Analytics Workflows

Overview

Leverage MobileFirst Analytics to best serve your business needs. Once your goals are identified collect the appropriate data using the analytics client SDK and build reports using the MobileFirst Analytics Console. The following typical scenarios demonstrate methods of collecting and reporting analytics data.

Jump to

- App Usage Analytics
- Crash Capture

App usage Analytics

Initializing your client app to capture app usage

App usage measures the number of times a specific app is brought to the foreground, and then sent to the background. To capture app usage in your mobile app, the MobileFirst Analytics client SDK must be configured to listen for the app lifecycle events.

You can use the MobileFirst Analytics API to capture app usage. Make sure you have first created a relevant device listener. Then send the data to the server.

iOS

Add the following code in your Application Delegate `application:didFinishLaunchingWithOptions` method in the **AppDelegate.m**/**AppDeligate.swift** file.

Objective-C

```
WLAnalytics *analytics = [WLAnalytics sharedInstance];
[analytics addDeviceEventListener:LIFECYCLE];
```

To send the analytics:

```
[[WLAnalytics sharedInstance] send];
```

Swift

```
WLAnalytics.sharedInstance().addDeviceEventListener(LIFECYCLE);
```

To send the analytics:

```
WLAnalytics.sharedInstance().send;
```

Android

Add the following code in your Application subclass `onCreate` method.

```
WLAnalytics.init(this);
WLAnalytics.addDeviceEventListener(DeviceEvent.LIFECYCLE);
```

To send the analytic data:

```
WLAnalytics.send();
```

Cordova

For Cordova apps, the listener must be created in the native platform code, similar to the iOS and Android apps. Send the data to the server:

```
WL.Analytics.send();
```

Web apps

For Web apps, no listeners are required. Analytics can be enabled and disabled through the `WLlogger` class.

```
ibmmfpfanalytics.logger.config({analyticsCapture: true});
ibmmfpfanalytics.send();
```

Default Usage and Devices chart

In the **Usage and Devices** page of the Apps section in the IBM MobileFirst Analytics Console, a number of default charts are provided to help you manage your app usage.

Total Devices

The **Total Devices** chart shows the number of total devices.

Total App Sessions

The **Total App Sessions** chart shows the number of total app sessions. An app session is recorded when an app is brought to the foreground of a device.

Active Users

The **Active Users** chart shows an interactive multi-line graph of the following data:

- Active Users - unique users for the displayed time frame.
- New Users - new users for the displayed time frame.

The default displayed time frame is one day with a data point for each hour. If you change the displayed time frame to greater than one day, the data points reflect each day. You can click the corresponding key in the legend to toggle whether to display the line. By default, all keys are displayed, and you cannot toggle all keys to not display any lines.

To see the most accurate data in the line graph, you must instrument your app code to provide the `userID` by calling the `setUserContext` API. If you want to provide anonymity for the `userID` values, you must hash the value first. If the `userID` is not provided, the ID of the device is used by default. If one device is used by multiple users and the `userID` is not provided, the line graph does not reflect accurate data because the ID of the device counts as one user.

App Sessions

The **App Sessions** chart shows a bar graph of app sessions over time.

App Usage

The **App Usage** chart shows a pie chart of the percentage of app sessions for each app.

New Devices

The **New Devices** chart shows a bar graph of new devices over time.

Model Usage

The **Model Usage** chart shows a pie chart of the percentage of app sessions for each device model.

Operating System Usage

The **Operating System Usage** chart shows a pie chart of the percentage of app sessions for each device operating system.

Creating a custom chart for average session duration

The duration of an app session is a valuable metric to visualize. With any app, you want to know the amount of time that users are spending on a particular session.

1. In the MobileFirst Analytics Console, click **Create Chart** in the Custom Charts page of the Dashboard section.
2. Give your chart a title.
3. Select **App Session** for **Event Type**.
4. Select **Bar Graph** for **Chart Type**.
5. Click **Next**.
6. Select **Timeline** for **X-Axis**.
7. Select **Average** for **Y-Axis**.
8. Select **Duration** for **Property**.
9. Click **Save**.

Crash capture

MobileFirst Analytics includes data and reports about application crashes. This data is collected automatically along with other lifecycle event data. The crash data is collected by the client and is sent to the server once the application is again up and running.

An app crash is recorded when an unhandled exception occurs and causes the program to be in an unrecoverable state. Before the app closes, the MobileFirst Analytics SDK logs a crash event. This data is sent to the server with the next logger send call.

Initializing your app to capture crash data

To ensure that crash data is collected and included in the MobileFirst Analytics Console reports, make sure the crash data is sent to the server.

Ensure that you are collecting app lifecycle events as described in Initializing your client app to capture app usage.

The client logs must be sent once the app is running again, in order to get the stacktrace that is associated with the crash. Using a timer ensures that the logs are sent periodically.

iOS

Objective-C

```
- (void)sendMFPAnalyticData {
  [OCLogger send];
  [[WLAnalytics sharedInstance] send];
}

// then elsewhere in the same implementation file:

[NSTimer scheduledTimerWithTimeInterval:60
  target:self
  selector:@selector(sendMFPAnalyticData)
  userInfo:nil
  repeats:YES]
```

Swift

```
overridefuncviewDidLoad() {
    super.viewDidLoad()
    WLAnalytics.sharedInstance();
    lettimer = NSTimer.scheduledTimerWithTimeInterval(10, target: self, selector: #selector(sendMFPA
nalyticData), userInfo: nil, repeats: true);
    timer.fire();
    // Do any additional setup after loading the view, typically from a nib.
  }

  funcsendMFPAnalyticData() {
    OCLogger.send()
    WLAnalytics.sharedInstance().send()
  }
```

Android

```
Timer timer = new Timer();
timer.schedule(new TimerTask() {
  @Override
  public void run() {
    Logger.send();
    WLAnalytics.send();
  }
}, 0, 60000);
```

Cordova

```
setInterval(function() {
  WL.Logger.send();
  WL.Analytics.send();
}, 60000)
```

Web

```
setInterval(function() {
  ibmmfpfanalytics.logger.send();
}, 60000);
```

App crash monitoring

After a crash, when the app is restarted, the crash logs are sent to the Analytics server. You can quickly see information about your app crashes in the Dashboard section of the MobileFirst Analytics Console. In the **Overview** page of the **Dashboard** section, the Crashes bar graph shows a histogram of crashes over time.

The data can be shown in two ways:

- **Display crash rate**: crash rate over time
- **Display total crashes**: total crashes over time

> **Note:** The Crashes chart queries against the `MfpAppSession` documents. You must instrument your app to collect app uses and crashes for data to appear in the charts. If `MfpAppSession` data is not collected, then `MfpAppLog` documents are queried. In this case, the chart can count the number of crashes, but cannot compute a crash rate because the number of app uses is unknown, which results in the following limitation:
>
> - The **Crashes** bar graph displays no data when **Display Crash Rate** is selected.

Default charts for crashes

In the **Crashes** page of the **Apps** section in the IBM MobileFirst Analytics Console, a number of default charts are provided to help you manage your app crashes.

The **Crash Overview** chart shows a table of an overview of crashes.
The **Crashes** bar graph shows a histogram of crashes over time. You can display the data by crash rate or total crashes. The Crashes bar graph is also in the Crashes page of the Applications section.

The **Crash Summary** chart shows a sortable table of a summary of crashes. You can expand the individual crashes by clicking the + icon to view a **Crash Details** table that includes more details about the crashes. In the Crash Details table, you can click the **>** icon to view more details about the specific crash instance.

App crash troubleshooting

You can view the **Crashes** page in the **Applications** section of the MobileFirst Analytics Console to better administer your apps.

The **Crash Overview** table shows the following data columns:

- **App:** app name
- **Crashes:** total number of crashes for that app
- **Total Uses:** total number of times a user opens and closes that app
- **Crash Rate:** percentage of crashes per use

The **Crashes** bar graph is the same chart that is displayed in the **Overview** page of the **Dashboard** section.

> **Note:** Both charts query against the `MfpAppSession` documents. You must instrument your app to collect app uses and crashes for data to appear in the charts. If `MfpAppSession` data is not collected, then `MfpAppLog` documents are queried. In this case, the charts can count the number of crashes, but cannot compute a crash rate because the number of app uses is unknown, which results in the following limitations:
>
> - The Crash Overview table has empty columns for Total Uses and Crash Rate.
> - The Crashes bar graph displays no data when Display Crash Rate is selected.

The **Crash Summary** table is sortable and includes the following data columns:

- Crashes
- Devices
- Last Crash
- App
- OS
- Message

You can click on the **+** icon next to any entry to display the **Crash Details** table, which includes the following columns:

- Time Crashed
- App Version
- OS Version
- Device Model
- Device ID
- Download: link to download the logs that led up to the crash

You can expand any entry in the **Crash Details** table to get more details, including a stacktrace.

> **Note:** The data for the **Crash Summary** table is populated by querying the fatal level client logs. If your app does not collect fatal client logs, no data is available.

Last modified on