Remote controlled 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. Example: `java.util.logging` in Java.

Logging libraries typically have verbosity controls, that are frequently called *levels*. 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.

The following topics are covered:

- Log capture
- Server preparation (in production)
- Admin control for client logs
- Crash capture
- API calls for specific environments
- For more information

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 to FATAL in production. You can control levels, and therefore verbosity, with API calls that are specific to your environment.

Server preparation (in production)

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:

- When a MobileFirst client connects successfully.
- When a MobileFirst adapter is called successfully.

You can turn the automatic behavior on or off with API calls that are specific to your environment, or by calling the `send` method explicitly in your application.

How to receive and process logs at the server?

There are two ways to receive client logs on the Worklight Server:

1. Install the Analytics Platform
2. Develop and deploy an adapter that is named WLClientLogReceiver with a function (procedure) that is named log

The Analytics Platform is available by default in the embedded Liberty server in the MobileFirst Studio development environment. A link to the Analytics Dashboard is provided by the the MobileFirst Platform Console. When the Analytics Platform has been properly installed and configured, client logs that are sent to the MobileFirst Platform Server will automatically be forwarded to the Analytics Platform.

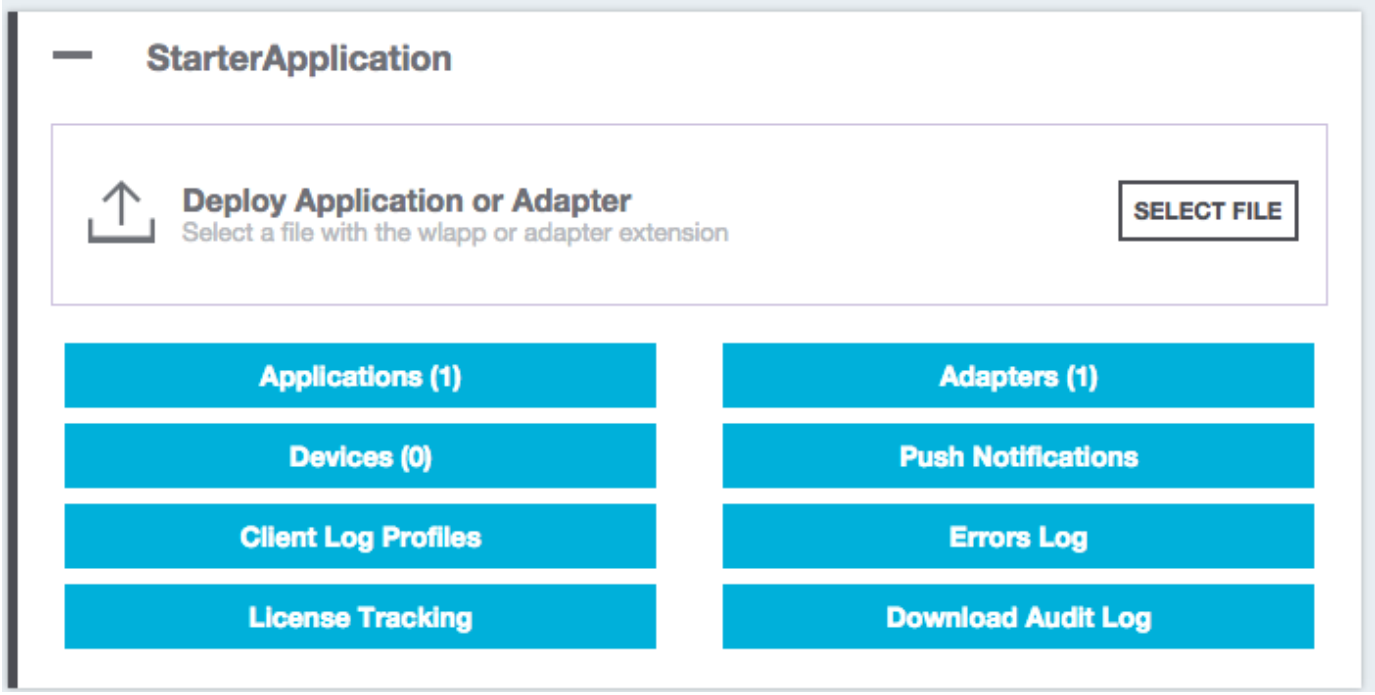
> For more information on installing Analytics in a production environment, review the "Installing the IBM MobileFirst Platform Operational Analytics" topic in the user documentation.

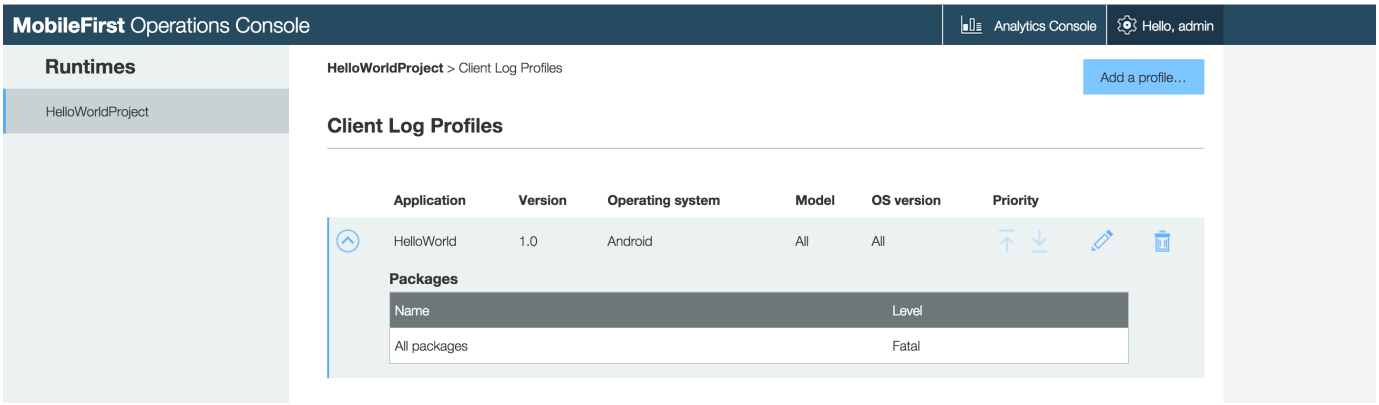
If you do not wish to use the Analytics Platform, you can still collect and process client logs by creating an adapter. By default, the MobileFirst Platform is configured to forward client logs to an adapter named "WLClientLogReceiver" that exposes a method called "log".

If the Analytics Platform and the adapter have both been configured, the MobileFirst Platform will send logs to both endpoints. Using both may be useful if you'd like to take advantage of the Analytics Platform's log searching capabilities, but also wish to do some custom operations on client logs using your own adapter.

Note: Installation and configuration of Analytics Platform and development of adapters is beyond the scope of this Getting Started tutorial. For more information, see the "Installing the MobileFirst Platform Operational Analytics" topic in the user documentation, and the Adapters overview tutorial (../../server-side-development/adapter-framework-overview/).

Admin control of client log capture





To configure log capture preferences for applications in production, use the MobileFirst Operations Console. Administrators can control the MobileFirst client SDK log capture and levels from the MobileFirst Operations Console.

Crash capture

The MobileFirst 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's view.

The MobileFirst 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 user documentation.