Reading and Writing Logs

The Android logging system provides a mechanism for collecting and viewing system debug output. Logcat dumps a log of system messages, which include things such as stack traces when the emulator throws an error and messages that you have written from your application by using the Log (/reference/android/util/Log.html) class. You can run LogCat through ADB or from DDMS, which allows you to read the messages in real time.

IN THIS DOCUMENT

The Log class

Log (/reference/android/util/Log.html) is a logging class that you can utilize in your code to print out messages to the LogCat. Common logging methods include:

- v(String, String) (verbose)
- d(String, String) (debug)
- i(String, String) (information)
- w(String, String) (warning)
- e(String, String) (error)

For example:

```
Log.i("MyActivity", "MyClass.getView() — get item number " + position);
```

The LogCat will then output something like:

```
I/MyActivity( 1557): MyClass.getView() — get item number 1
```

Using LogCat

You can use LogCat from within DDMS or call it on an ADB shell. For more information on how to use LogCat within DDMS, see Using DDMS (/tools/debugging/ddms.html#logcat). To run LogCat, through the ADB shell, the general usage is:

```
[adb] logcat [<option>] ... [<filter-spec>] ...
```

You can use the `logcat` command from your development computer or from a remote adb shell in an emulator/device instance. To view log output in your development computer, you use

```
$ adb logcat
```

and from a remote adb shell you use

```
# logcat
```

The following table describes the `logcat` command line options:

`-c`	Clears (flushes) the entire log and exits.
`-d`	Dumps the log to the screen and exits.
`-f <filename>`	Writes log message output to `<filename>`. The default is `stdout`.

`-g`	Prints the size of the specified log buffer and exits.
`-n <count>`	Sets the maximum number of rotated logs to `<count>`. The default value is 4. Requires the `-r` option.
`-r <kbytes>`	Rotates the log file every `<kbytes>` of output. The default value is 16. Requires the `-f` option.
`-s`	Sets the default filter spec to silent.
`-v <format>`	Sets the output format for log messages. The default is `brief` format. For a list of supported formats, see Controlling Log Output Format.

Filtering Log Output

Every Android log message has a *tag* and a *priority* associated with it.

- The tag of a log message is a short string indicating the system component from which the message originates (for example, "View" for the view system).
- The priority is one of the following character values, ordered from lowest to highest priority:
  - `V` — Verbose (lowest priority)
  - `D` — Debug
  - `I` — Info
  - `W` — Warning
  - `E` — Error
  - `F` — Fatal
  - `S` — Silent (highest priority, on which nothing is ever printed)

You can obtain a list of tags used in the system, together with priorities, by running LogCat and observing the first two columns of each message, given as `<priority>/<tag>`.

Here's an example of logcat output that shows that the message relates to priority level "I" and tag "ActivityManager":

```
I/ActivityManager(  585): Starting activity: Intent { action=android.intent.action..
```

To reduce the log output to a manageable level, you can restrict log output using *filter expressions*. Filter expressions let you indicate to the system the tags-priority combinations that you are interested in — the system suppresses other messages for the specified tags.

A filter expression follows this format `tag:priority ...`, where `tag` indicates the tag of interest and `priority` indicates the *minimum* level of priority to report for that tag. Messages for that tag at or above the specified priority are written to the log. You can supply any number of `tag:priority` specifications in a single filter expression. The series of specifications is whitespace-delimited.

Here's an example of a filter expression that suppresses all log messages except those with the tag "ActivityManager", at priority "Info" or above, and all log messages with tag "MyApp", with priority "Debug" or above:

```
adb logcat ActivityManager:I MyApp:D *:S
```

The final element in the above expression, `*:S`, sets the priority level for all tags to "silent", thus ensuring only log messages with "View" and "MyApp" are displayed. Using `*:S` is an excellent way to ensure that log output is restricted to the filters that you have explicitly specified — it lets your filters serve as a "whitelist" for log output.

The following filter expression displays all log messages with priority level "warning" and higher, on all tags:

```
adb logcat *:W
```

If you're running LogCat from your development computer (versus running it on a remote adb shell), you can also set a default filter expression by exporting a value for the environment variable `ANDROID_LOG_TAGS`:

```
export ANDROID_LOG_TAGS="ActivityManager:I MyApp:D *:S"
```

Note that `ANDROID_LOG_TAGS` filter is not exported to the emulator/device instance, if you are running LogCat from a remote shell or using `adb shell logcat`.

Controlling Log Output Format

Log messages contain a number of metadata fields, in addition to the tag and priority. You can modify the output format for messages so that they display a specific metadata field. To do so, you use the `-v` option and specify one of the supported output formats listed below.

- `brief` — Display priority/tag and PID of the process issuing the message (the default format).
- `process` — Display PID only.
- `tag` — Display the priority/tag only.
- `raw` — Display the raw log message, with no other metadata fields.
- `time` — Display the date, invocation time, priority/tag, and PID of the process issuing the message.
- `threadtime` — Display the date, invocation time, priority, tag, and the PID and TID of the thread issuing the message.
- `long` — Display all metadata fields and separate messages with blank lines.

When starting LogCat, you can specify the output format you want by using the `-v` option:

```
[adb] logcat [-v <format>]
```

Here's an example that shows how to generate messages in `thread` output format:

```
adb logcat -v thread
```

Note that you can only specify one output format with the `-v` option.

Viewing Alternative Log Buffers

The Android logging system keeps multiple circular buffers for log messages, and not all of the log messages are sent to the default circular buffer. To see additional log messages, you can run the `logcat` command with the `-b` option, to request viewing of an alternate circular buffer. You can view any of these alternate buffers:

- `radio` — View the buffer that contains radio/telephony related messages.
- `events` — View the buffer containing events-related messages.
- `main` — View the main log buffer (default)

The usage of the `-b` option is:

```
[adb] logcat [-b <buffer>]
```

Here's an example of how to view a log buffer containing radio and telephony messages:

```
adb logcat -b radio
```

Viewing stdout and stderr

By default, the Android system sends `stdout` and `stderr` (`System.out` and `System.err`) output to `/dev/null`. In processes that run the Dalvik VM, you can have the system write a copy of the output to the log file. In this case, the system writes the messages to the log using the log tags `stdout` and `stderr`, both with priority I.

To route the output in this way, you stop a running emulator/device instance and then use the shell command `setprop` to enable the redirection of output. Here's how you do it:

```
$ adb shell stop
$ adb shell setprop log.redirect-stdio true
$ adb shell start
```

The system retains this setting until you terminate the emulator/device instance. To use the setting as a default on the emulator/device instance, you can add an entry to `/data/local.prop` on the device.

Debugging Web Apps

If you're developing a web application for Android, you can debug your JavaScript using the console JavaScript APIs, which output messages to LogCat. For more information, see Debugging Web Apps (/guide/webapps/debugging.html).