OpenTelemetry Custom Traces

Send your **custom traces** to Coralogix using our OpenTelemetry-compatible endpoint.

Overview

Custom traces can be delivered directly to the Coralogix OpenTelemetry-compatible endpoint using any gRPC client or OpenTelemetry SDKs.

The examples below guide you using gRPCurl and OpenTelemetry **Java** SDK. This is an alternative to using the OpenTelemetry Collector.

Prerequisites

If you are sending us your data using gRPCurl, you are required to have Git and gRPCurl installed.

Data Model

The custom metric API implementation is based on the OpenTelemetry tracing specification. This ensures that our tracing implementation adheres to industry best practices and can seamlessly integrate with other components and tools in the OpenTelemetry ecosystem.

Example Tracing Span


```
{
  "resource_spans": [
    {
      "resource": {
        "attributes": [
          {
            "key": "cx.application.name",
            "value": {
              "string_value": "my-test-application"
            }
          },
          {
            "key": "cx.subsystem.name",
            "value": {
              "string_value": "my-test-subsystem"
```

```
            }
          },
          {
            "key": "service.name",
            "value": {
              "string_value": "my-test-service"
            }
          }
        ]
      },
      "scope_spans": [
        {
          "scope": {
            "name": "test"
          },
          "spans": [
            {
              "trace_id": "d0XALf50O6rdjlMxreTJsA==",
              "span_id": "YolJ2eFjJCs=",
              "name": "test-span",
              "kind": "SPAN_KIND_INTERNAL",
              "start_time_unix_nano": "1665753217736722000",
              "end_time_unix_nano": "1665753217737156416",
              "status": {}
            }
          ]
        }
      ]
    }
  ]
}
```

Sending Data with gRPCurl

gRPC is a modern way of calling APIs on top of HTTP/2. Similar to cURL, gRPCurl is a command-line tool used to communicate with gRPC services.

Coralogix currently supports gRPC for its custom traces endpoint. REST APIs will be added in the future.

Assuming the example in the data model is saved as `traces.json`, use the following command to send your data to Coralogix:

```
# Clone OpenTelemetry protobuf definitions
git clone https://github.com/open-telemetry/opentelemetry-proto.git
# Send traces to Coralogix
grpcurl -v -d @ \
  -rpc-header 'Authorization: Bearer <send-your-data-api-key>' \
  -proto opentelemetry-
proto/opentelemetry/proto/collector/trace/v1/trace_service.proto \
  -import-path opentelemetry-proto \
  <open-telemetry-endpoint> \
  opentelemetry.proto.collector.trace.v1.TraceService/Export \
  < traces.json
```

Notes:

- For `<open-telemetry-endpoint>`, input the Coralogix OpenTelemetry endpoint associated with your Coralogix domain.
- For the `<send-your-data-api-key>`, input your Coralogix Send-Your-Data API key.
- Set the `start_time_unix_nano` and the `end_time_unix_nano` in the `traces.json` to a timestamp that is within the last 24 hours.

Sending Data Using the OpenTelemetry Java SDK

The example below guides you using OpenTelemetry Java SDK to send your custom traces to Coralogix. Others SDKs may also be used.

STEP 1. Add the following libraries to your maven pom.xml:

```
<dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-sdk-trace</artifactId>
  <version><!-- put a recent version of opentelemetry sdk here --><version>
</dependency>
<dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-exporter-otlp</artifactId>
  <version><!-- put a recent version of opentelemetry sdk her--><version>
</dependency>
```

STEP 2. Use this code snippet to create and report a span:

```
SdkTracerProvider traceProvider =
```

```
  SdkTracerProvider.builder()
    .addSpanProcessor(BatchSpanProcessor.builder(
      OtlpGrpcSpanExporter.builder()
        .setEndpoint("https://<open-telemetry-endpoint>")
        .addHeader("Authorization", "Bearer <send-your-data-api-key>")
        .build()
    ).build())
    .setResource(Resource.create(Attributes.of(
            ResourceAttributes.SERVICE_NAME, "my-test-service",
            AttributeKey.stringKey("cx.application.name"), "my-test-
application",
      AttributeKey.stringKey("cx.subsystem.name"), "my-test-subsystem"
        )))
    .build();

Tracer tracer = traceProvider.tracerBuilder("test").build();

Span span = tracer.spanBuilder("test-span")
  .setSpanKind(SpanKind.INTERNAL)
  .startSpan();
span.end();

traceProvider.forceFlush();
```

Limits & Quotas

Coralogix places a **hard limit of 10MB** of data to our OpenTelemetry endpoints, with a **recommendation of 2MB**.

Limits apply to single requests, regardless of timespan.

Documentation	Coralogix Endpoints
External	GitHub

Support

Need help?

Our world-class customer success team is available 24/7 to walk you through your setup and answer any questions that may come up.

Feel free to reach out to us **via our in-app chat** or by sending us an email at support@coralogix.com.

Last updated: April 22, 2025

Was this helpful?

Leave your feedback here.

◯ **Yes** ◯ **No**

Send