Using Analytics API in client applications

Overview

To populate your custom charts you can use the Anlaytics API to send customized data. Custom data is any key/value pair that you would like to collect that is not an out of the box feature, like button presses.

When collecting custom analytics data, the app writes to the devices file system and is not sent until the `send` API is called. After the `send` API is called the app deletes the data from the file system and begins collecting again.

Before you being in the native app import the Analytics SDK.

Android

```
import com.worklight.common.WLAnalytics;
```

iOS

```
import "WLAnalytics.h"
```

Jump to:

Initializing Analytics

Before you can start collecting the out of the box data that Operational Analytics provides you first need to initialize Analytics.

In cordova this is done through your native application. So the code snippets below are going to show you how to initialize analytics in iOS and Android.

Android:

```
WLAnalytics.init(this.getApplication());
```

iOS:

```

```

After initializing analytics the app will start collecting user information, like app sessions.

After sending data to the MobileFirst server you will start to see charts filled out like the one below:

Recording App Sessions

After initializing the Analtyics SDK app sessions will start to be recorded on the users device. A session in MobileFirst Operational Anlaytics is recorded when the app is moved to the foreground, then to the background thTt sequence of events is one full session.

As soon as the device is set up to record sessions and you send your data, you will see a chart like the one below.

!(sessions-chart)[./analytics-app-sessions.png]

You can enable or disable the collecting of app sessions with the API below:

Android:

```
//DeviceEvent.LIFECYCLE records app sessions
WLAnalytics.addDeviceEventListener(DeviceEvent.LIFECYCLE);
WLAnalytics.removeDeviceEventListener(DeviceEvent.LIFECYCLE);
```

Client Network Activities

Collection on adapters and network occur in two different locations on the client and on the server.

The client is going to collect the information when you start collecting on the device event `Network`. Collecting on the Client is going to collect information like roundtrip time and payload size.

The server is going to collect more backend information like server processing time, adapter usage, procedures, etc.

Since the client and the server are each collecting their own information this means that all the charts will not display data until the client is configured to do so. To configure your client you need to start collecting on the device event `Analytics`.

To enable or disable network events on the client use the API below:

Android:

```
//DeviceEvent.Network records client information about adapters like 'Average Procedure Response Size'
WLAnalytics.addDeviceEventListener(DeviceEvent.NETWORK);
WLAnalytics.removeDeviceEventListener(DeviceEvent.NETWORK);
```

Custom Events

JavaScript API

JavaScript API is used in Cordova applications.

Creating custom events in Cordova is simply just calling:

```
WL.Analytics.log({"key" : 'value'});
WL.Analytics.send();
```

Android

After setting the first two configurations you can start to log data like in the example below.

```
JSONObject json = new JSONObject();
try {
    json.put("key", "value");
} catch (JSONException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

WLAnalytics.log("Message", json);
```

Be sure to send your data with:

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

Objective-C API

Objective-C API is used in iOS applications.

To collect custom analytics in iOS you need to import WLAnalytics.

```
#import "WLAnalytics.h";
```

After importing WLAnalytics you can now use the API to collect custom data like below:

```
NSDictionary *inventory = @{
    @"property" : @"value",
};

[[WLAnalytics sharedInstance] log:@"Custom event" withMetadata:inventory];
[[WLAnalytics sharedInstance] send];
```