Sections can contain a name and description to identify the section to the form user.

Not all pages have to contain section breaks. It is possible to have a form with some pages containing section breaks and some that do not.

Note

If a page contains a section break and there are fields between a page break and section break, those fields will automatically be grouped into a section.

3.1.2.1.2. Page Break

Page Breaks are used to indicated where a new page of a form begins. Any fields after a page break will be contained in the next page of the form. Like sections, pages can contain a name and description to identify the page to the form user.

3.1.2.2. Content Fields

Content fields are fields that users will enter data into.

Each field, as described below, can contain type-specific configuration settings. However there are a number of settings that are universal to all fields.

- Required — A required field must be submitted when completing a form.
- Validate Immediately — This option allows the value entered into the field to be validated when the value is entered. Otherwise the field is validated when the user clicks *Submit* on the last page of the form.
- Admin Only — Admin Only fields are visible only on the submission editor in the Studio. Admin Only fields will not appear in the form in the Client App. In addition, Admin Only fields cannot be the subject of any field or page rules.
- Repeating — Repeating fields contain a configurable number of entries between *Min* and *Max*. Repeating fields cannot be the subject of field or page rules.
- Field Code — Fields can be assigned a user-defined code that identifies the field. This must be unique within the form. This is useful for adding data to a field from external sources (for example, CSV files).

3.1.2.2.1. Text Field

A text field allows users to enter a single line of text. Valid input includes letters, numbers or special characters.

There are two validation options for text fields:

- Length Limit — The *Min* or *Max* number of characters that must be present in the field. If left blank, no validation will be performed on the length of the text input.

- Field Format — the format of the text field, defined as a combination of letters and numbers represented by **c** and **n** characters, respectively, or a regular expression to be matched to the field.

3.1.2.2.2. Paragraph Field

Similar to the text field, the Paragraph field also accepts all types of input, including letters, numbers, and special characters. Additionally, the Paragraph field spans across multiple lines.

There is one validation option for paragraph fields:

- Length Limit — The *Min* or *Max* number of characters that must be present in the field. If left blank, no validation will be performed on the length of the text input.

3.1.2.2.3. Number Field

The Number field is a single line field that allows the user to enter a numeric value. Only numbers can be entered in the number field. Letters or special characters will not be accepted.

There is one validation option for number fields:

- Min/Max Value — The range of numbers that can be entered into the field. If left blank, no validation will be performed on the size of the number input.

3.1.2.2.4. Email Field

The Email field allows the user to input an email address. Both @ and a . characters must be found. At least one character must also be entered for the domain. For example, **`test@example.com`** is valid.

3.1.2.2.5. Website Field

The Website field allows the user to enter the URL of a website.

3.1.2.2.6. Dropdown Menu

The Dropdown Menu allows users to select a single option from a drop-down list of options.

Options can be added by adding inputs into the *Options* section of the field configuration tab. Pre-selected values can also be added by selecting the check box beside the added option.

3.1.2.2.7. Radio Buttons

The Radio Button field allows users to select a single option from a list of radio buttons.

Options can be added by adding inputs into the *Options* section of the field configuration tab. Pre-selected values can also be added by selecting the check box beside the added option.

3.1.2.2.8. Check boxes

The Check box component allows the user to select any, all, or none of the options from a list.

Options can be added by adding inputs into the *Options* section of the field configuration tab. Pre-selected values can also be added by selecting the check box beside the added option.

There is one validation option for the check boxes field:

- Selected Options Limit — The *Min* and *Max* number of options that can be submitted.

3.1.2.2.9. Map

The Map field uses the Google Maps API to calculate and display the user's current position. The user's position on the map is indicated by a marker.

3.1.2.2.10. Location

The Location field lists the coordinates of the user's location as displayed in the Map field. Co-ordinates can be displayed in either Latitude/Longitude or Eastings/Northings format.

3.1.2.2.11. File

The File field allows the form user to attach a file to a submission.

There is one validation option for the file field:

- Max File Size — The maximum size, in kilobytes, of the file that can be uploaded using this field. If no value is entered, then the file size will not be validated.

3.1.2.2.12. Photo Capture

Allows the end user to upload an image. The user has an option to either take a photo and upload it, or to choose from a library.

There are several options available when defining a Photo Capture field:

- Max Height — The maximum height of the photo taken, in pixels.
- Max Width — The maximum width of the photo taken, in pixels.
- Quality — The quality of the photo. Note: Modern smart phone cameras can take very high-resolution photos. This can increase the file size of the photo submitted.
- Photo Source — Source of the photo can be limited to camera, library or both.
- Photo Type — Photos can be taken as either JPEG, or PNG files.
- Save To Photo Album — Allows any photo taken by a camera to be saved to the photo library for future use.

3.1.2.2.13. Signature Capture

This field is used to capture the user's signature.

3.1.2.2.14. Datestamp

Allows the user to enter a date and time.

There are three options when defining a Datestamp field:

- Date and Time — Allows the user to enter the date and time. The format of the input is configurable. The default is **`YYYY-MM-DD HH:mm`**.

- Date Only — Allows the user to enter a date value only.
- Time Only — Allows the user to enter a time value only.

3.2. MANAGING FORMS PROJECTS

3.2.1. Overview

The *Forms Projects* screen allows you to view any projects that are associated with forms or themes and create a new forms project.

Projects can be searched by name in addition to viewing the number of forms related to each project.

You can create a Forms Project in one of two ways:

- Create the project from scratch.
- Add a form to an existing project.

For more information, see the How to create a Forms Project guide.

Requirements

To manage forms projects, the user must be a member of one or more teams with the following permissions:

- Domain — Drag & Drop Apps (View & Edit)
- Domain — Projects (View & Edit)

3.2.2. Creating a Forms Project

You can create a new Forms project in one of two ways.

3.2.2.1. Creating a New Project

To create a new forms project, click *Create New Project* button, enter a name for the new project and select the *Next* button. The new project will contain a single forms client and Cloud App to immediately interact with the associated forms. When the new forms project is created you will be directed to configure the project with forms and a theme.

3.2.2.2. Associating with an Existing Project

Forms can be associated with any project in the Studio. To associate forms with an existing project, click *Use Existing Project*, select the project you wish to associate forms with, and click *Next*.

When the existing project is selected, you will be directed to configure the project with forms and a theme.

3.2.3. Adding Forms and Themes to a Project

The *Project Forms & Themes* screen allows you to associate forms and a theme with the project.

You can associate a theme by selecting a theme from the drop-down list. To preview a theme, select the theme in the *Project Theme* list and click one of the forms in the*Forms* list. This will render the form into the preview area.

To add forms to the project, click into the forms list. You will be presented with a list of available forms. Typing part of the form name will filter the list. When you have found the form, select it from the list. You will now see it in the forms associated with the project.

When you are finished adding forms, click on the *Save* button to update the project.

3.2.4. Configuring a Forms Project

The *Advanced Config* screen allows you to configure Client and Cloud Apps.

Note

Any updates to these settings will update live Client and Cloud Apps in this project.

3.2.4.1. Client Options

Client options are available to the forms Client App and affect the operation of forms on the client.

These settings provide the default values for the Client App, however app users can be allowed to change these setting by adding the device id to the **`Admin Users`** section described below.

Note

In the forms template app, the **Settings** section contains a **Show Device Id** button that will display the current device id to the app user.

3.2.4.1.1. Camera Settings

Camera setting affect the operation of the *Photo Capture* field. This section allows you to edit the height, width and quality of the photos taken.

Note

These settings are overridden by any **height**, **width** or **quality** settings set on the **Photo Capture** field.

3.2.4.2. Submission Settings

This section allows you to define some parameters for when apps are submitting form data:

- Max Retries – The maximum number of times that a Client App will try to submit a form before alerting the user to a submission error.
- Timeout – The number of seconds that will cause a submission upload to report an error if exceeded.

- Min Sent Items To Save – A list of options the client user has when selecting the number of sent form items to keep on-device.

3.2.4.3. Client Logging

This section allows you to define some parameters of client logging. These settings are useful if there is an issue with submitting forms.

- Logging Enabled – If this check box is checked, all Client Apps will store logs on-device.
- Logging level – Allows you to set the level of verbosity of client logging. **debug** will have the most logs whereas **`error`** will only log errors on the client.
- Log Limit – The maximum number of log entries that are permitted on the client before logs rotate.
- Log Email – The email address the client can send logs to if required.

3.2.4.4. Admin Users

Admin users are client device IDs that are permitted to alter configuration on device.

To add a device ID to the list, click into the text box and enter the device ID.

Note

In the forms template app, the **Settings** section contains a **Show Device Id** button that will display the current device ID to the app user.

3.2.4.5. Cloud Logging

Enabling cloud logging will enable logging for any Cloud App related to this project. This is useful when there is an issue with submitting forms.

3.3. DYNAMICALLY DISPLAYING FORM FIELDS AND PAGES

3.3.1. Overview

Fields in a form can be hidden or displayed dynamically based on values of one or more other fields.

Pages can also be hidden or displayed based on a predefined set of rules. When applying rules to pages, the 'object' affected by the rule will always be a page. This means that although the subject of the rule will be a field, a page will always be affected. Rules are applied in the same way they are applied to fields.

3.3.2. Defining Rules

For example, a rule could be applied to an age field, where if the input is an age of 17 or older, a text field asking the user to list previous jobs appears. If the user is too young to have had a job, the field is hidden.

- Subject — This is the field to which the rule is applied.

- Comparison — The comparison determines how the subject's value is compared to the given value. The comparisons include *contains*, *does not contain*, *begins with*, *ends with*, and others.
- Value — This is the value that the rule will be tested against.
- Action — The action to be taken if the condition is satisfied. The action can be either show, or hide.
- Object — The field or page that will be shown or hidden if the condition of the rule is met.

When constructing rules, different comparisons can be selected depending on the type of field the rule is being applied to. For example, a text field and a number field will have different comparison options. When applying a rule to a text field, a rule can be constructed based on whether an input begins with a particular letter, contains certain letters, or ends in a particular letter, whereas when applying a rule to a number field, rules are constructed based on whether or not input to a particular field is greater than, less than, or equal to a particular value.

Different types of fields have different rule comparisons available for selection.

- textfield — is, is not, contains, does not contain, begins with, ends with
- paragraph — is, is not, contains, does not contain, begins with, ends with
- number — is greater than, is less than, is equal to
- email — is, is not, contains, does not contain, begins with, ends with
- web — is, is not, contains, does not contain, begins with, ends with
- dropdown — is, is not, contains, does not contain, begins with, ends with
- radio buttons — is, is not, contains, does not contain, begins with, ends with
- checkbox — is, is not
- date stamp — is at, is before, is after

Note

Map, Photo capture, Signature capture, File upload, and Location fields can not have rules applied to them, however they can be displayed or hidden based on input to other fields.

3.4. USING DATA SOURCES

3.4.1. Overview

To make your forms applications more dynamic, you can set certain field types in Forms to have values loaded dynamically. The values can be loaded from a database, from an external web service, from a back-end system, or any other source accessible through an MBaaS service.

Data sources are the mechanism which binds a form field to an MBaaS service which provides the values.

The following field types can have values loaded from a data source:

- Dropdown

- Check boxes
- Radio Buttons
- Read Only

For a step-by-step guide to defining and using a data source, follow the guide Dynamically Populating Form Fields From an MBaaS Service.

3.4.2. How it works

The data source acts as a binding layer between the form field, and the MBaaS service endpoint. When a form definition is accessed, the data for fields backed by a data source are not retrieved directly from the MBaaS service. Instead, the data source always returns a version of the data cached in the environment.

For example, when a data source is invoked from a form deployed in a *Dev* environment, the associated MBaaS service must also be deployed in a *Dev* environment, and the data is cached only for the *Dev* environment.

3.4.2.1. Refresh Period

The cached data from the MBaaS service is refreshed periodically on the cloud side, according to the configured period – from as little as one minute, up to seven days.

If a data source refresh fails for any reason, a back-off algorithm will progressively increase the refresh period to a maximum of seven days. Any successful update to the data source in that time, either from an automatic or manual refresh, will allow the data source to resume the configured period.

3.4.3. Requirements

- The MBaaS service used as a data source must use **`fh-mbaas-api`** version **`5.8.1`** or higher
- The Cloud App associated with a form, which uses a data source, must use **`fh-mbaas-api`** version **`5.0.0`** or higher

3.4.3.1. JSON Format

The HTTP endpoint of the MBaaS service used as a data source must return a JSON response in the following format:

- For *Dropdown*, *Check boxes*, and *Radio Buttons* field types:

  A list of objects with three keys:

  - **`key`** – (Optional) internal identifier, used as the **`name`** attribute in the HTML form field
  - **`value`** – the user-friendly display text
  - **`selected`** – (Optional) used to preselect a single option in case of *Radio Buttons* and *Dropdown*, or multiple options for *Check boxes*

```
[
  {
    "value": "January",
```

```
    "selected": "true"
  },
  {
    "key": "02",
    "value": "February"
  },
  {
    "value": "March"
  },
  ...
]
```

Note

There is no limit on the number of objects returned by the service, and all returned objects get rendered in the form as options. Using too many options can make the form difficult to navigate.

- For *Read Only* text field type:

```
[
  {
    "value": "<b>Note:</b> You can use <i>HTML</i> in the Read Only
field."
  }
]
```

A subset of HTML is supported in the **`value`** and will be rendered in the form.

If the list contains more than one object, the values of the **`value`** fields from all objects are separated by newlines and concatenated.

3.4.3.2. MBaaS Service Endpoint

You can use any MBaaS service as a data source, as long as it uses the required version of **`fh-mbaas-api`** and the output of the HTTP endpoint conforms to the required format, as described above.

Three MBaaS service templates are provided by default, which demonstrate common use cases:

- Static Data Source – serves data statically defined in a JSON file in the MBaaS service
- MongoDB Data Source – serves data stored in the built-in MongoDB database, loaded through **`$fh.db`**
- Internet Data Source – serves data parsed from an external website

3.5. THEMES

Overview

Themes are used to customise the appearance of a form. Form Editors can create and manage themes, as well as associating them with forms to personalize form appearance. Elements of a theme that can be customised include the theme logo, background colours, typography, and border settings.

3.5.1. Creating a Theme

When creating a theme, select a theme from a list of templates.

When a theme template is selected, a preview of the theme is displayed. Once a template has been selected, the user then enters a theme name, and description (optional). The theme can now be edited to customise the appearance of the form.

3.5.2. Editing Themes

3.5.2.1. Logo

You can upload a logo via the file upload option in the Logo section. Any image to be used as a logo must be in either the PNG or JPEG format.

3.5.2.2. Backgrounds

In the *Background Color* section, you can change background colours for different parts of the theme. The colour can be set by clicking on the coloured square beside the section title, and then selecting a colour from the picker. The update will take place once the colour picker has been closed.

3.5.2.3. Typography

The Typography section allows for the customization of font size, type, and text decorations. The colour of a selected field can also be changed by clicking on the coloured square beside the section title. Clicking the coloured square displays a colour picker.

3.5.2.4. Borders

Customizations can be made to the border sections of a theme. You can make customizations based on colour, border type and border width.

3.5.2.5. Spacing

Certain fields have options for customizing the margin and padding between fields.

Note

To get a preview of how changes made to the various sections affect the theme, select a form from the drop-down menu at the bottom of the preview area.

3.6. VIEWING SUBMISSIONS

3.6.1. Overview

The *Submissions* page is used to view forms that have been submitted either via Studio or via a Client App. Submissions can be viewed on a per Form basis, per Project basis, or by Recent Submissions.

Form Submissions can be viewed using the Submissions Viewer, and edited by the Submissions Editor.

Submissions can be exported in Comma Separated Value (CSV) format using the *Export Submissions* button. When exporting Submissions, the headers for the exported fields can be

- **Field Name** — The name of the field.
- **Field Code** — The Field Code assigned to the field. If no Field Code has been assigned to the field, the **Field Name** will be used instead.

Note

Field Names and **Field Codes** are assigned to a Field within a Form using the Form Builder.

The Recent Submissions page also shows a list of forms that have been recently submitted. A variety of submission information is provided along with the Form and Project name. This includes the current environment the Project is deployed on, the date/time the submission was received, and the input to the form fields.

Clicking on a submission will display the details of the submission.

3.6.2. Sorting Submissions

3.6.2.1. Sorting Submissions By Form

The *By Form* Submissions page allows the user to view Submissions on a per form basis. A user can select a form from the dropdown list, and then view information regarding all the individual submissions relating to that particular form.

After selecting a submission from the list, extra Submission Information is displayed. In the image below, a preview of the form is displayed along with additional Submission Information.

The Form Submission section shows a preview of the form that was submitted, along with the values that were entered for each field.

The Submission Information section displays more technical information such as what App ID, Device ID, and IP address are associated with the Submission.

Note

If a Submissions Editor makes any modification to a Form Submission this is recorded in the 'Date time edited' and 'Edited by' fields.

3.6.2.2. Sorting Submissions By Project

The Per Project Submissions page allows users to view submissions on a per Project basis. First, a Project is selected from the list. Then, a Form is selected from the list of Forms associated with that Project.

3.6.2.3. Filtering Submissions

The *By Form* and Per Project Submissions page contain a search field. The following fields will be compared to the value when the search button is clicked.

- The ID of the submission.
- The name of the form the submission was made against.
- The ID of the form the submission was made against.
- ID of the device that made the submission.
- The IP address the submission was made from.

3.6.3. Searching Submissions

Advanced submission search allows for searching form submissions with customized criteria. Users can build up clauses filtering the values of fields to find the relevant submissions.

3.6.3.1. Creating Search Criteria

To search submissions for a form:

1. Select a form from the list of available forms. This will populate the *Single Fields* and *Repeating Fields* boxes with a list of fields matching the type.
2. Add a clause by clicking the **+** icon and remove clauses by clicking the **-** icon.
3. Populate a clause by selecting the field, selecting the clause type (for example, is, is not, etc.) and enter the value to compare.
4. When all clauses have been created, click *Search* to find the list of submissions that match the created clauses.

Note

If the field is repeating, each entry of the repeating field will be compared to the clause.

Note

Submissions matching the created search criteria can be exported using the *Export Submissions* button.

3.6.4. Viewing Submission Details

All non-admin fields submitted by the user are displayed along with any images and files the user uploaded.

In addition, *Admin Only* fields are displayed for adding additional information to the submission.

Images are displayed inline. To save the image to your computer:

1. Right-click on the image.
2. Select *Save Image As*.
3. Select a location to save the image.

Files can be opened by clicking on the link in each file entry.

To save the file to your computer:

1. Right-click on the image.
2. Select *Save Link As…*.
3. Select a location to save the file.

The buttons at the bottom of the screen allow for 4 functions

- Download PDF: Allows for converting the submission to a PDF document. When the conversion is complete, this process triggers a download in the browser.
- Print Submission: Allows for converting the document to a printable format. When complete, this process triggers the print function in the browser.
- Delete Submission: The user is prompted for a confirmation to delete the submission.
- Edit Submission: Allows for editing a submission to add additional data.

Caution

Deleting a submission is irreversible. All content related to the submission is deleted, including files.

3.6.5. Editing a Submission

Editing a submission can be performed by entering data into the field (for example, text, number).

Admin fields are indicated using the *(Admin)* icon.

If a field is repeating, additional values can be added to the field with the *Add Input* button.

The contents of a file or a photo field can be changed by clicking *Choose File* below the file or a photo. The file or the photo can then be added from your local computer. When the new file has been chosen, a message will appear showing the file is queued for upload.

To save any changes made to the submission, click the *Update Submission* button. This will update the submission data in addition to uploading any files selected.

Clicking *Cancel* will abort any changes entered into the submission.

3.6.6. Receiving Notifications on Every Submission

The Notifications section allows users to manage email addresses that will receive an automated email once a submission is received.

Here, email addresses can be added to and removed from a notification list. Any email address that has been added to the list will receive an email whenever a copy of that Form is submitted.

CHAPTER 4. ADMINISTRATION AND MANAGEMENT

4.1. ACCOUNT

Overview

The Account Settings Section has a number of functions. Users can view account overview information such as the account type, and associated email. Users can also manage their account password settings from here. API and SSH Keys can also be managed from the Account section.

Requirements

There are no special requirements to be able to view user account information.

Related Resources

- FHC CLI
- Platform API - API Keys

4.1.1. Manage Account

This section of the Account area shows your account details such as Name, e-mail address and your account type. Account Preferences, such as re-enabling the Welcome introduction or always showing the context help, can be modified here. Your password can be changed on this page too.

4.1.2. SSH Key Management

Accessing your App repos via Git requires you to upload an SSH Public key. This section of the Account area helps you do just that. You can upload multiple keys if you have a different key on various machines, or want to allow someone access. Keys can also be searched, based on the label you gave it, or deleted.

Your SSH key is typically located in:

```
~/.ssh/id_rsa.pub
```

Or, to generate a new key use:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Warning

Any SSH key added here will have push & pull access to the Git repositories of every Project/App you can see in the Projects tab.

Warning

An SSH key can only be uploaded once. If you attempt to uploaded an already existing key, even in someone else's account, it will fail. This behaviour is essential for mapping keys to a Users account. If you need to use multiple SSH keys, see our guide for configuring your local SSH to use multiple keys.

4.1.3. API key Management

In a traditional web application the user authenticates with the server, and then a cookie is incorporated into all subsequent requests sent to that user. RHMAP replaces the traditional method and uses an API key which is similar to a cookie but is sent in a header called **`X-FH-AUTH-USER`**. This header must be incorporated into every request sent to the user.

User API keys can be viewed in the **My Account** → **My Profile** → **API Keys** section of the App Studio. In the **API Keys** section of the App Studio, users can create, list, edit and revoke their user API keys. When accessing the REST API programmatically, a valid user API key must be manually set in the "X-FH-AUTH-USER" header field for each request.

By default, you will have no User API keys (with the exception of any system created keys for example, **`FH_MBAAS_API_key`**). Multiple API keys can be created, so its a good idea to give each a unique label. The key value will be generated automatically. Labels can be modified later if required. If you no longer require a specific key or wish to prevent it from being used any more (for security reasons), simple Delete/Revoke it.

4.2. ADMINISTRATION

Overview

The Administration section of the platform allows Users, Auth Policies and Deploy Targets to be managed. It also allows you to control all aspects of distributing apps to end user devices as well as tracking and managing this usage.

Requirements

The user must be a member of one or more teams with the following permissions:

- Domain — Authorization Policy (View & Edit)
- Domain — Authorization Policy (View & Edit)
- Domain — Deploy Target(View & Edit)
- Domain — App Store (View & Edit)
- Domain — Administrator (View & Edit)

Related Resources

- FHC CLI
- Platform API - MAM

- Platform API - User Administration
- Platform API - User Roles

4.2.1. Users

All aspects of User management from creating Users, assigning Roles, disabling Users to deleting Users are available to Administrators.

A full list of available options:

- View, create and update a user's details, be they App users (used in conjunction with our $fh.auth API) or App Studio users
- Assign roles to users
- Disable users - rendering them unable to login
- Mark a user for data purge (used with $fh.auth, this flag can be checked to wipe data from a user's app/device)
- Send/re-send an invite email to a user
- Delete Users

4.2.1.1. Viewing Users

In the User list overview area, details such as the User's user id, name, email, active status and when they last signed in are shown. There is an option to search for a specific user, which updates automatically as you type. There is an option to create new User's, or edit a specific User. Additional User details are available in the user edit area.

A user also can be deleted from the list view. You have to confirm the deletion before it's actually executed. A user can only be deleted if there are no apps in the Studio created by that user. If there are still apps owned by that user, the deletion will throw an error.

4.2.1.2. Creating Users

When creating a new user you need to specify the user id. User id is the unique identifier of the user and it has to be unique in your domain and it cannot be changed. You will get an error message if the user id is already taken. You can also optionally specify email, name and password for the user. If you want an invitation email to be sent to the user, you need to specify the email address for the user. You can allow users to specify their own passwords by checking the "Send Invite Email" option.

Roles can be assigned to a new User as well. More details on User Roles is available below.

Users can also be associated with Auth policies in the "Auth Policies" section.

4.2.1.3. Updating Users

User fields such as their name, email and password can be updated. A User's password cannot be viewed, but it can be set to be a different value.

In addition, the **`Teams`** that the User is a **`Member`** of will be displayed. To add a User as a **`Member`** of a **`Team`**, click into the Teams box and select the team to add.

A user can be disabled or enabled in this view. Once a user is disabled, that user will not be able to authenticate in $fh.auth API. A special value will be returned by the API to indicate that the user has been disabled.

There is another option here called "Purge User Data". If this option is checked, two things will happen:

- The user will be disabled.
- Next time when the user login using $fh.auth, a special value will be returned by the API to indicate all the existing app data should be deleted (the data delete function is subject to developers to implement).

For more details, check the $fh.auth API.

4.2.1.4. Invitation emails

An invitation email contains a unique link, which when opened, allows a user to activate their account by inputting the password to use. The text of this email is typically something like:

```
Hello User,

Your account has been created and is awaiting activation.

To activate your account, please click on the link below where you can
also set your password.

  https://[your-studio-domain].feedhenry.com/studio/activate.html?
t=12345678901234567890123

Regards
```

When creating a new user, there is an option to automatically send an invitation email. Alternatively, there is an option in the update user area to resend invitation emails.

4.2.2. User Roles

Note

The permissions system in Red Hat Mobile Application Platform is based around the concept of teams, as opposed to the role-based system previously used in the FeedHenry v2 platform. If the **Teams** based permissions system is active, your User Profile in Studio will contain a **`Teams`** section. If not, the **Roles** based permissions system is active. For a comparison of Roles and Teams, click here.

User Roles are added to user accounts in order to grant them different privileges within the Platform. Users can perform different tasks, and access different sections of the Platform, depending on what roles have been assigned to their account. For example, only a user with the **`Analytics`** role can access the **`Analytics`** section of the Platform. Only **`Admin`** or **`Domain Admin`** users have the authority to add or remove user roles to an account.

4.2.2.1. Analytics (analytics)

- Can access the Analytics section of the Platform.
- Can monitor project usage.

4.2.2.2. Forms Editor (formseditor)

- Can create forms and themes.
- Can create Form Projects.
- Can edit forms, themes, and Projects associated with their group(s).
- Can associate forms and themes with projects in their group(s).

4.2.2.3. Forms Administrator (formsadmin)

- Can create forms and themes.
- Can create Form Projects.
- Can create groups.
- Can assign users to groups.
- Can view, access and manage all forms & themes created on a specific domain.
- Can view submissions from all projects on the domain.
- Can edit submissions from any project on the domain.

Note

It is the responisiblilty of the Forms Administrator to add users to groups. If a Forms Editor is not associated with a group, they will not be able to see any forms/themes they create, even though they have sufficient privileges to edit them.

4.2.2.4. Submission Viewer (submissionsviewer)

- Has access to view submissions from any project in their group(s).

4.2.2.5. Submission Editor (submissionseditor)

- Has access to view submissions from projects in their group(s).
- Has access to edit submissions from projects in their group(s).

4.2.2.6. Developer (dev)

- Can create and manage any type of Project.
- Can build App binaries for all available platforms.
- Can upload development resources such as private keys and certificates.
- Can create, edit, and maintain deployment targets.

- Can add public services to a project.

4.2.2.7. Developer Administrator (devadmin)

- Can create and manage any type of Project.
- Can build App binaries for all available platforms.
- Can upload development resources such as private keys and certificates.
- Can create, edit, and maintain deployment targets.
- Can view and access all Projects & Apps on a specific domain.
- Can view and edit all deployment targets on a specific domain.

4.2.2.8. Service Administrator (serviceadmin)

- Can provision mBaaS Services.
- Can manage mBaaS Services.

4.2.2.9. Domain Administrator (portaladmin)

- Can create and edit Auth Policies.
- Can update the App Store for their domain.
- Can create and manage App Store Apps.
- Can create and manage App Store Groups.
- Can manage App Store Devices.
- Can view App Store Logs.
- Can access the 'Mobile App Management' section on the Admin page.
- Can add and remove users.
- Can assign roles to users.

4.2.2.10. Customer Administrator (customeradmin)

- Can manage users in any domain belonging to this customer.

4.2.2.11. Reseller Administrator (reselleradmin)

- Can manage users in any domain belonging to any of its customers.
- Can assign the Customer Admin role to users.

4.2.2.12. Administrator (admin)

- Can perform any action in the Platform.
- Can assign Reseller Admin and Customer Admin roles to users.

4.2.3. Auth Policies

The Platform has a feature rich Authentication and Authorization mechanism. At the core of this is the concept of Auth Policies. An Auth Policy allows you to define how your Users will Authenticate themselves when accessing your Applications and App Store Apps, and also allows you to define various Authorization checks.

RHMAP supports the following Authentication providers:

- OAuth: Specifically, OAuth2, this allows you to authenticate your users against OAuth providers such as Google.
- LDAP: Both Active Directory and Open LDAP servers are supported, typically used for more 'Enterprise' type integrations.
- FeedHenry: The RHMAP platform authentication mechanism.
- MBAAS: Authenticating users via an mBaaS Service, which means you can use any authentication mechanisms.

4.2.3.1. Viewing Auth Policies

In the Auth Policy list overview area, details such as the Policy Id and Authentication Type are shown. There are options to create a new Policy or Edit an existing Policy.

4.2.3.2. Creating an Auth Policy

When you create an Auth Policy you need to specify an Id for the Policy, along with specific information for the Authentication Type.

4.2.3.3. OAuth Auth Type

OAuth Authentication allows for application developers to access user specific information from a third party service without that user having to divulge their username and password for accessing that service. The basic flow is as follows:

- The user wishes to login to your application and click the sign in with Google option.
- The application retrieves a request token from the provider and redirects the user with this token to the provider.
- The user logs in to their account and approves access to requested information and are redirected back to the application.
- The application receives an OAuth token and a OAuth verifier which it exchanges with the provider for an OAuth access token.
- The access token is sent with any future requests to access the user’s information.

4.2.3.4. LDAP Auth Type

The following LDAP Authentication methods are supported in the Platform:

- Simple: Simple Bind, where the users DN and password are sent to the LDAP Server. This option is supported out of the box with Active Directory and OpenLDAP.

- SASL: Two types of Simple Authentication and Security Layer are supported, Digest MD5 and CRAM MD5. Consult your LDAP servers documentation for information on the type of SASL it supports.

4.2.3.4.1. URL

The URL field is the address of your LDAP server, for example, ldap://foo.example.com:389. Note that '389' is the standard LDAP port.

4.2.3.4.2. DN Prefix

The 'Distinguished Name (dn) prefix' field is the prefix for the full user DN. The format of a Users full DN required for Authentication is typically: **`<prefix><user><dn>`**, for example, **`uid=fred,ou=people,dc=example,dc=com`**. The prefix is typically 'cn' for Active Directory or 'uid' for Open LDAP. Consult with your LDAP Administrator for what this setting should be. Note that this can also be blank, it is not a mandatory setting.

4.2.3.4.3. DN

The 'Distinguished Name (dn)' field can be used to provide the full directory path to the user, for example, the **`ou=people,dc=example,dc=com`** in the example above. Again, this can be blank, consult with your LDAP Administrator regarding this setting.

4.2.3.4.4. DN and DN Prefix

The DN and DN Prefix fields are used together to build the name that the platform uses to Bind to the LDAP server. In the example **`uid=fred,ou=people,dc=example,dc=com`**, the prefix could be set to **`uid=`** and the **`dn`** could be set to **`,ou=people,dc=example,dc=com`**. This would allow users to use **`fred`** as the user name. These fields can be left blank and the platform will attempt to bind using the data entered by the user. These fields can be used to construct whatever value that the LDAP server allows for binding, including the **`userPrincipalName`**. If the **`userPrincipleName`** is an email address, the **`dn`** field could be used to add a common domain name, for example, if the **`dn`** field is set to **`@example.com`** and a user uses a login name of **`fred`** the platform will attempt to bind to the LDAP server as **`fred@example.com`**.

4.2.3.5. Sample Active Directory Settings

Here are some sample instructions for how to Authenticate Users against your Active Directory (AD) server:

- First, your server needs to be accessible from the RHMAP Cloud. For private RHMAP deploys this is usually not an issue as all services will be behind the same firewall. For public RHMAP cloud, you will either need to set up a VPN for us to access your AD Server, you will need to contact us to discuss this. Alternatively, ensure your AD Server is publicly accessible.
- Next, create an Auth Policy and specify the following values:
  - By default, AD supports the 'Simple' Authentication Method out of the box, so choose 'Simple'as the Authentication Method.
  - In the URL field, enter the address of your AD server, for example, **`ldap://foo.example.com:389`**.
  - Leave the 'DN Prefix' and 'DN' fields blank. These are not required by default for AD. When a User authenticates, their email address alone is passed on to AD and this is normally

sufficient to identify the AD User.

4.2.3.6. FeedHenry Auth Type

You can use the Platform Authentication as a means of Authenticating Users. Using this method, a User must exist in the Platform. See the Users section for instructions on how to administer Platform Users.

There are no additional configurations options if you choose the FeedHenry Authentication Policy Type, simply specify the Authentication Type as 'FEEDHENRY'.

4.2.3.7. MBAAS Auth Type

This type supports user authentication via mBaaS Services. If your application needs an authentication type that is not supported by the Core platform, you can easily create new mBaaS services to support it.

4.2.3.7.1. Service

The name of the mBaaS service this auth policy should use. You should create the mBaaS service first and then select the name from the dropdown list.

4.2.3.7.2. Endpoint

The full path of an endpoint defined by the mBaaS service this auth policy should be calling.

The endpoint will receive the 'params' object specified in the Auth API in the request body, and it should return 200 if authentication is succeeded.

If Authorization is required, this endpoint should also return a JSON response with a **`userId`** key. The value of this key will be used to lookup user in the system for authorization checks.

4.2.3.7.3. Default Environment

The environment to use, if the environment value is missing in a request.

4.2.3.8. Authorization

Further to your User Authenticating themselves, you can also specify Authorization measures:

4.2.3.8.1. Check User Exists

This check will ensure the User is a valid User that exists on the Platform. See the Users section for more information on Administering Users in the Platform.

4.2.3.8.2. Check User is Approved

This simply allows you to add specific Users that you want to be Authorized to access your App. If the User is not in this list, they will not be allowed access to your App.

4.2.4. Mobile App Management

RHMAP contains a built-in App Store for distribution of binaries to mobile devices. Integrations for some third-party mobile application management solutions are also supported and listed in the section Third Party MAM/MDM.

4.2.4.1. App Store

The App Store gives end users the ability to install mobile apps for iOS, Android, and Windows directly onto their mobile devices.

4.2.4.1.1. Configuring the App Store

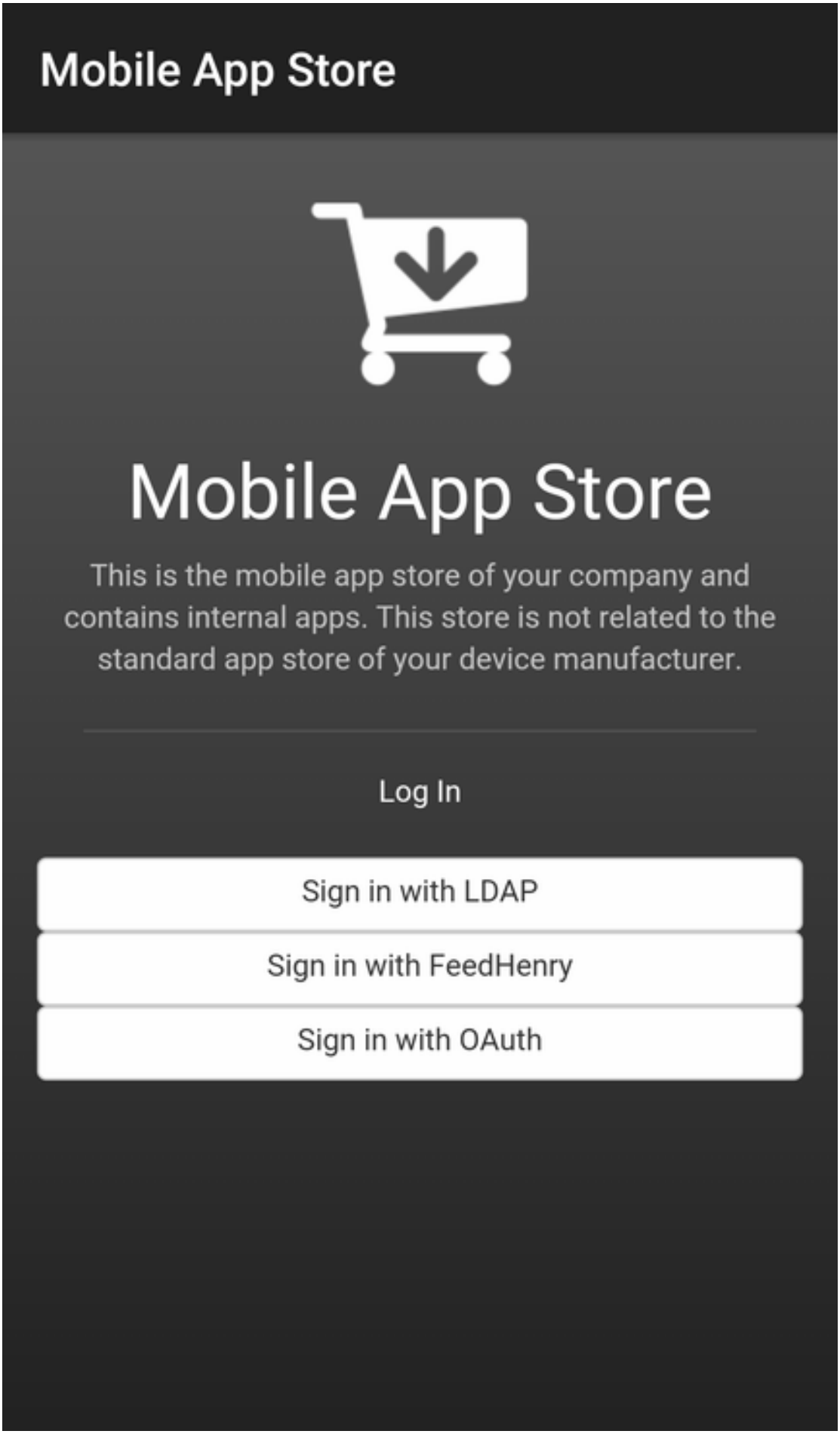
Each domain comes with an App Store. In order to use this App Store you will first need to update some of the details:

- Name: this name shown to users of the App Store
- Description: the description to display on the App Store landing page
- App Store icon: this is the icon that will be displayed along with your App Store
- App Store Apps: items available to install from the Store
- Auth Policies: used to configure which users can access the Store, and which Items they are allowed to install

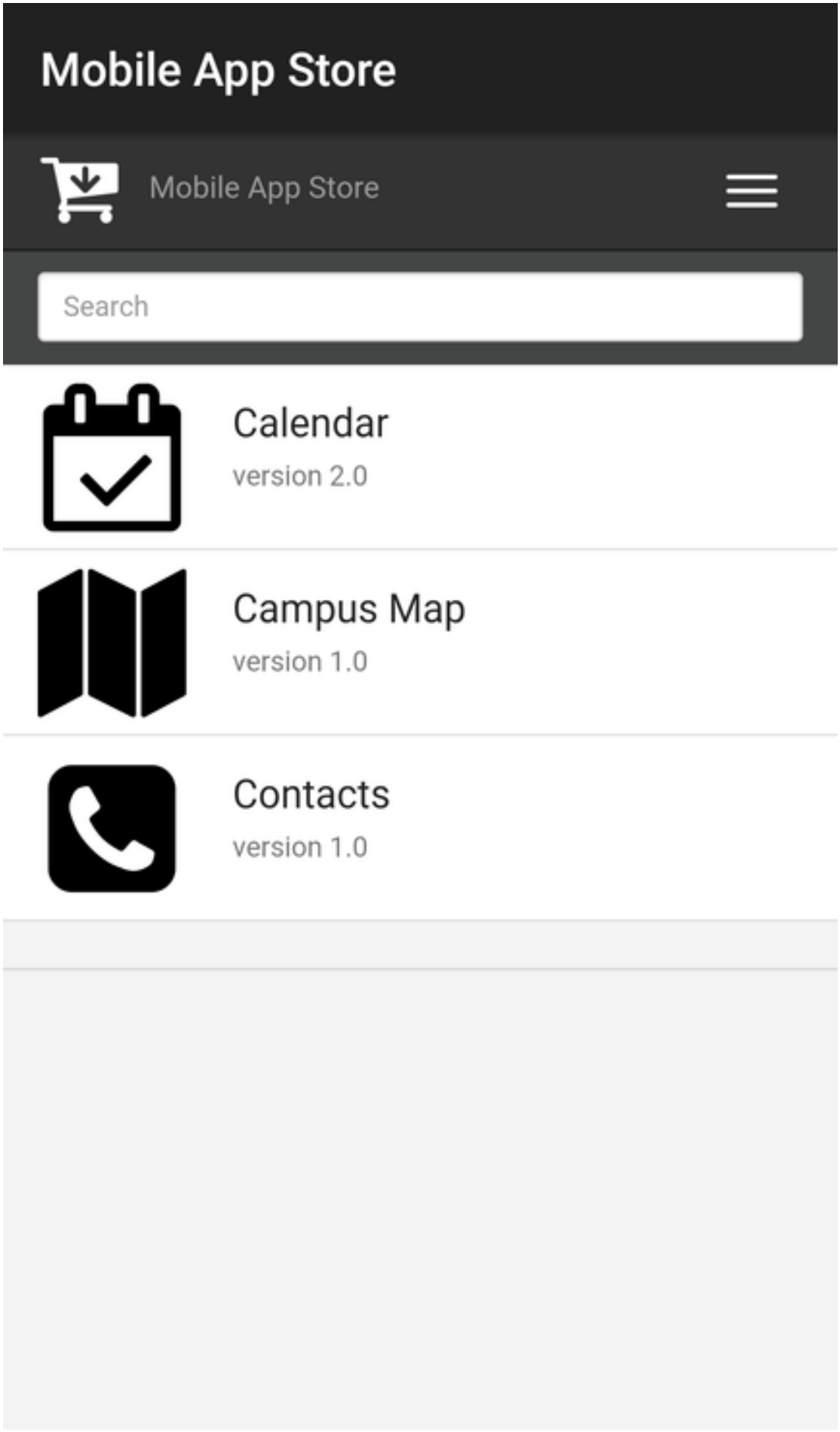
4.2.4.1.2. Using the App Store

To visit the App Store, go to the App Store URL on you mobile browser. This URL can be obtained from the *Admin > Mobile App Management > App Store* page, and is typically located at **`domain.feedhenry.com/store`**.

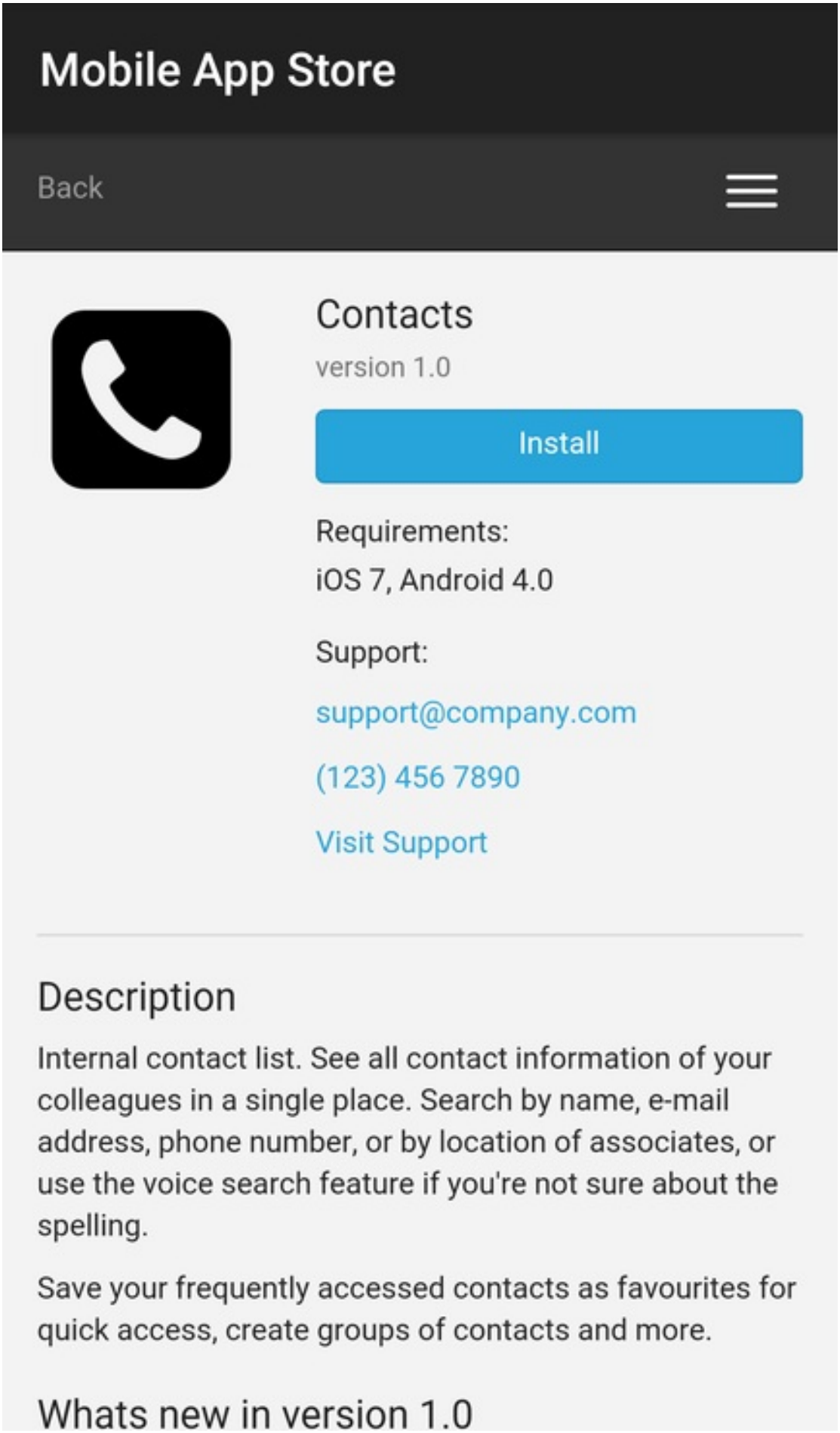
When you browse to the App Store URL on your device, you will be presented with the login screen of the Mobile App Store. This screen shows some details about the App Store, and a list of available authentication mechanisms for example, OAuth or LDAP. These mechanisms are preconfigured by following other parts of this guide. If no authentication mechanisms have been setup, it will not be possible to log in to the App Store, even if Apps have been added to it.



After logging in, a list of available Apps for your device will be shown.



Choosing an app will display more information, including an install button.



There are several possible installation binary types for iOS applications:

- iPhone - can be installed on iPhone, iPod Touch and iPad.

- iPad - can be installed on iPad.
- iOS - universal application which can be installed on iPod Touch, iPhone and iPad in one package.

The options available depend on the package produced at App build time. For Android apps, there is typically only 1 installation option available.

4.2.4.2. App Store Apps

App Store Apps (formerly Store Items) can be added for distribution via the App Store. You can upload native binaries for new and existing Apps for Android, iOS (Universal), iPhone & iPad. You can also assign icons to your App Store Apps which will appear in your App Store, and give your App a name and a description.

Here's a list of all actions for App Store Apps:

- Add new App Store Apps (typically mobile apps) for distribution.
- Upload native binaries for new and existing Apps.
- View the App binary history for existing Apps.
- View the Audit Logs for App binary history for existing Apps.
- Add Auth Policies to be used with the application.
- Add an icon for your App.
- Restrict a App to be displayed only to users in the same App Groups.
- Add a App to a App Group.

4.2.4.2.1. Adding App Store Apps

To create a new App, click on the App Store Apps option in the menu. A form with a Name field will appear. Once the field has been filled out, you can click the "Create Item" button.

4.2.4.2.2. Updating App Store Apps

To update an App, click the Edit button for the App. You will then be on the "Details" tab of the Update App page.

A form with various input fields will appear.

- The Name field: this is for the unique name of your App.
- The Icon field: click the "Choose File" button. Choose an appropriate icon and press OK. This icon will now upload immediately and be displayed along with your App.
- The Auth Token Field: this token is a unique string which is then used with the $fh.auth API to identify your application and perform the authentication checks you have specified in your Auth Policies.
- The Description Field: this field is for adding a description which will be visible to viewers of your App Store.

- Auth Policies: This swap select shows the policies you have available for use with your App Store Apps. These policies define authentication methods to be used to authenticate a user who is attempting to access your App application.
- Restrict to Groups: This check box controls the App Groups functionality for this App. If it is checked then the user must be in one of the same App Groups as the App.
- App Groups: This swap select shows the App Groups you have available for use with your App Store Apps. These groups restrict the visibility of these items to users who are in the same App Group or App Groups (assuming that "Restrict to Groups" is checked).

Once these fields have been filled out with the required information, you can click the "Update Item Details" button.

Note

If this is the first time you are updating a App after creating it, the Studio will show the Binaries tab on the assumption that you need to add a binary to make this App visible in your App Store.

4.2.4.2.3. Uploading Binaries

In order to upload a binary or view the history, you must first have an existing App.

4.2.4.2.3.1. Upload Binary

Click the edit button on one of your existing App Store Apps and then click the "Binaries" tab of the Update App page. To upload a binary for your App you will need to have a valid binary built either through the Platform or via Xcode or Eclipse etc…

Select the type of binary you will be uploading and click "Choose File". Select your binary file and the upload should begin. Once it has completed it will then be available as an option for download with that App.

4.2.4.2.3.2. iOS Binaries

Next to the iOS binary options, you will notice an empty configuration field. This field is for the bundle identifier of your iOS app. You can find this identifier in the App Studio under configuration and then iPhone | iPad | iOS Universal. It is labeled under Bundle Identifier.

Also for iOS based apps you will need to upload the .ipa file. If you are building your application using the Platform, you can download this file when you do an iOS build. If you are building the app with Xcode, the .ipa file can be obtained by using the archive option.

4.2.4.2.3.3. Windows Phone 8 Binaries

The Windows Phone 8 binaries uploaded here should be signed with a company certificate. If you want to distribute the AET file along with the binary, you can upload it as the configuration file for the binary. For more details, check out Deploying an App on Windows Phone 8.

4.2.4.2.3.4. Binary History

Click the edit button on one of your existing App Store Apps and then click the "Binaries" tab of the Update App page. To view the current and older binaries click the "[+] History" link of the desired binary. This will cause a list of the binary history to be displayed.

4.2.4.3. Third Party MAM/MDM

You can enable or disable integrations with individual mobile application management (MAM) and mobile device management (MDM) providers. Enabling the integration does not result in all binaries being automatically published. To enable publishing, see Publishing App Binaries to Third-party MAM and MDM providers.

4.2.5. Audit Log

Audit logs are available for information about the number of downloads of App App binaries. In order to view the Audit Log, you must first have an existing App.

Click the edit button on one of your existing App Store Apps and then click the "Audit Logs" tab of the Update App page. This will show the current Audit Log entries for this App. The full list of Audit Log entries for all the Apps can be viewed and more comprehensively processed by clicking the "Audit Log" menu item in the "Mobile App Management" menu on the left hand side of the page.

4.2.5.1. Searching & Filtering

The search box allows you to search through the audit logs. The options at the top of the tab allows you to filter Audit Logs by :

- App name
- user email
- App type (iOS, iPhone, iPad or Android)
- number of records to display (10, 100 or 1000)

Once the filter options are selected , pressing the "Filter" button will get the matching Audit Logs. Pressing the "Reset" button will get the default Audit Logs.

4.2.6. Groups

App Store groups (formerly Store Item Groups) can be created to restrict Apps to a particular set of users. Apps can be marked as restricted to a particular set of groups, and only users in those groups will be able to view those items in the app store.

The App Groups Administration section of the App Studio allows an administrator to:

- Create a new App Group
- Assign users to a group
- Assign App Store Apps to a group

4.2.6.1. Overview

Create your Users and App Store Apps as usual. When creating App Store Apps, you can mark them as being restricted to groups, you don’t have to select groups at this stage, since you may not have created them yet.

Once your Users and App Store Apps are created you can create your App Groups. The groups that you create could correspond to departments within your organisation, you might create a Sales group containing apps for your sales department, and Engineering group containing apps for your engineering department. Once the groups are created, it’s possible to assign App Store Apps and Users to them.

For example, you could assign your Sales users your Sales App Store Apps to your Sales group, so that when a Sales user logs into your App Store, they can see the Sales App Store Apps. Users and App Store Apps can be assigned to multiple groups, for example, your Sales and Engineering managers might be members of both groups, to get access to both departments App Store Apps. Of course, you don’t have to restrict your App Store Apps to groups, so that they will be visible to all users who log into your App Store.

4.2.6.2. Adding a New Group

To create a new App Group, click on the Groups option in Mobile App Management section of the menu, then click the "Create" button. A form with 2 input fields will appear.

- The Name field: this is for the unique name of your App Group.
- The Description Field: this field is for adding a memorable description to help describe the purpose of the group.

Once these fields have been filled out with the required information, you can click the "Create group" button.

4.2.6.3. Editing a Group

Groups can be edited to change the description text, or to assign users or Apps to the group. To edit your App Group, click on the Groups option in Mobile App Management section of the menu, then click the "Edit" button corresponding to the particular group to be edited.

- App Store Apps: items which are currently assigned to your group.
- Group Users: users which are currently assigned to your group.

4.2.7. Devices

Active Devices can be viewed, assigned friendly names, blocked or marked for data purge. Active Apps and Users on a Device are also available.

Here are all the actions that can be performed:

- View Active Devices.
- Assign friendly names to device Ids.
- View Apps in use on devices.
- View Users active on device.
- Block a device — prevents any apps or users from authenticating from that device.
- Purge device data - sends a kill pill signal to any app on the device the next time the app attempts to authenticate.

4.2.7.1. Viewing Devices

In the device list view, all the devices that have used $fh.auth to access apps will be listed here. The device label, device id, device status and the last access time will be displayed in the list. Clicking on any of the control buttons will bring you to the device update view.

The device id is a unique identifier of the device and it can not be changed. It is sent from the device and will remain the same across different apps on the same device (for hybrid or native apps). The only exception is web apps loaded in web browsers: the device id is a generated value and it is saved to the cookie. If the cookie is deleted, a new value will be generated.

4.2.7.2. Updating Devices

In this view, you can assign a friendly label name to a device, disable it from authentication or mark it for data purge. You can also check which users on the platform have been using the device and what apps have been accessed from the device.

If a device is disabled from authentication, all the apps on that particular device that are using $fh.auth to authenticate will fail and a special value will be returned to indicate that the device has been disabled. This is useful when a device is lost or stolen.

If the "Mark for Data Purge" option is checked, two thing will happen:

- The device is disabled from authentication.
- When apps on that device are using $fh.auth to authenticate, the authentication process will fail and a special value will be returned by the API to indicate all the existing app data should be deleted (the data delete function is subject to developers to implement).

4.2.7.3. Viewing Users of Devices

Every time $fh.auth is called from an app on a device, an access log entry will be created if the authentication process is succeeded and the user is a platform user. Thus you can see which users have been accessing from which devices in the "View Device" area.

There is a "Users" section showing a list of users ids that have been accessing using that device. Hovering on each user id will show more details about the user, and clicking on it will bring you to the user manage view of that user.

4.2.7.4. Viewing Apps on Devices

Similar to capturing user access, app information is captured every time $fh.auth is called. You can view the list of apps in the "App" section in the "View Device" area. You can hover on each App entry to get an overview of the app. If you click on any of them will bring you to the manage view of that particular App.

4.3. ANALYTICS

Overview

Both Aggregated (reporting for all Projects in your domain) and individual Reports are available from the App Studio. Analytics users can access both the Aggregated and individual App reports from the Analytics sub Nav. Additionally, Aggregated Reports can be broken down by individual Project.

Requirements

The user must be a member of one or more teams with the following permissions:

- Domain — Project (View)
- Domain — Analytics (View)

Related Resources

- Cloud API - Statistics API

4.3.1. Aggregated Reporting

Aggregated reporting provides a high-level cumulative view of all the reports in your domain over the selected date range. For example, the median number of active devices, or total number of app installations.

4.3.1.1. Monthly Active Devices

This is the number of devices that have made a cloud request in a 31-day period within the selected date range. A device, identified with a unique device identifier, is considered active if it makes at least one cloud request (via **`$fh.act`** or **`$fh.cloud`**) in the given period.

4.3.1.1.1. Device Identifiers

Devices are identified by the Client Unique Identifier (CUID) field, which is added to each cloud request automatically by the RHMAP client SDKs.

The CUID is generated using different APIs on each platform:

- On iOS, the CUID value is generated using the Advertising Identifier.
- On Android, the CUID value is generated using the Android ID.
- For browsers that use the JavaScript SDK, the CUID is a generated random string saved in the browser (through any storage mechanism that is available). If the app data is removed, a new CUID is generated.

Note

Due to the limitations on each platform, it is not guaranteed that the value of CUID remain constant for a given device. Depending on the operating system of the device, the value could change after restarting the device, re-installation of apps, or removing app data on the device.

On iOS, different apps on the same device will get different CUID values. This means the same device will be counted as a new device for each app. This is a limitation of the iOS platform.

4.3.2. Per Project Reporting

There are 4 types of Reporting available for a Project: Device Installs, App Startups, Cloud Requests, and Active Devices. Depending on the configuration of your domain, the Cloud Requests and Active Devices reports may not be available to you.

4.3.2.1. Device Installs

Device Install reports show the number of App installs for a given date range. A Device Install is tracked the first time an App starts up on a new device and makes a call back to the cloud. Note that Device Installs differ from User Installs as tracked by App Stores in the following ways:

- App Stores count an install as a user account performing a download — regardless of how many devices the user installs the App on. This may vary for the different report types available from the individual App Stores.
- App stores typically report daily figures in PST timezone (in the case of Google Play and iTunes Connect), while we report in UTC.
- An App must be started up at least once with a valid Internet connection before we track the App install. We track the install as having happened at the time of the first startup, not the time of download.

4.3.2.2. App Startups

An App Startup is tracked every time an App is launched on a previously seen device with Internet connectivity. Note that App Startups are only tracked for Apps starting up 'from cold' - that is, if an App that is running in the background is brought to the foreground, this is not tracked as an 'App Startup'.

4.3.2.3. Cloud Requests

Cloud Request reports show the number of times an App Client makes a request (using **`$fh.cloud`**) to the App Cloud.

4.3.2.4. Active Devices

Active device reports show the median number of active devices for an app over the selected date range. A device, identified with a unique device identifier, is considered active if it makes at least one cloud request (via **`$fh.act`** or **`$fh.cloud`**) in a 24-hour period (from midnight to midnight in the GMT time zone).

4.3.3. Reporting Charts & Graphs

Each of the report types can be viewed in a number of ways, depending on which chart best shows the data you are looking for. In general, more info on individual chart points and areas is available by hovering over them. If a chart shows multiple items being compared, an item can be toggled on/off by clicking on the name of it in the legend (most charts support this). Chart data can often be exported in CSV, PNG or PDF format by clicking the export button associated with an individual chart.

The from and to date range can be changed for each chart, either by selecting a range using the calendars, or by choosing one of the quick date selections for example, "Last 30 Days", "Last 7 Days", "Previous 30 Days" etc.

The data for these charts is updated daily, so the latest data will be from yesterday.

4.3.3.1. Aggregated Reporting

Aggregated Reporting on the Reporting tab gives a high level view of all the App reports, for example, By Date, By Device, By Location. You can then easily drill down into specific reports by

clicking the "View Details" button associated with each analytics type.

4.3.3.2. Per-App Reporting

Per-App Reporting on the Reporting tab allows you to view reporting for an individual App by selecting an App from the list on the left - you can also search this list if you have many Apps. Once an App is selected, you will be presented with an overview of reporting for this particular App. If you're interested in a particular metric, you can drill down via the "View Details" button.

4.3.3.3. Individual App Analysis

This Individual App Screen provides an overview of reporting for a particular type of analytics (Startups, Installs etc.). Either clicking their Report titles, or switching views via the navigation pills will show individual Charts.

4.3.3.4. By Date Analysis

This chart type shows the increase/decrease of the value over time. The y-axis represents the value (for example, installs) and the x-axis shows the date. The values for each platform series (for example, Android, iPhone) are represented with separate lines.

4.3.3.5. By Platform Analysis

The aggregated values, per platform, can be viewed as a pie chart. The aggregation of values is for the selected date range.

4.3.3.6. By Location Analysis

The total values for a date range, based on User location, can be viewed as a Geo Chart. The value per country is shown, as well as the position of that value on a scale from min to max, when hovering over a country.

4.4. TEAMS AND COLLABORATION

4.4.1. Overview

The Teams and Collaboration section allows you to restrict access to specific features and entities within the Red Hat Mobile Application Platform (RHMAP).

The functionality in this section includes

- Creating new Teams.
- Adding Users to one or more Teams. When a User is added to a Team, they become a **`Member`** of that team. All of the Permissions assiged to the team will apply to the user.
- Assigning Permissions to teams to restrict access to specific features of the Platform.

Teams are split into two categories

1. **Default Teams** are Teams created by the Platform. The permissions and details (for example, name and description) of a **`Default Team`** are not editable. Only Users can be added/removed as **`Members`** of a **`Default Team`**.

2. **User Created Teams** are teams created by a User. The details and Permissions of a User Created Team are fully editable.

4.4.1.1. Team List

This screen presents a list of currently available teams. In this screen, it is possible to

- Display a list of Teams in grid or tabular format.
- Search for a Team by name.
- Filter the list of Teams by **`Customer`** and **`Domain`**.
- Create a new Team by selecting the **`Create Team`** button.

4.4.2. Create

A Team can be created by entering a **`name`** and **`description`** and selecting the **`Create Team`** button.

4.4.3. Dashboard

The Dashboard screen displays the details of the Team. In this screen, it is possible to

- View a list of Users currently assigned to the Team. These are refereed to as **`Members`** of a Team.
- Edit the **`Members`** list by clicking on the **`Edit Members`** link or the **`Members`** tab located on the left of the screen.
- View the current **`Permissions`** assigned to the Team.
- Edit the **`Permissions`** assigned to this team by selecting the **`Edit Permissions`** link or the **`Permissions`** tab located on the left of the screen.
- Edit the name and description of a Team.
- Delete the currently selected team by selecting the **`Delete this Team`** button

Note

The **`name`** or **`description`** of a **`Default Team`** cannot be edited.

Note

A **`Default Team`** cannot be deleted.

4.4.4. Members

The Members section allows you manage the Members of a Team. In this screen it is possible to

- View a list of current Members of the current Team.

- Add a User to the team by clicking into the text area and selecting a User. The selected User is atuomatically added as a Member of the Team.
- Remove a Member by selecting the "Remove" button beside each Member of a Team.
- Edit the name and description of a Team by selecting the **`Edit`** button.

Note

A User can be a **`Member`** of multiple teams.

4.4.5. Permissions

The Permissions section allows you to set the Permissions associated with the currently selected Team. All **`Members`** of the team will be restricted to the Permissions associated with this team.

4.4.5.1. Key Terms

Here are key terms used when interacting with Permissions.

- **Business Object**: A label describing an Entity in the Platform. (For example, Project).
- **Entity**: An instance of a Business Object. (For example, A project called **My First Project** is an instance of the Project Business Object.) **Permission**: An Access Level applied to a business object for example (read,write).
- **Team**: A collection of permissions that are applied to chosen business objects.
- **Hierarchy**: Describes where business objects sit in relation to each other and also describes functional areas of the ui

4.4.5.2. Business Objects

All Permissions are set on Business Object.

Business Objects relate to functional areas of the Platform organised into a hierarchy that describe its structure.

As an example, setting Permissions on the **`Administrator`** element under **`Reseller`** means Setting Permissions for Administrator features for this **`Reseller`**.

4.4.5.3. Access Levels

There are three Access Levels that apply to Business Objects.

- **No Access:** The **`Team`** cannot access this Business Object.
- **View:** The **`Team`** can view the Business Object, but cannot perform any action that would modify the Business Object.
- **View & Edit:** The **`Team`** can view the Business Object and perform actions that would modify the Business Object.

4.4.5.4. Inheritance

Business Objects are organised into a hierarchy. Business Objects can refer to a large functonal area of the Platform (for example, An Entire **Domain**). Sub-Business Objects refer to a more specific functional area of the Platform (for example, a **Project** in a **Domain**).

Using this structure, the **Permissions** of sub-Business Objects can be inherited from its parent. This is determined by the **Inherit Permissions** selector. If the **Inherit Permissions** selector is set to **ON**, the sub-Business Object will inherit the permissions of its parent. If **Inherit Permissions** is set to **OFF**, a specific **Access Level** must be set for the sub-Business Object.

Note

Business Objects can inherit **Permissions** from more than one level above as long as the parent level has **Inherit Permissions** set to **ON**.

The order of precedence of a **Permission** is determined by the **Access Level** set on the Business Object.

The following order is followed when determining what **Access Level** is set on the Business Object:

1. Read & Write.
2. View.
3. No Access.
4. Inherit Permissions.

4.4.5.5. Multiple Teams

A User can be a **Member** of multiple **Teams**. In this case, the order of precedence above is followed when determining what **Access Level** applies to the Business Object.

To illustrate this concept, consider the following scenario:

- **Team A** sets an **Access Level** of **Inherit Permissions** on **Business Object A**.
- **Team B** sets an **Acesss Level** of **View** on **Business Object A**.
- **User A** is a member of **Team A** and **Team B**.

In this scenario, **Business Object A** will have an **Access Level** of **View**.

4.4.5.6. Filterable Business Objects

Some Business Objects have multiple instances. For example, the **Project** Business Object relates to any **Project** available on a domain. As multiple projects can exist in a Domain it is necessary to be able to apply permissions to specific Projects.

A Filterable Business Object can be filtered in two ways

1. Setting **`Access All`** to **`ON`** will allow the Team to access all of the instances of the Business Object. For example, setting **`Access All`** to **`ON`** for the **`Project`** Business Object will allow the team to access all Projects in a Domain.
2. Setting **`Access All`** to **`OFF`** allows you to select which instances of the Busiess Object the Team has access to.

Any instance of a Business Object created by a User is visible to that user. However the permissions assigned to the Business Object will still apply.

As an example, consider the following scenario:

- **User A** is a **`Member`** of **Team A**.
- The **`Project`** Business Object **`Access Level`** is set to **`Read & Write`**.
- **User A** creates a new **`Project`** called **Project A**.
- For **Team A** the **`Project`** Business Object **`Access All`** is set to **`OFF`**.
- No instances of the **`Project`** Business Objects are selected.

In this scenario, **User A** will be able to see **Project A** only in the Projects section of the Studio.