This page details the formatting and rules that apply when exporting log entries from Stackdriver Logging to BigQuery.

BigQuery table schemas for exported logs are based on the structure of the LogEntry (/logging/docs/reference/v2/rest/v2/LogEntry) type and the contents of the log payloads. Stackdriver Logging also applies some special rules to shorten BigQuery schema field names for audit logs (/logging/docs/audit). You can view the table schema by selecting a table with exported log entries in the BigQuery web UI (/bigquery/docs/bigquery-web-ui).

There are a few naming conventions that apply to the log entry fields:

- For log entry fields that are part of the LogEntry (/logging/docs/reference/v2/rest/v2/LogEntry) type, the corresponding BigQuery field names are exactly the same as the log entry fields.
- For any user-supplied fields, letter case is normalized to the lower case but naming is otherwise preserved.
    - For fields in structured payloads, as long as the `@type` specifier is not present, letter case is normalized to the lower case but naming is otherwise preserved.

      For information on structured payloads where the `@type` specifier is present, go to Payload fields with @type (#type-specifier).

The following examples show how these naming conventions are applied:

Log entry field	LogEntry (/logging/docs/reference/v2/rest/v2/LogEntry) type mapping	BigQuery field name
`insertId`	`insertId`	`insertId`
`textPayload`	`textPayload`	`textPayload`
`httpRequest.status`	`httpRequest.status`	`httpRequest.status`

Log entry field	LogEntry (/logging/docs/reference/v2/rest/v2/LogEntry) type mapping	BigQuery field name
`httpRequest.requestMethod.GET`	`httpRequest.requestMethod.[ABC]`	`httpRequest.requestMethod.get`
`resource.labels.moduleid`	`resource.labels.[ABC]`	`resource.labels.moduleid`
`jsonPayload.MESSAGE`	`jsonPayload.[ABC]`	`jsonPayload.message`
`jsonPayload.myField.mySubfield`	`jsonPayload.[ABC].[XYZ]`	`jsonPayload.myfield.mysubfield`

The mapping of structured payload fields to BigQuery field names is more complicated when the structured field contains a `@type` specifier. This is discussed in the following section.

This section discusses special BigQuery schema field names for log entries whose payloads contain type specifiers (`@type` fields). This includes exported audit log entries held in BigQuery. For example, this section explains why an audit log entry's `protoPayload` field might be mapped to the BigQuery schema field `protopayload_auditlog`.

Payloads in log entries can contain structured data, and that structured data can have nested structured fields. Any structured field can include an optional **type specifier** in the following format:

Structured fields that have type specifiers are customarily given BigQuery field names that have a [TYPE] appended to their field name.

For example, the following table shows the mapping of the top-level structured payload fields to BigQuery field names:

Payload	Payload @type	Payload field	BigQuery field name
`jsonPayload`	(none)	`statusCode`	`jsonPayload.statusCode`
`jsonPayload`	`type.googleapis.com/abc.Xyz`	`statusCode`	`jsonPayload_abc_xyz.statuscode`
`protoPayload`	(none)	`statusCode`	`protoPayload.statuscode`
`protoPayload`	`type.googleapis.com/abc.Xyz`	`statusCode`	`protopayload_abc_xyz.statuscode`

If `jsonPayload` or `protoPayload` contains other structured fields, then those inner fields are mapped as follows:

- If the nested structured field **does not** have a `@type` specifier, then its BigQuery field name is the same as the original field name, except it is normalized to lowercase letters.
- If the nested structured field **does** have a `@type` specifier, then its BigQuery field name has `[TYPE]` (respelled) appended to the field name and is normalized to lowercase letters.

There a few exceptions to the preceding rules for fields with type specifiers:

- In App Engine request logs, the payload's name in logs exported to BigQuery is `protoPayload`, even though the payload has a type specifier.
- Stackdriver Logging applies some special rules to shorten BigQuery schema field names for audit logs. This is discussed in the Exported audit log schema fields (#audit-logs) section on this page.

This example shows how structured payload fields are named and used when exported to BigQuery.

Assume that a log entry's payload is structured like the following:

The mapping to BigQuery fields is as follows:

- The fields `jsonPayload` and `name_a` are structured, but they do not have `@type` specifiers. Their BigQuery names are `jsonPayload` and `name_a`, respectively.
- The fields `sub_a` and `sub_b` are not structured, so their BigQuery names are `sub_a` and `sub_b`, respectively.
- The field `name_b` has a `@type` specifier, whose [TYPE] is `google.cloud.v1.SubType`. Therefore, its BigQuery name is `name_b_google_cloud_v1_subtype`.

In summary, the following 5 BigQuery names are defined for the log entry's payload:

If you are not working with audit logs that have been exported to BigQuery, then you can skip this section.

The audit log payload fields `protoPayload.request`, `protoPayload.response`, and `protoPayload.metadata` have `@type` specifiers but are treated as JSON data. That is, their BigQuery schema names are their field names with `Json` appended to them, and they contain string data in JSON format.

The two sets of audit log payload field names are listed in the following table:

Log entry field	BigQuery field name
`protoPayload`	`protopayload_auditlog`
`protopayload.metadata`	`protopayload_auditlog.metadataJson`
`protoPayload.serviceData`	`protopayload_auditlog.servicedata_v1_bigquery` Example: `protopayload_auditlog.servicedata_v1_bigquery.tableInsertRequest`
`protoPayload.request`	`protopayload_auditlog.requestJson`
`protoPayload.response`	`protopayload_auditlog.responseJson`

Note that the `serviceData` naming convention is specific to audit logs that are generated by BigQuery and that are then exported from Stackdriver Logging to BigQuery. Those audit log entries contain a `serviceData` field that has a @type specifier of `type.googleapis.com/google.cloud.bigquery.logging.v1.auditdata`.

An audit log entry generated by BigQuery has a field with the following name:

If this log entry were then exported to BigQuery, how would the `tableInsertRequest` field be referenced? Before the name shortening, the corresponding exported field name would be:

After the name shortening, the same field is referenced in BigQuery tables like this:

This section provides an overview of partitioned tables (/bigquery/docs/partitioned-tables) for logs exports to BigQuery.

When you export logs to a BigQuery dataset, Logging creates tables to hold the exported log entries. There are two table types by which Logging organizes the data it exports: **date-sharded tables** and **partitioned tables**. Both table types partition the logs data based on log entries' `timestamp` fields. However, there are two key differences between the table types:

- Performance: A partitioned table divides a large table into smaller partitions, so that you can improve query performance and, thus, better control your BigQuery costs by reducing the number of bytes read by a query.
- Table nomenclature: The table types use different naming conventions, as discussed in the section below.

Exported log entries are placed in BigQuery tables whose names are based on the entries' log names and timestamps.

The following table shows examples of how log names and sample timestamps are mapped to table names in BigQuery:

Log name	Log entry timestamp	BigQuery table name (date-sharded)	BigQuery table (partitioned)
syslog	2017-05-23T18:19:22.135Z	syslog_20170523	syslog
apache-access	2017-01-01T00:00:00.000Z	apache_access_20170101	apache_acces
compute.googleapis.com/activity_log	2017-12-31T23:59:59.999Z	compute_googleapis_com_activity_log_20171231	compute_goog

When creating a sink to export your logs to BigQuery, you can use either date-sharded tables or partitioned tables. The default selection is a date-sharded table.

For instructions using the Google Cloud Console, go to Exporting with the Logs Viewer (/logging/docs/export/configure_export_v2).

For instructions using Cloud SDK, the command-line interface, go to `gcloud alpha logging sinks create` (/sdk/gcloud/reference/alpha/logging/sinks/create).

The first exported log entry determines the schema for the destination BigQuery table. The log entry generates a table whose columns are based on the log entry's fields and their types. If new fields appear in subsequent log entries, the table schema is updated. However, if the value type changes for an existing field, then newer log entries that don't match the schema are dropped.

For example, if your initial export contains a log entry where `jsonPayload.user_id` is a `string`, then that log entry generates a table with a string type for that field. If you start logging `jsonPayload.user_id` as an `array`, those log entries aren't inserted into the table and are lost.

Logging communicates this data loss for the Google Cloud project that contains the export in the following ways:

- Project Owners (/logging/docs/access-control) receive an email. Details include: Google Cloud project ID, sink name, and export destination.
- The Google Cloud Console Activity page displays an error, `Stackdriver Config error`. Details include the sink name and export destination, and a link to an example of a log entry that caused the error.
- The system logs-based metric `exports/error_count` (/logging/docs/logs-based-metrics/#system_logs-based_metrics) informs you of the total number of log entries that weren't exported due to errors.

To correct the issue for subsequent log entries, so that you don't incur further data loss, fix the field type so that it matches the current schema. You can also rename the table or change the sink's parameters, so that Logging recreates the table in a different dataset. For instructions, go to Updating sinks (/logging/docs/export/configure_export_v2#updating_sinks).

To view your logs using the BigQuery web UI (/bigquery/docs/bigquery-web-ui) , select a table with your exported log entries.