Client-side log collection

Overview

Logging is the instrumentation of source code that uses API calls to record messages in order to facilitate diagnostics and debugging. MobileFirst Platform Foundation provides a set of `Logger` API methods for this purpose.

Availability

The `Logger` API methods can be used with iOS, Android and Cordova applications.

Logger API

The `Logger` API is similar to commonly used logger APIs, such as `console.log` (JavaScript), `java.util.logging` (Java) and `NSLog` (Objective-C).

The `Logger` API has the additional capability of persistently capturing logged data for sending to the MobileFirst Server to be used for analytics gathering and developer inspection. Use the `Logger` APIs to report log data at appropriate levels so that developers who inspect logs can triage and fix problems without having to reproduce problems in their labs.

Logging libraries typically have verbosity controls that are frequently called **levels**. From least to most verbose: ERROR, WARN, INFO, LOG and DEBUG.

> **Note:** Using FATAL will result in collecting an app crash. To not skew your app crash data we recommend not using this keyword.

Log Capture

Client log capture can be controlled via several ways:

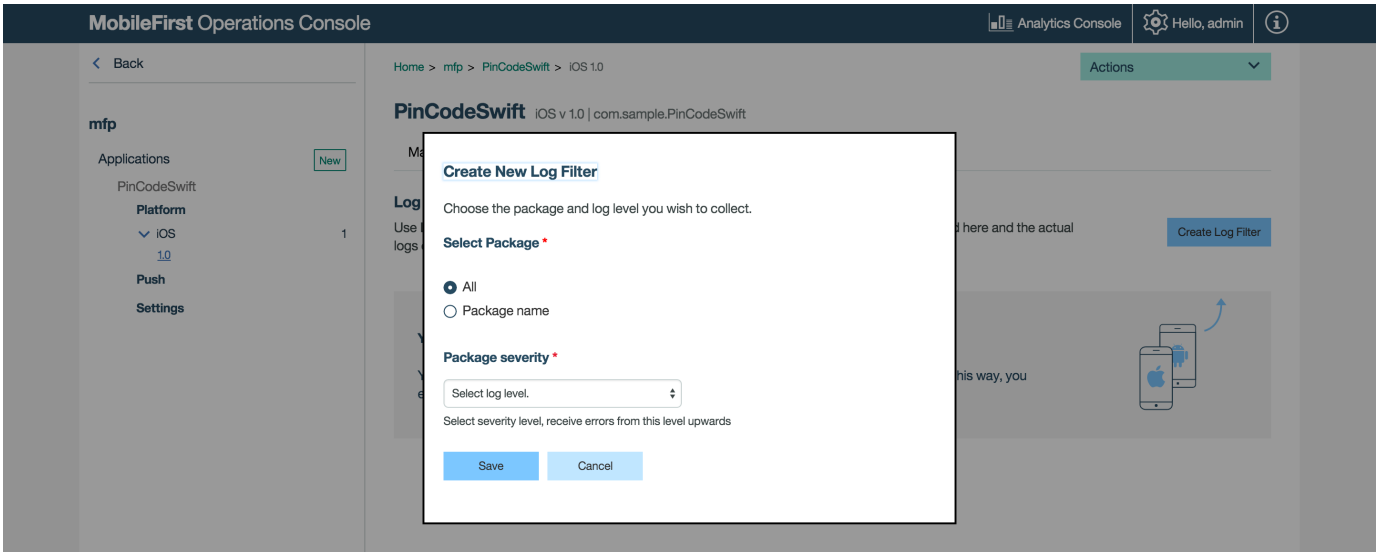
- By the client itself,
- Or by the MobileFirst Runtime Server.

Logging from client applications:

- Logging in Cordova applications (cordova/)
- Logging in iOS applications (ios/)
- Logging in Android applications (android/)

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. When the client receives and empty payload the `Logger` configuration will default to what was originally configured on the client.

Crash capture

The MobileFirst client SDK, on Android and iOS applications, captures a stack trace upon application crash and logs it at FATAL level. This type of crash is a true crash where the UI disappears from the user's view.

The MobileFirst client SDK, in Cordova applications, captures JavaScript global errors and if possible a JavaScript call stack, and logs it at FATAL level. This type of crash is not a crash event, and might or might not have any adverse consequences to the user experience at run time.

Crashes, uncaught exceptions, and global errors are caught and logged automatically.

For more information

> For more information about logging and log capture, see the user documentation.