Structured logging

This document discusses the concept of structured logging and the methods for adding structure to log entry payload fields. When the log payload is formatted as a JSON object and that object is stored in the `jsonPayload` field, the log entry is called a *structured log*. For these logs, you can construct queries that search specific JSON paths and you can index specific fields in the log payload. In contrast, when the log payload is formatted as a string and stored in the `textPayload` field, the log entry is *unstructured*. You can search the text field, but you can't index its content.

To create structured log entries, do any of the following:

- Call the `entries.write` (/logging/docs/reference/v2/rest/v2/entries/write) API method and supply a fully formatted `LogEntry` (/logging/docs/reference/v2/rest/v2/LogEntry).
- Use the `gcloud logging write` command.
- Use a Cloud Logging client library which writes structured logs.
- Use the BindPlane service.
- Use an agent to write logs:

★ **Note:** We recommend that you use an agent to write logs, when that option is available. To write structured logs, configure your application to write serialized JSON objects. For a list of logging frameworks, see Recommended logging frameworks (/stackdriver/docs/instrumentation/choose-approach#logging-frameworks).

  - Some Google Cloud services contain an integrated logging agent that sends the data written to `stdout` or `stderr` as logs to Cloud Logging. You can use this approach for Google Cloud services such as Google Kubernetes Engine, App Engine flexible environment, and Cloud Run functions.
  - For Compute Engine virtual machines (VMs), you can install and configure the Ops Agent or the legacy Logging agent, and then use the installed agent to send logs to Cloud Logging.

For more information about these approaches, see the following sections.

Write logs by using client libraries or the API

You can write log data by using the Cloud Logging client libraries (/logging/docs/reference/libraries), which call the Cloud Logging API, or by directly calling the Cloud Logging API. Client libraries can simplify population of the special JSON fields by automatically capturing some information and by providing interfaces to appropriately populate the fields. However, for full control over the structure of your payloads, directly call the Cloud Logging API and pass the full `LogEntry` (/logging/docs/reference/v2/rest/v2/LogEntry) structure to the Cloud Logging API.

For more information, see the `entries.write` (/logging/docs/reference/v2/rest/v2/entries/write) reference.

For code examples, see Writing structured logs (/logging/docs/samples/logging-write-log-entry).

Write logs by using the gcloud CLI

You can write log data by using the gcloud CLI. The interface supports unstructured logs and structured logs. When you want to write a structured log, provide the command a serialized JSON object.

For a quickstart, see Write and query log entries with the Google Cloud CLI (/logging/docs/write-query-log-entries-gcloud).

For code examples, see the `gcloud logging write` (/sdk/gcloud/reference/logging/write) reference.

Write logs by using BindPlane

You can use the BindPlane service to send logs to Logging. For these logs, the payloads are in JSON format and are structured according to the source system. For information on finding and viewing logs ingested by using BindPlane, see the BindPlane Quickstart Guide (https://bindplane.com/docs/getting-started/quickstart-guide).

Write logs by using an agent

Note: The content of this section only applies to Compute Engine VM. This section doesn't apply if you are using other services such as Google Kubernetes Engine, App Engine flexible environment, and Cloud Run functions.

To get logs from your Compute Engine instances, you can use the Ops Agent (/logging/docs/agent/ops-agent) or the legacy Cloud Logging agent (/logging/docs/agent/logging). Both agents can collect metrics from third-party applications, and both provide support for structured logging:

- The Ops Agent (/logging/docs/agent/ops-agent) is the recommended agent for collecting telemetry from your Compute Engine instances. This agent combines logging and metrics into a single agent, provides a YAML-based configuration, and features high-throughput logging.

  For information about how to configure the Ops Agent to support structured logging or to customize the form of a structured log, see Configure the Ops Agent (/logging/docs/agent/ops-agent/configuration).

- The legacy Cloud Logging agent (/logging/docs/agent/logging) collects logs. This agent doesn't collect other forms of telemetry.

The remainder of this section is specific to the legacy Logging agent.

Logging agent: special JSON fields

Some fields in the JSON object are recognized as special by the legacy Logging agent and extracted into the `LogEntry` (/logging/docs/reference/v2/rest/v2/LogEntry) structure. These special JSON fields can be used to set the following fields in the `LogEntry`:

- `severity`
- `spanId`
- `labels` defined by the user
- `httpRequest`

Because JSON is more precise and versatile than text lines, you can use JSON objects to write multiline messages and add metadata.

To create structured log entries for your applications using the simplified format, see the following table, which lists the fields and their values in JSON:

Note: Each field is optional.

JSON log field	`LogEntry` (/logging/docs/reference/v2/rest/v2/LogEntry) field	Cloud Logging agent function
`severity`	`severity`	The Logging agent attempts to match a variet strings, which includes the list of LogSeverity (/logging/docs/reference/v2/rest/v2/LogEntr recognized by the Logging API.
`message`	`textPayload` (or part of `jsonPayload`)	The message that appears on the log entry lin
`log` (legacy Google Kubernetes Engine only)	`textPayload`	Only applies to legacy Google Kubernetes Eng purpose fields, only a `log` field remains, then **Payload**.
`httpRequest`	`httpRequest`	A structured record in the format of the **LogE** (/logging/docs/reference/v2/rest/v2/LogEntr
time-related fields	`timestamp`	For more information, see Time-related fields (/logging/docs/agent/logging/configuration#
`logging.googleapis.com/insertId`	`insertId`	For more information, see `insertId` (/logging/docs/reference/v2/rest/v2/LogEntr `LogEntry` page.

JSON log field	**LogEntry** (/logging/docs/reference/v2/rest/v2/LogEntry) field	Cloud Logging agent function
`logging.googleapis.com/labels`	`labels`	The value of this field must be a structured re see **`labels`** (/logging/docs/reference/v2/rest/v2/LogEnt **`LogEntry`** page.
`logging.googleapis.com/operation`	`operation`	The value of this field is also used by the Logs log entries. For more information, see **`operat`** (/logging/docs/reference/v2/rest/v2/LogEnt the **`LogEntry`** page.
`logging.googleapis.com/sourceLocation`	`sourceLocation`	Source code location information associated more information, see **`LogEntrySourceLoc`** (/logging/docs/reference/v2/rest/v2/LogEnt on the **`LogEntry`** page.
`logging.googleapis.com/spanId`	`spanId`	The span ID within the trace associated with t information, see **`spanId`** (/logging/docs/reference/v2/rest/v2/LogEnt **`LogEntry`** page.
`logging.googleapis.com/trace`	`trace`	Resource name of the trace associated with t information, see **`trace`** (/logging/docs/reference/v2/rest/v2/LogEnt **`Entry`** page.
`logging.googleapis.com/trace_sampled`	`traceSampled`	The value of this field must be either **`true`** or information, see **`traceSampled`** (/logging/docs/reference/v2/rest/v2/LogEnt on the **`LogEntry`** page.

To create log entries in the simplified format, create a JSON representation of the entry using the fields. All of the fields are optional.

The following is an example of a simplified JSON log entry:

```
{
  "severity":"ERROR",
  "message":"There was an error in the application.",
  "httpRequest":{
    "requestMethod":"GET"
  },
  "times":"2020-10-12T07:20:50.52Z",
  "logging.googleapis.com/insertId":"42",
  "logging.googleapis.com/labels":{
    "user_label_1":"value_1",
    "user_label_2":"value_2"
  },
  "logging.googleapis.com/operation":{
    "id":"get_data",
    "producer":"github.com/MyProject/MyApplication",
    "first":"true"
  },
  "logging.googleapis.com/sourceLocation":{
    "file":"get_data.py",
    "line":"142",
    "function":"getData"
  },
  "logging.googleapis.com/spanId":"000000000000004a",
  "logging.googleapis.com/trace":"projects/my-projectid/traces/06796866738c859f2
  "logging.googleapis.com/trace_sampled":false
}
```

The following is an example of the resulting log entry:

```
{
  "insertId": "42",
  "jsonPayload": {
    "message": "There was an error in the application",
    "times": "2020-10-12T07:20:50.52Z"
  },
  "httpRequest": {
    "requestMethod": "GET"
  },
  "resource": {
    "type": "k8s_container",
    "labels": {
      "container_name": "hello-app",
      "pod_name": "helloworld-gke-6cfd6f4599-9wff8",
      "project_id": "stackdriver-sandbox-92334288",
      "namespace_name": "default",
      "location": "us-west4",
      "cluster_name": "helloworld-gke"
    }
  },
  "timestamp": "2020-11-07T15:57:35.945508391Z",
  "severity": "ERROR",
  "labels": {
    "user_label_2": "value_2",
    "user_label_1": "value_1"
  },
  "logName": "projects/stackdriver-sandbox-92334288/logs/stdout",
  "operation": {
    "id": "get_data",
    "producer": "github.com/MyProject/MyApplication",
    "first": true
  },
  "trace": "projects/my-projectid/traces/06796866738c859f2f19b7cfb3214824",
  "sourceLocation": {
    "file": "get_data.py",
    "line": "142",
    "function": "getData"
  },
  "receiveTimestamp": "2020-11-07T15:57:42.411414059Z",
  "spanId": "000000000000004a"
}
```

Logging agent: configuration

The legacy Logging agent, **`google-fluentd`**, is a Cloud Logging-specific packaging of the Fluentd (http://www.fluentd.org/) log data collector. The Logging agent comes with the default Fluentd configuration and uses Fluentd input plugins to pull event logs from external sources such as files on disk, or to parse incoming log records.

Fluentd has a list of supported parsers (https://docs.fluentd.org/parser#list-of-built-in-parsers) that extract logs and convert them into structured (JSON) payloads.

By configuring a log source with **`format [PARSER_NAME]`**, you can build on the built-in parsers provided by Fluentd. For information about configuring the legacy Logging agent, see Configure the Logging agent (/logging/docs/agent/logging/configuration).

The following code samples show the Fluentd configuration, the input log record, and the output structured payload, which is part of a Cloud Logging log entry:

- Fluentd configuration:

```
<source>
  @type tail

  format syslog # This uses a predefined log format regex named
                # `syslog`. See details at https://docs.fluentd.org/parse

  path /var/log/syslog
  pos_file /var/lib/google-fluentd/pos/syslog.pos
  read_from_head true
  tag syslog
</source>
```

- Log record (input):

```
<6>Feb 28 12:00:00 192.168.0.1 fluentd[11111]: [error] Syslog test
```

- Structured payload (output):

```
jsonPayload: {
    "pri": "6",
    "host": "192.168.0.1",
    "ident": "fluentd",
    "pid": "11111",
    "message": "[error] Syslog test"
}
```

For more information about how the `syslog` parser works, see the detailed Fluentd documentation (https://docs.fluentd.org/parser/syslog).

Logging agent: standard parsers enabled by default

The following table includes the standard parsers that are included in the agent if you enable structured logging:

Parser Name	Configuration file
`syslog` (https://docs.fluentd.org/parser/syslog)	`/etc/google-fluentd/config.d/syslog.conf`
`nginx` (https://docs.fluentd.org/parser/nginx)	`/etc/google-fluentd/config.d/nginx.conf`
`apache2` (https://docs.fluentd.org/parser/apache2)	`/etc/google-fluentd/config.d/apache.conf`
`apache_error` (https://docs.fluentd.org/parser/apache_error)	`/etc/google-fluentd/config.d/apache.conf`

For instructions on enabling structured logging when installing the legacy Logging agent, see the Installation (#structured-log-install) section.

Logging agent: installation

To enable structured logging, you must change the default configuration of the legacy Logging agent when installing or reinstalling it. Enabling structured logging replaces the previously listed configuration files but doesn't change the operation of the agent itself.

When you enable structured logging, the listed logs are converted to log entries with different formats than they had before you enabled structured logs. If the logs are being routed to destinations outside of Logging, the change could affect any post-processing applications. For example, if routing logs to BigQuery, BigQuery rejects the new log entries for the remainder of the day as having an incorrect schema (/logging/docs/export/bigquery).

For instructions on installing the legacy Logging agent and enabling structured logging, refer to Installing the Logging agent (/logging/docs/agent/logging/installation#joint-install).

You can find the legacy Logging agent configuration files at **`/etc/google-fluentd/config.d/`**, which should now include the Standard parsers enabled by default (#default-parsers).

Logging agent: configure Apache access log format

By default, the legacy Logging agent stores Apache access log data in the **`jsonPayload`** field. For example:

```
{
  "logName": ...,
  "resource": ...,
  "httpRequest": ...,
  "jsonPayload": {
    "user"   : "some-user",
    "method" : "GET",
    "code"   : 200,
    "size"   : 777,
    "host"   : "192.168.0.1",
    "path"   : "/some-path",
    "referer": "some-referer",
    "agent"  : "Opera/12.0"
  },
  ...
}
```

Alternatively, you can configure the legacy Logging agent to extract certain fields to the **`httpRequest`** field. For example:

```
{
  "logName": ...,
  "resource": ...,
  "httpRequest": {
    "requestMethod": "GET",
    "requestUrl": "/some-path",
    "requestSize": "777",
    "status": "200",
    "userAgent": "Opera/12.0",
    "serverIp": "192.168.0.1",
    "referrer":"some-referrer",
  },
  "jsonPayload": {
    "user":"some-user"
  },
  ...
}
```

Configuring the **`httpRequest`** field, as shown in the prior sample, assists tracing: the Google Cloud console presents all logs for a given HTTP request in a parent-child hierarchy.

To configure this extraction, add the following to the end of your **`/etc/google-fluentd/config.d/apache.conf`**:

```
<filter apache-access>
  @type record_transformer
  enable_ruby true
  <record>
    httpRequest ${ {"requestMethod" => record['method'], "requestUrl" => record[
    "referer" => record['referer']} }
  </record>
  remove_keys method, path, size, code, agent, host, referer
</filter>
```

For more details on how to configure your log entries, see Modifying log records (/logging/docs/agent/logging/configuration#modifying_log_records).

Logging agent: configure nginx access log format

By default, the legacy Logging agent stores nginx access log data in the **`jsonPayload`** field. For example:

```
{
  "logName": ...,
  "resource": ...,
  "httpRequest": ...,
  "jsonPayload": {
    "remote":"127.0.0.1",
    "host":"192.168.0.1",
    "user":"some-user",
    "method":"GET",
    "path":"/some-path",
    "code":"200",
    "size":"777",
    "referrer":"some-referrer",
    "agent":"Opera/12.0",
    "http_x_forwarded_for":"192.168.3.3"
  },
  ...
}
```

Alternatively, you can configure the legacy Logging agent to extract certain fields to the `httpRequest` field. For example:

```
{
  "logName": ...,
  "resource": ...,
  "httpRequest": {
    "requestMethod": "GET",
    "requestUrl": "/some-path",
    "requestSize": "777",
    "status": "200",
    "userAgent": "Opera/12.0",
    "remoteIp": "127.0.0.1",
    "serverIp": "192.168.0.1",
    "referrer":"some-referrer",
  },
  "jsonPayload": {
    "user":"some-user",
    "http_x_forwarded_for":"192.168.3.3"
  },
  ...
}
```

Configuring the `httpRequest` field, as shown in the prior sample, assists tracing: the Google Cloud console presents all logs for a given HTTP request in a parent-child hierarchy.

To configure this extraction, add the following to the end of your **`/etc/google-fluentd/config.d/nginx.conf`**:

```
<filter nginx-access>
  @type record_transformer
  enable_ruby true
  <record>
    httpRequest ${ {"requestMethod" => record['method'], "requestUrl" => record[
  </record>
  remove_keys method, path, size, code, agent, remote, host, referer
</filter>
```

For more details on how to configure your log entries, see Modifying log records (/logging/docs/agent/logging/configuration#modifying_log_records).

Write your own parser

If your logs aren't supported by the standard parsers, you can write your own parser. Parsers consist of a regular expression that is used to match log records and apply labels to the pieces.

The following code examples show a log line in the log record, a configuration with a regular expression that indicates the format of the log line, and the stored log entry:

- A log line in the log record:

  ```
  REPAIR CAR $500
  ```

- A configuration with a regular expression that indicates the format of the log line:

  ```
  $ sudo vim /etc/google-fluentd/config.d/test-structured-log.conf
  $ cat /etc/google-fluentd/config.d/test-structured-log.conf
  <source>
  ```

```
  @type tail

  # Format indicates the log should be translated from text to
  # structured (JSON) with three fields, "action", "thing" and "cost",
  # using the following regex:
  format /(?<action>\w+) (?<thing>\w+) \$(?<cost>\d+)/
  # The path of the log file.
  path /tmp/test-structured-log.log
  # The path of the position file that records where in the log file
  # we have processed already. This is useful when the agent
  # restarts.
  pos_file /var/lib/google-fluentd/pos/test-structured-log.pos
  read_from_head true
  # The log tag for this log input.
  tag structured-log
</source>
```

- The resulting log entry:

```
{
insertId:  "eps2n7g1hq99qp"
jsonPayload: {
  "action": "REPAIR"
  "thing": "CAR"
  "cost": "500"
}
labels: {
  compute.googleapis.com/resource_name:  "add-structured-log-resource"
}
logName:  "projects/my-sample-project-12345/logs/structured-log"
receiveTimestamp:  "2023-03-21T01:47:11.475065313Z"
resource: {
  labels: {
    instance_id:  "3914079432219560274"
    project_id:  "my-sample-project-12345"
    zone:  "us-central1-c"
  }
  type:  "gce_instance"
}
timestamp:  "2023-03-21T01:47:05.051902169Z"
}
```

Troubleshoot issues

To troubleshoot common issues found with installing or interacting with the legacy Logging agent, see Troubleshooting the agent (/logging/docs/agent/logging/troubleshooting).

What's next

- To query and view log entries, see View logs by using the Logs Explorer (/logging/docs/view/logs-explorer-interface).
- To read log entries using the Google Cloud CLI, see Reading log entries (/logging/docs/api/gcloud-logging#reading_log_entries).
- To read log entries using the Logging API, see the `entries.list` (/logging/docs/reference/v2/rest/v2/entries/list) method.