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 Foundation provides a set of logging API methods for this purpose.

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

Availability

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

Logging levels

Logging libraries typically have verbosity controls that are frequently called **levels**.
The logging levels from the most verbose to the least are as follows:

- TRACE - used for method entry and exit points
- DEBUG - used for method result output
- LOG - used for class instantiation
- INFO - used for reporting initialization
- WARN - used to log deprecated usage warnings
- ERROR - used for unexpected exceptions
- FATAL - used for unrecoverable crashes or hangs

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

The client SDKs are configured at the FATAL verbosity by default, which means little or no raw debug logs are output or captured. You can adjust the verbosity programmatically, or adjust it, by setting a configuration profile on the MobileFirst Operations Console, which must be retrieved explicitly by your app.

Logging from client applications:

- Logging in JavaScript (Cordova, Web) applications (javascript/)
- Logging in iOS applications (ios/)
- Logging in Android applications (android/)

Adjusting log verbosity

Once logging level is set, either by setting the client or retrieving the server profile, the client filters the logging messages it sends. If a message below the threshold is explicitly sent, the client ignores it.

For example, to set the verbosity level to DEBUG:

iOS

Objective-C

```
[OCLogger setLevel:OCLogger_DEBUG];
```

Swift

```
OCLogger.setLevel(OCLogger_DEBUG);
```

Android

```
Logger.setLevel(Logger.LEVEL.DEBUG);
```

JavaScript (Cordova)

```
WL.Logger.config({ level: 'DEBUG' });
```

JavaScript (Web)

For the web SDK the default trace level cannot be changed from 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. 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 once the app is running again.

Viewing the logs

After the logs are collected and sent to the server, view them in the MobileFirst Analytics Console. Choose the **Apps** panel from the navigation bar and click the **Client Log Search** tab.