Remote controlled client-side log collection

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/6.3/advanced-client-side-development/remote-controlled-client-side-log-collection/index.html) | 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. Example: `java.util.logging` in Java.

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

- Developers can filter by level in their log viewer
- Capture tools can filter by level in their configurations

Topics Covered:

- Log Capture
- Server Preparation
- Admin Control
- Crash Capture
- API calls for specific environments

Log Capture

Log capture is the ability to persistently record messages that are passed to the logging API.
Capture is on by default but can be turned off.

Logging defaults to DEBUG level in development, and FATAL in production. Control levels, and therefore verbosity, with API calls specific to your environment.

Server preparation

Persistently captured log output in browsers and devices in production is useless unless that captured data can be sent for diagnostic inspection.

The client sends accumulated captured log messages automatically:

- At Worklight client connect success
- At Worklight adapter call success

Automatic behavior can be turned on or off with API calls specific to your environment, or by calling the `send` method explicitly in your application.

Server preparation (in production)

How to receive and process received logs at the server?

The Analytics Platform, and therefore the Analytics Dashboard link in the MobileFirst Platform Console, is installed and available by default in the embedded Liberty server (in MobileFirst Platform Studio development environment).

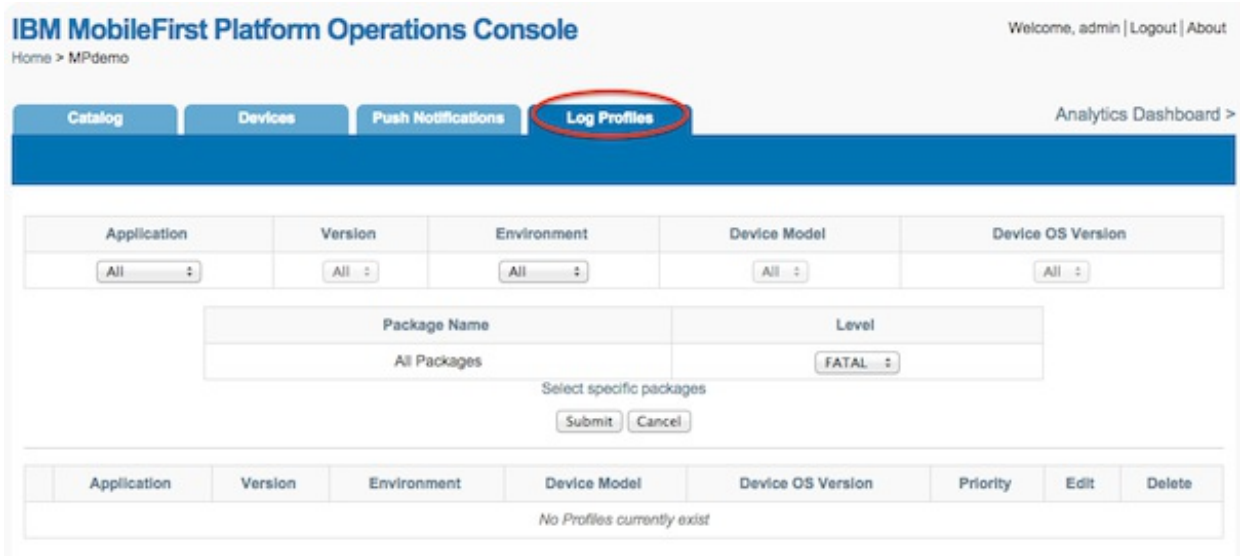
How to receive and process received logs at the server (in MobileFirst Platform production environment)?

Two options:

- Install the Analytics Platform WAR file, and/or
- Develop and deploy an adapter that is named `WLClientLogReceiver` with a function (procedure) that is named `log`

> **Note:** Installation and configuration of Analytics Platform and development of adapters is beyond the scope of this getting started guide. For more information, see the related getting started modules or the IBM MobileFirst Platform user documentation.

Admin control of client log capture



Use the MobileFirst Platform Operations Console to configure log capture preferences for applications in production. Administrators can control the MobileFirst Platform client SDK log capture and levels from the MobileFirst Platform Operations Console.

Automatic behavior can be turned on or off with API calls specific to your environment, or by calling the `updateConfigFromServer` method explicitly in your application.

Crash Capture

MobileFirst Platform client SDK, on Android and iOS, 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.

MobileFirst Platform client SDK, in JavaScript, 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.

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

For more information

For more information about Logging and Log Capture, see the IBM MobileFirst Platform Foundation user documentation.