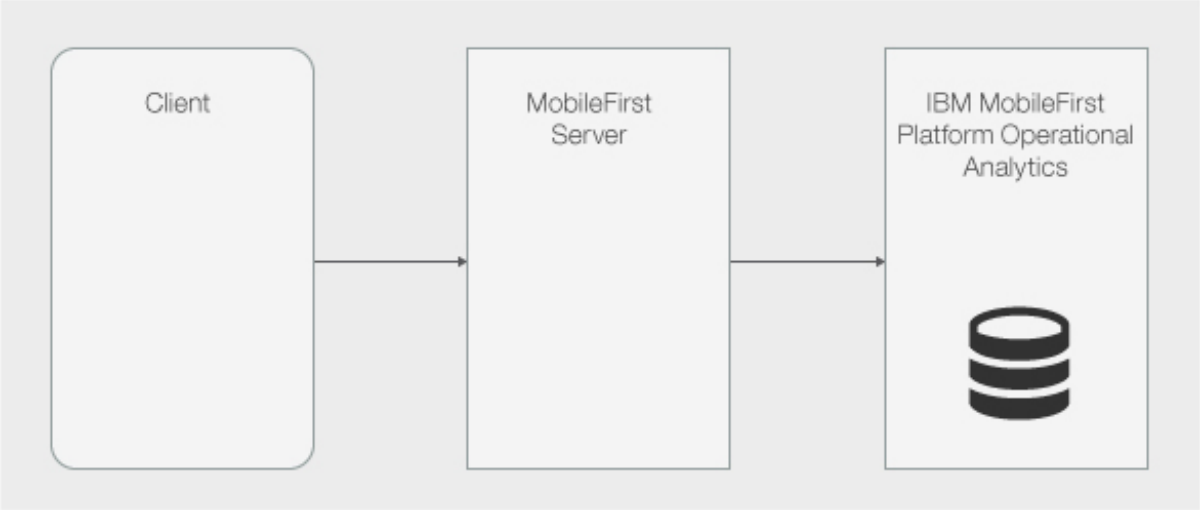
MobileFirst Foundation Operational Analytics

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/8.0/analytics/index.md) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

MobileFirst Foundation Operational Analytics collects data from app to server activities, client logs, client crashes, and server side logs from the MobileFirst Runtime Server and client devices. Collecting analytics from all these data points populates the Operational Analytics Charts with infrastructure and client side information.

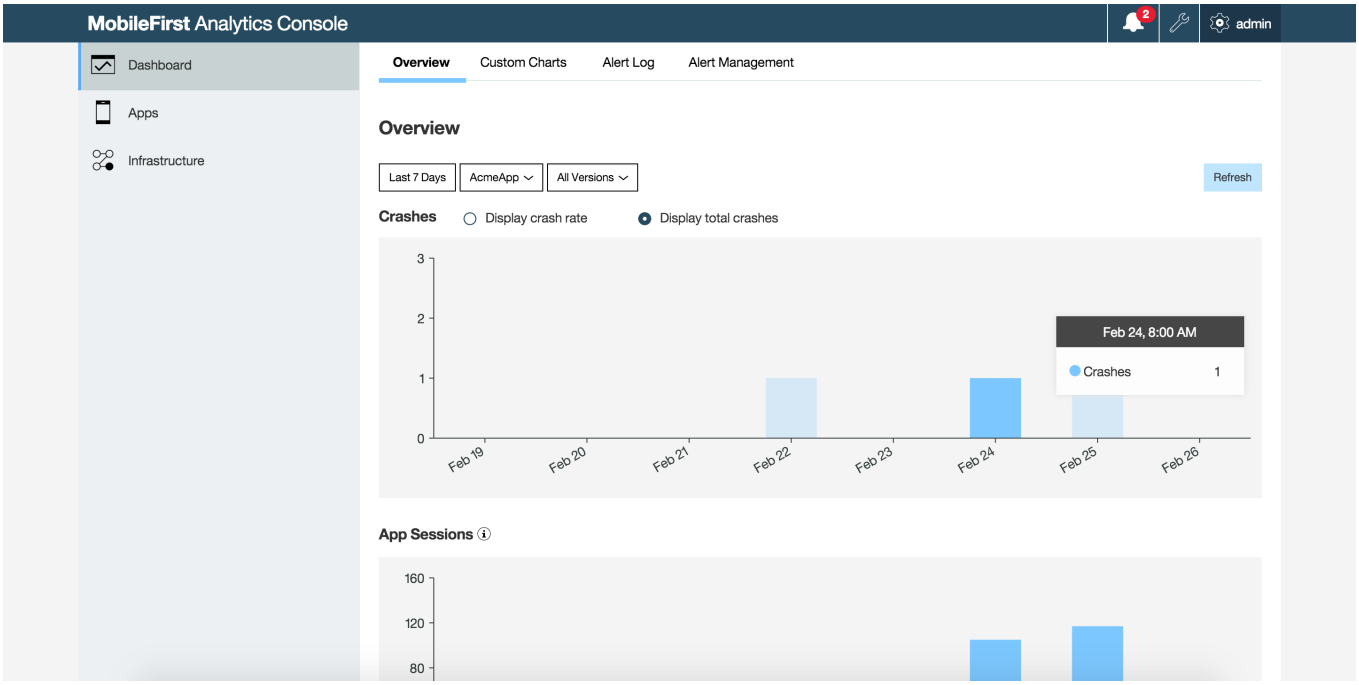


Jump to

MobileFirst Analytics Console

You can open the Analytics Console from the MobileFirst Operations Console, by clicking on "Analytics Console".

The default URL is for the MobileFirst Analytics Console is `http://localhost:9080/analytics/console`.
After navigating to the Analytics console you will see a dashboard like below (but with empty data).



From the Analytics Console you can then:

- Create custom charts
- Manage alerts
- Monitor App Crash
- Monitor Network Data

Custom Charts

Custom charts allow you to visualize the collected analytics data in your analytics data store in charts that are not avaliable in the out of the box MobileFirst Analytics Console. This visualization is a powerful way to analyze data important to your business needs.

Manage Alerts

You can set thresholds in alert definitions to better monitor your activities.

You can configure thresholds which, if exceeded, trigger alerts to notify the MobileFirst Analytics Console monitor. The triggered alerts can be visualized on the console, or the alerts can be handled by a custom webhook. A custom webhook allows control who and how someone is notified when an alert is triggered.

Monitor App Crash

App crashes are visualized on the MobileFirst Operational Analytics Console, where you can quickly view crashes and act on them accordingly. Crash logs are collected on the device by default. When crash logs are sent to the analytics server, they will automatically populate the crash charts.

Monitor Network Data

The MobileFirst Operational Analytics Console monitors network data when it is sent to the analytics server and allows the user to query this information in different ways.

> For more information about the data collected by MobileFirst Analytics, see the user documentation

Elasticsearch

Behind the scenes, running search queries and storing data for Operational Analytics is **Elasticsearch 1.5x**.
Elasticsearch is a real-time distributed search and analytics engine that provides the ability to explore data at speed and at a scale. Elasticsearch is used for full-text search, structured search.

Elasticsearch is used for storing all mobile and server data in JSON format in the Operational Analytics server in Elasticsearch instances.
The Elasticsearch instances are queried in real-time in order to populate the MobileFirst Operational Analytics Console.

MobileFirst Operational Analytics does not hide any Elasticsearch functionality. If knowledge how to fully utilize Elasticsearch, debug Elasticsearch, or optimize Elasticsearch instances is present, Operational Analytics does not prevent using it.

If you have interest in any Elasticsearch functionality besides what MobileFirst Operational Analytics offers out of the box, you can read more about it in the Elasticsearch documentation.

> Read more in the Elasticsearch documentation (https://www.elastic.co/guide/en/elasticsearch/reference/1.5/index.html)

Elasticsearch properties

Elasticsearch properties are available through JNDI variables or environment entries.
One of the more useful JNDI properties to get started viewing the Elasticsearch data is:

```
<jndiEntry jndiName="analytics/http.enabled" value="true"/>
```

This JNDI property will allow you to view your Operational Analytics raw data in JSON format and allow you to access your Elasticsearch instance through the port defined by Elasticsearch (default port 9500).

> **Note**: This is not secure and should not be enabled on a production environment.

Viewing data
You can view all your data by visiting the tenant's search REST endpoint.
Being able to access an Elasticsearch instance provides the ability to run custom queries and view more detailed information about the Elasticsearch cluster.

```
http://localhost:9500/*/_search
```

View cluster health

```
http://localhost:9500/_cluster/health
```

View information on current nodes

```
http://localhost:9500/_nodes
```

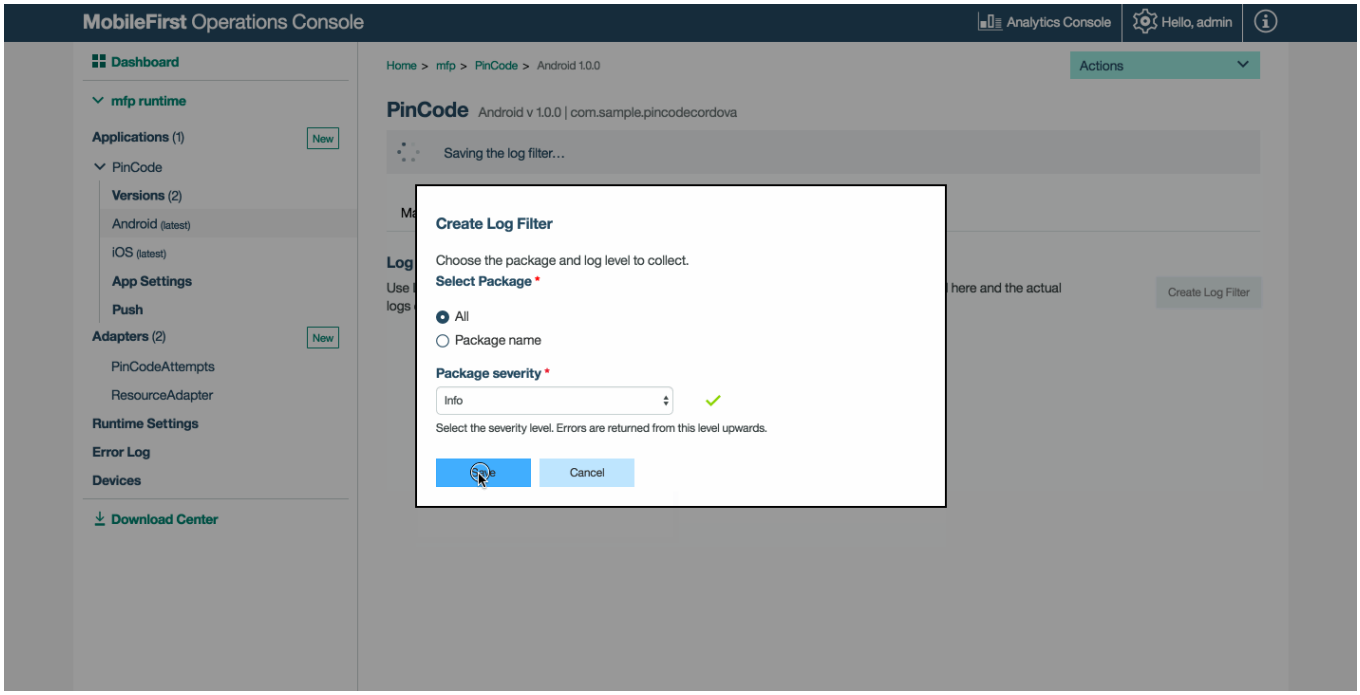
View the current mappings

```
http://localhost:9500/*/_mapping
```

> Elasticsearch exposes many more REST endpoints. To learn more, visit the Elasticsearch documentation.

Server Control of Client Log Capture

Administrators can control the MobileFirst client SDK log capture and levels from the **MobileFirst Operations Console → [your application] → Log Filters**.
Through `Log Filters` you are able to create a filter level that you can log at.



In order to use the server configuration the client has to use the `updateConfigFromServer` method in the `Logger` API.

Android

```
Logger.updateConfigFromServer();
```

iOS

```
[OCLogger updateConfigFromServer];
```

Cordova

```
WL.Logger.updateConfigFromServer();
```

The `Logger` configuration values returned from the server will take precidence over any value set on the client side. When the Client Log Profile is removed and the client tries to retrieve the Client Log Profile, the client will receive an empty payload. If an empcdty payload is received then the `Logger` configuration will default to what was originally configured on the client.

Forwarding Server Logs

The MobileFirst Foundation Operations Console also gives the server administrator the ability to persist logs and send those logs to the MobileFirst Analytics Console.

To forward server logs navigate to the Runtime's **Settings** screen and provide the used logger package

under **Additional Packages**.
The collected logs can then be viewed in the Analytics console. This is useful for a user when they want to take advantage of triaging adapter logs in the Analytics console without having to collect all server logs.

Tutorials to follow next

- Analytics API (analytics-api)
- Analytics REST API (analytics-rest-api)

Related Blogposts

- More on Instrumenting Custom Analytics (file:////home/travis/build/MFPSamples/DevCenter/_site/blog/2016/01/22/howto-custom-in-app-behavior-analytics/)
- More on Instrumenting Webhooks (file:////home/travis/build/MFPSamples/DevCenter/_site/blog/2015/10/19/using-mfp-adapters-endpoint-analytics-alerts-webhooks/)