

# Client-side log collection

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/8.0/using-the-mfpf-sdk/client-side-log-collection/index.md>) | report issue (<https://github.ibm.com/MFPSamples/DevCenter/issues/new>)

## 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 `Logger` 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

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

## Logging levels

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

## 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.

## For more information

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