

Split audit log entries

This document describes how Cloud Logging splits oversized [audit log entries](/logging/docs/audit) (/logging/docs/audit) and provides guidance on how to reassemble these split audit logs.

When a single audit log entry exceeds the [size limit](/logging/quotas#audit-logging-limits) (/logging/quotas#audit-logging-limits), Cloud Logging splits that entry and distributes the data contained in the original audit log entry across several entries. Users might want to reassemble the split audit logs as the individual split log entries don't contain all the fields from the original audit log.

Recognizing split audit log entries

Split log entries contain information about the original entry from which they were split. If a log entry contains a [`split`](/logging/docs/reference/v2/rest/v2/LogEntry#FIELDS.split) (/logging/docs/reference/v2/rest/v2/LogEntry#FIELDS.split) field, then the entry is the result of splitting a larger original log entry. The `split` field is a [`LogSplit`](/logging/docs/reference/v2/rest/v2/LogEntry#LogSplit) (/logging/docs/reference/v2/rest/v2/LogEntry#LogSplit) object that contains the information needed to identify related split log entries.

Each split log entry contains the following fields:

- `split.uid`: a unique identifier for the group of log entries that were split from a common original log entry. The value of this field is the same for all entries split from the original log entry.
- `split.index`: the position of this entry in the series of split entries. The first entry from the split has index 0. `split.index` is also appended to the `LogEntry.insertId` field.

★ **Note:** The `split.index` field doesn't display in the Logs Explorer when the `index` value is 0. While Logs Explorer doesn't show this field, Cloud Logging does properly set and process the value for this field.

- `split.totalSplits`: the number of log entries that the original log entry was split into. The value of this field is the same for all entries split from the original log entry.

How a log entry is split

When an oversized audit log entry is split, the fields are distributed among the resulting split log entries as follows:

- All fields except the `protoPayload` field are duplicated into each split entry.
- The following `protoPayload` subfields can be split across multiple entries:
 - `protoPayload.metadata`
 - `protoPayload.request`
 - `protoPayload.response`
- All other `protoPayload` subfields are included in all of the split entries.
- For the `protoPayload.metadata`, `protoPayload.request`, and `protoPayload.response` subfields, the following field types can be split across multiple entries:
 - **Struct**
(<https://developers.google.com/protocol-buffers/docs/reference/google.protobuf#google.protobuf.Struct>)
 - **Value**
(<https://developers.google.com/protocol-buffers/docs/reference/google.protobuf#google.protobuf.Value>)
 - **ListValue**
(<https://developers.google.com/protocol-buffers/docs/reference/google.protobuf#google.protobuf.ListValue>)
 - `string`
 - Any repeated fields, regardless of type

If a field is a member of a splittable field but isn't one of the splittable field types, then it's only present in one of the split logs.

For example, a boolean field in the `protoPayload.request` subfield can only appear in one split log entry, but a string field in the `protoPayload.request` subfield can have its content split across multiple split log entries.

For an example of how a long entry is split, see [Example log entry split](#) (#example).

Repeated fields with many values

When the value of the `protoPayload.metadata`, `protoPayload.request` or `protoPayload.response` field contains a list of repeated values, the list might be broken up and distributed across multiple split log entries.

For example, the list of values `["foo", "bar", "baz"]` might be split into 2 lists: `["foo", "ba"]` and `["", "r", "baz"]`. The first list is the entry with the `split.index` of 0, and the second list is in the entry with `split.index` of 1. The second list begins with an empty string to maintain the positions of the elements in the list and to indicate that elements in the same positions in the different lists must be joined together. In the example, `ba` is the second list element in entry 0, and `r` is the second list element in entry 1, so they are recombined in the order `bar` when reassembling the original list.

Large, non-repeated fields

When large, non-repeated `Struct` and `string` fields are split, these fields are handled as follows:

- A `string` field is split at the character level and not at the byte level. So, multi-byte characters aren't altered.
- `LogEntry.split.index` controls the ordering of the contents of split, non-repeated fields.

Reassemble split log entry

To reassemble a set of split logs, complete the following steps:


1. Sort the set of split audit logs by `LogEntry.split.index` in ascending order.
2. Create a copy of the first split log, where `LogEntry.split.index == 0`. This copy is the beginning of the reassembled log.
3. For the remaining log entries, iterate through all splittable fields of `protoPayload` and complete the following steps for each field:
 - a. If the field already exists in the reassembled log, append the content of that field into the reassembled log.
 - b. If the field doesn't exist in the reassembled log, copy that field into the reassembled log

When split, repeated fields preserve the index of its elements, so you can apply these steps at the element level when reassembling a repeated field.

4. After iterating through the splittable fields, clear `LogEntry.split` of the reassembled log.
5. Remove the `.0` suffix from the `LogEntry.insert_id` of the reassembled log.

Sample queries

To find all log entries that were split from the same original log entry, run the following query in the [Logs Explorer](/logging/docs/view/logs-explorer-interface) (/logging/docs/view/logs-explorer-interface), after replacing the *UID* variable with the chosen value:

```
split.uid="UID 
```

For example:

```
split.uid="abc123"
```

To find all log entries which are part of any split, run the following query:

```
split:*
```

Filter out split audit logs

You can exclude all split audit logs from a query by using the following filter:

```
split.totalSplits = 0
```

You can also include only the first entry of a split audit log and exclude the rest of the entries by using the following filter:

```
split.index = 0
```

Example log entry split

The following example shows an audit log entry before it's split into four new log entries. The new entries show how different fields are handled in the splitting operation.

Oversized audit log entry before splitting

```
{
  "insertId": "567",
  "logName": "projects/1234/logs/cloudaudit.googleapis.com%2Fdata_access",
  "resource": {
    "type": "audited_resource"
  },
  "protoPayload": {
    "serviceName": "example.googleapis.com",
    "methodName": "google.cloud.example.ExampleMethod",
    "resourceName": "projects/1234/resources/123",
    "status": {
      "code": 0
    },
    "authenticationInfo": {
      "principalEmail": "user@example_company.com"
    },
    "authorizationInfo": [
      {
        "resource": "example.googleapis.com/projects/1234/resources/123",
        "permission": "examples.get",
        "granted": "true"
      }
    ],
    "request" {
```

```

    "boolField": true,
    "numberField": 123,
    "stringField": "Very long string that needs 2 log entries.",
    "structField": {
      "nestedNumberField": 1337,
      "nestedStringField": "Another long string that needs 2 log entries.",
    },
    "listField" [
      {"value": "short 1"},
      {"value": "Yet another long string."},
      {"value": "short 2"},
      {"value": "short 3"},
    ]
  }
}
}

```

Oversized audit log entry after splitting

The original log entry is split into the following entries. Note that each entry includes the `split` object with a `uid` and a `totalSplits` value of 4. Each entry has a `split.index` value of 0, 1, 2, or 3, which indicates the order of the split log entries.

Split log entry, index 0

Here is the first split log entry, with a `split.index` value of 0.

```

{
  "insertId": "567.0",
  "logName": "projects/1234/logs/cloudaudit.googleapis.com%2Fdata_access",
  "resource": {
    "type": "audited_resource"
  },
  "split": {
    "uid": "789+2022-02-22T12:22:22.22+05:00",
    "index": 0,
    "totalSplits": 4,
  },
  "protoPayload": {
    // The following fields are included in all split entries

```

```

    "serviceName": "example.googleapis.com",
    "methodName": "google.cloud.example.ExampleMethod",
    "resourceName": "projects/1234/resources/123",
    "status": {
      "code": 0
    },
    "authenticationInfo": { // small size; included in all split entries
      "principalEmail": "user@example_company.com"
    },
    // The following field is included in this split entry only.
    "authorizationInfo": [
      {
        "resource": "spanner.googleapis.com/projects/1234/datasets/123",
        "permission": "databases.read",
        "granted": "true"
      }
    ],
    // The following field is split across all the split entries
    "request" {
      // boolField and numberField can only be in one split.
      "boolField": true,
      "numberField": 123,
      // Split with the next LogEntry.
      "stringField": "Very long string that ",
    }
  }
}

```

Split log entry, index 1

Here is the next split log entry, with a `split.index` value of 1.

```

{
  "insertId": "567.1",
  "logName": "projects/1234/logs/cloudaudit.googleapis.com%2Fdata_access",
  "resource": {
    "type": "audited_resource"
  },
  "split": {
    "uid": "567+2022-02-22T12:22:22.22+05:00",
    "index": 1,
  }
}

```

```

    "totalSplits": 4,
  },
  "protoPayload": {
    "serviceName": "example.googleapis.com",
    "methodName": "google.cloud.example.ExampleMethod",
    "resourceName": "projects/1234/resources/123",
    "status": {
      "code": 0
    },
    "authenticationInfo": {
      "principalEmail": "user@example_company.com"
    },
    "request" {
      // boolField and numberField aren't present
      // Continued from the previous entry.
      "stringField": "needs 2 log entries.",
      "structField": {
        "nestedNumberField": 1337,
        // Split with the next LogEntry.
        "nestedStringField": "Another long string ",
      }
    }
  }
}

```

Split log entry, index 2

Here is the next split log entry, with a `split.index` value of 2.

```

{
  "insertId": "567.2",
  "logName": "projects/1234/logs/cloudaudit.googleapis.com%2Fdata_access",
  "resource": {
    "type": "audited_resource"
  },
  "split": {
    "uid": "567+2022-02-22T12:22:22.22+05:00",
    "index": 2,
    "totalSplits": 4,
  },
  "protoPayload": {

```



```

"serviceName": "example.googleapis.com",
"methodName": "google.cloud.example.ExampleMethod",
"resourceName": "projects/1234/resources/123",
"status": {
  "code": 0
},
"authenticationInfo": {
  "principalEmail": "user@example_company.com"
},
request {
  "structField": {
    // Continued from the previous entry.
    "nestedStringField": "that needs 2 log entries.",
  }
  "listField" [
    {"value": "short 1"},
    {"value": "Yet another "}, // Split with the next LogEntry.
    // Missing two values, split with the next LogEntry.
  ]
}
}
}

```

Split log entry, index 3

Here is the final split log entry, with a `split.index` value of 3.

```

{
  "insertId": "567.3",
  "logName": "projects/1234/logs/cloudaudit.googleapis.com%2Fdata_access",
  "resource": {
    "type": "audited_resource"
  },
  "split": {
    "uid": "567+2022-02-22T12:22:22.22+05:00",
    "index": 3,
    "totalSplits": 4,
  },
  "protoPayload": {
    "serviceName": "example.googleapis.com",
    "methodName": "google.cloud.example.ExampleMethod",

```

```
"resourceName": "projects/1234/resources/123",
"status": {
  "code": 0
},
"authenticationInfo": {
  "principalEmail": "user@example_company.com"
},
"request" {
  "listField" [
    {}, // Padding to ensure correct positioning of elements in split repeat
    {"value": "long string."}, // Continuation of the second repeated field
    {"value": "short 2"},
    {"value": "short 3"},
  ]
}
}
```

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-04-30 UTC.