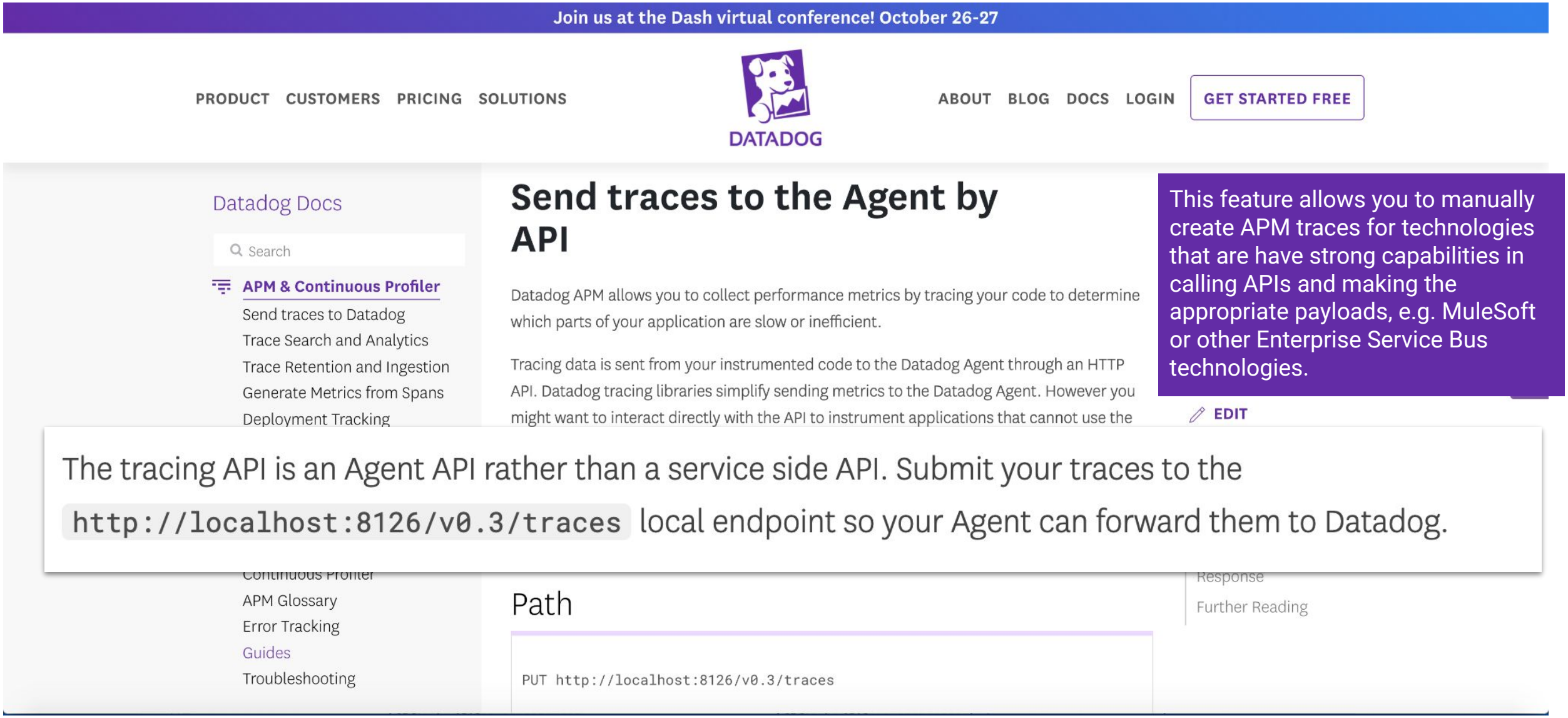
DATADOG

Application Performance Monitoring (APM) Distributed Tracing with MuleSoft

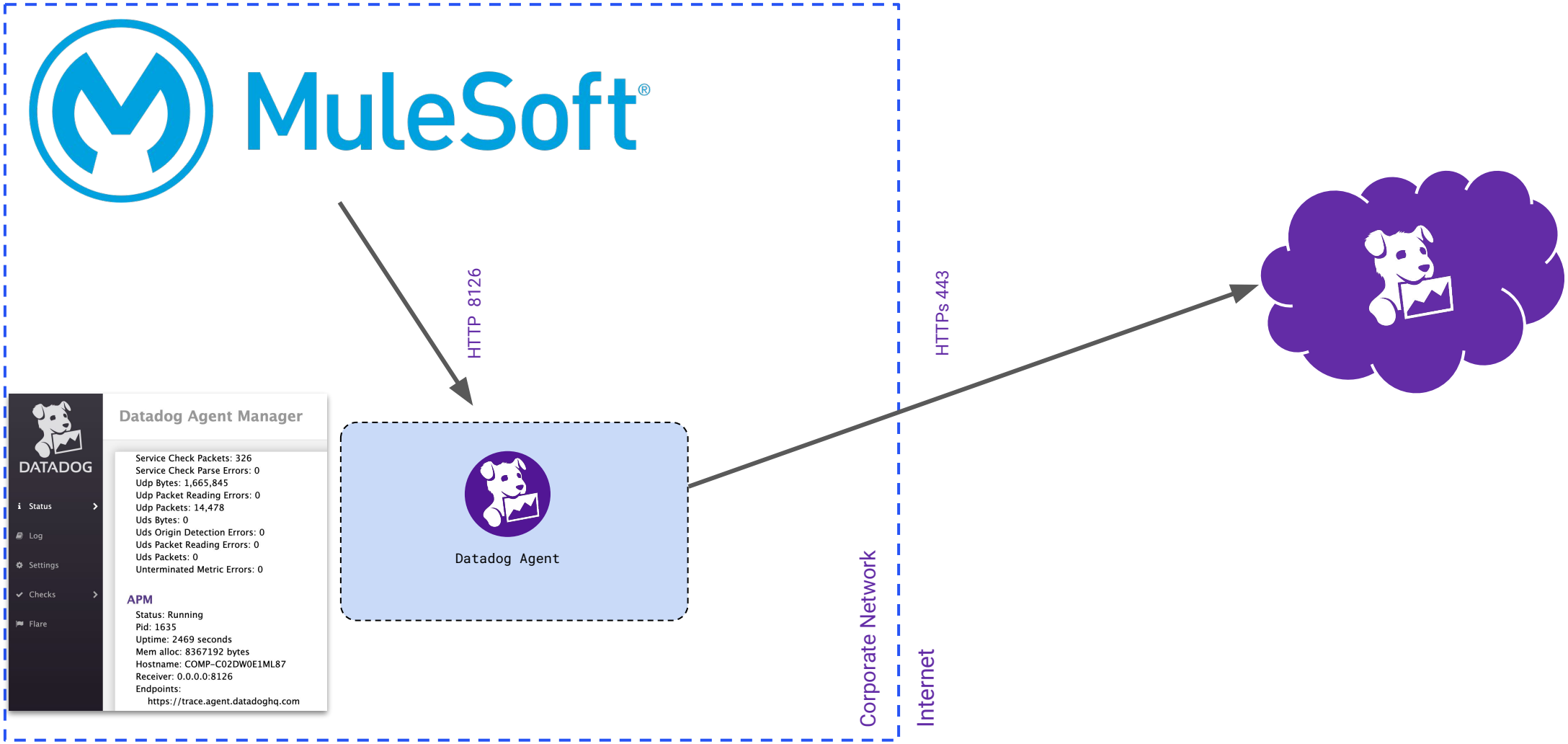
January 28, 2022

Send Traces to Agent Via API

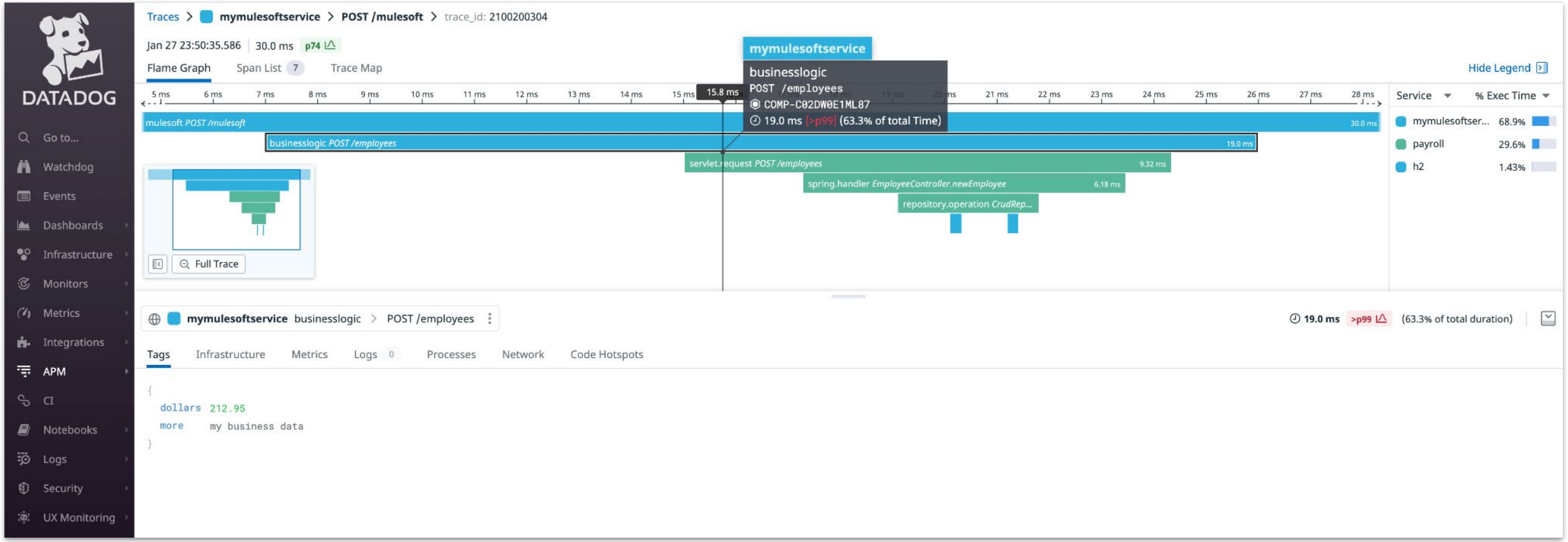


This feature allows you to manually create APM traces for technologies that are have strong capabilities in calling APIs and making the appropriate payloads, e.g. MuleSoft or other Enterprise Service Bus technologies.

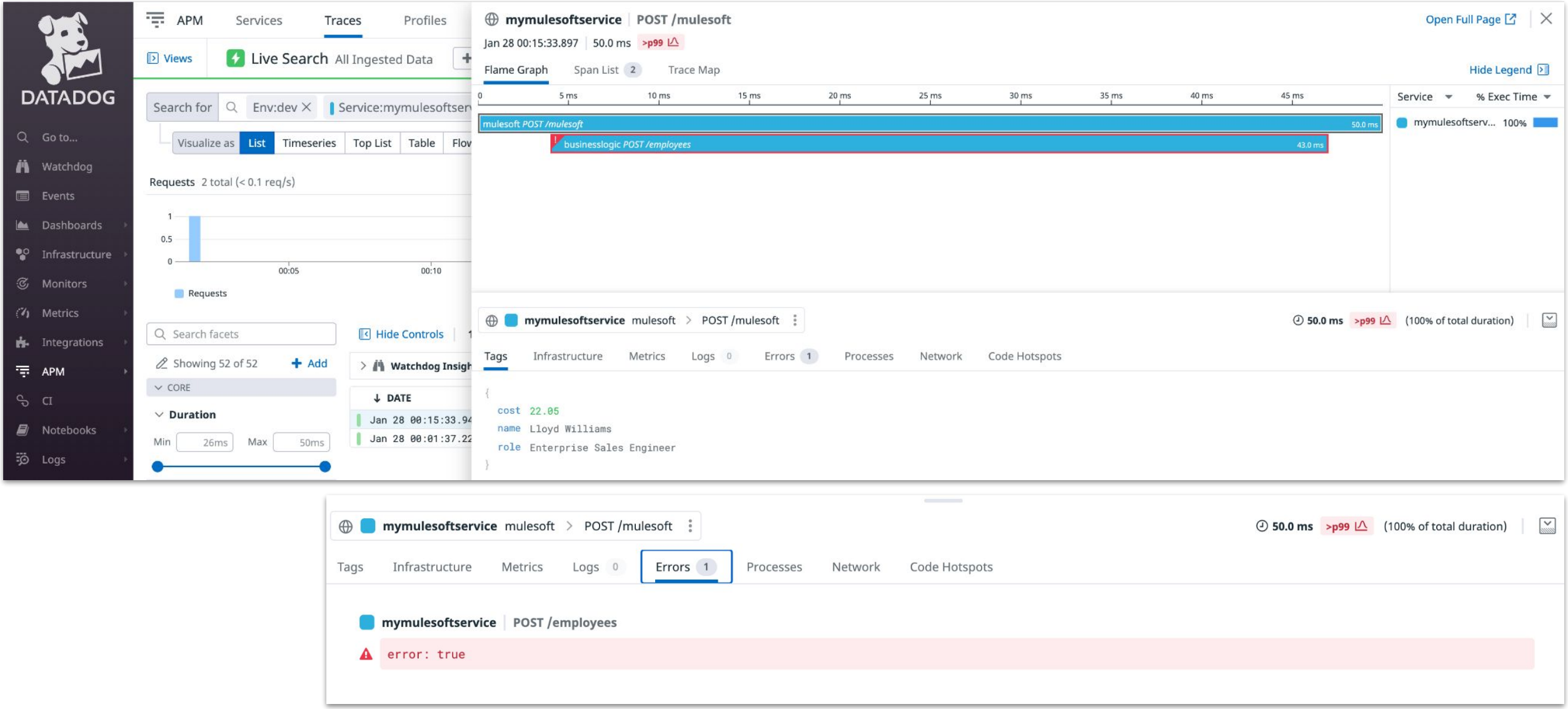
Send Traces to Agent Via API



Datadog APM


DATADOG
Go to...
Watchdog
Events
Dashboards
Infrastructure
Monitors
Metrics
Integrations
APM
CI
Notebooks
Logs
Security
UX Monitoring
Traces > mymulesoftservice > POST /mulesoft > trace_id: 2100200304
Jan 27 23:50:35.586 | 30.0 ms p74
Flame Graph
Span List 7
Trace Map
Hide Legend
mymulesoftservice
businesslogic
POST /employees
COMP-C02DW0E1ML87
19.0 ms [>p99] (63.3% of total Time)
15.8 ms
mulesoft POST /mulesoft
30.0 ms
businesslogic POST /employees
19.0 ms
servlet.request POST /employees
9.32 ms
spring.handler EmployeeController.newEmployee
6.18 ms
repository.operation CrudRep...
Full Trace
Service
% Exec Time
mymulesoftser... 68.9%
payroll 29.6%
h2 1.43%
mymulesoftservice businesslogic > POST /employees
19.0 ms >p99 (63.3% of total duration)
Tags
Infrastructure
Metrics
Logs 0
Processes
Network
Code Hotspots
{
dollars 212.95
more my business data
}

Datadog APM


DATADOG
APM
Services
Traces
Profiles
Views
Live Search All Ingested Data
Search for
Env:dev
Service:mymulesoftserv
Visualize as
List
Timeseries
Top List
Table
Requests 2 total (< 0.1 req/s)
Requests
Search facets
Hide Controls
Showing 52 of 52
Add
CORE
Duration
Min 26ms
Max 50ms
DATE
Jan 28 00:15:33.94
Jan 28 00:01:37.22
Go to...
Watchdog
Events
Dashboards
Infrastructure
Monitors
Metrics
Integrations
APM
CI
Notebooks
Logs
mymulesoftservice | POST /mulesoft
Jan 28 00:15:33.897 | 50.0 ms >p99
Open Full Page
Flame Graph
Span List 2
Trace Map
Hide Legend
mulesoft POST /mulesoft 50.0 ms
businesslogic POST /employees 43.0 ms
Service
% Exec Time
mymulesoftserv... 100%
mymulesoftservice mulesoft > POST /mulesoft
50.0 ms >p99 (100% of total duration)
Tags
Infrastructure
Metrics
Logs 0
Errors 1
Processes
Network
Code Hotspots
{
cost 22.05
name Lloyd Williams
role Enterprise Sales Engineer
}
mymulesoftservice mulesoft > POST /mulesoft
50.0 ms >p99 (100% of total duration)
Tags
Infrastructure
Metrics
Logs 0
Errors 1
Processes
Network
Code Hotspots
mymulesoftservice | POST /employees
error: true

Send Traces to Agent Via API

Request

Traces can be sent as an array of traces:

```
[ trace1, trace2, trace3 ]
```

And each trace is an array of spans:

```
trace1 = [ span, span2, span3 ]
```

and each span is a dictionary with a `trace_id`, `span_id`, `resource` and so on. Each span within a trace should use the same `trace_id`. However, `trace_id` and span_id must have different values.

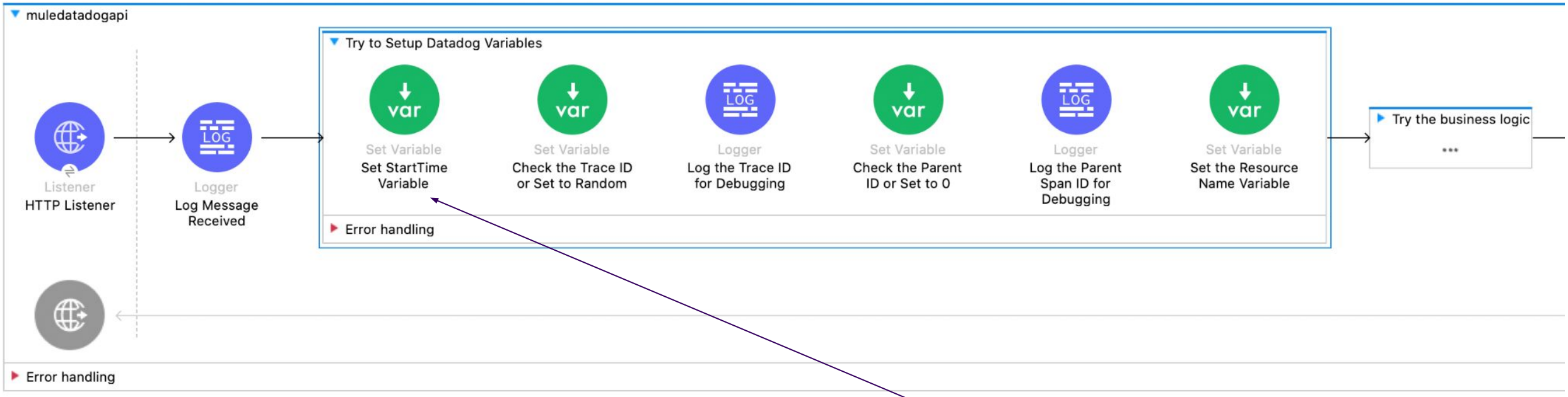
You can send multiple traces at once, but more importantly, you can send multiple spans within the same trace all at once.

Model

FIELD	TYPE	DESCRIPTION
`duration`	int64	The duration of the request in nanoseconds.
`error`	int32	Set this value to 1 to indicate if an error occurred. If an error occurs, you should pass additional information, such as the error message, type and stack information in the meta property.

The “Model” section of the documentation explains how to create the required payload (i.e. the ‘Body’ of the API call.)

Send Traces to Agent Via API



You start by tracking the overall 'Start' Time. We need "nanoseconds" but we can convert it later.

Display Name: Set StartTime Variable

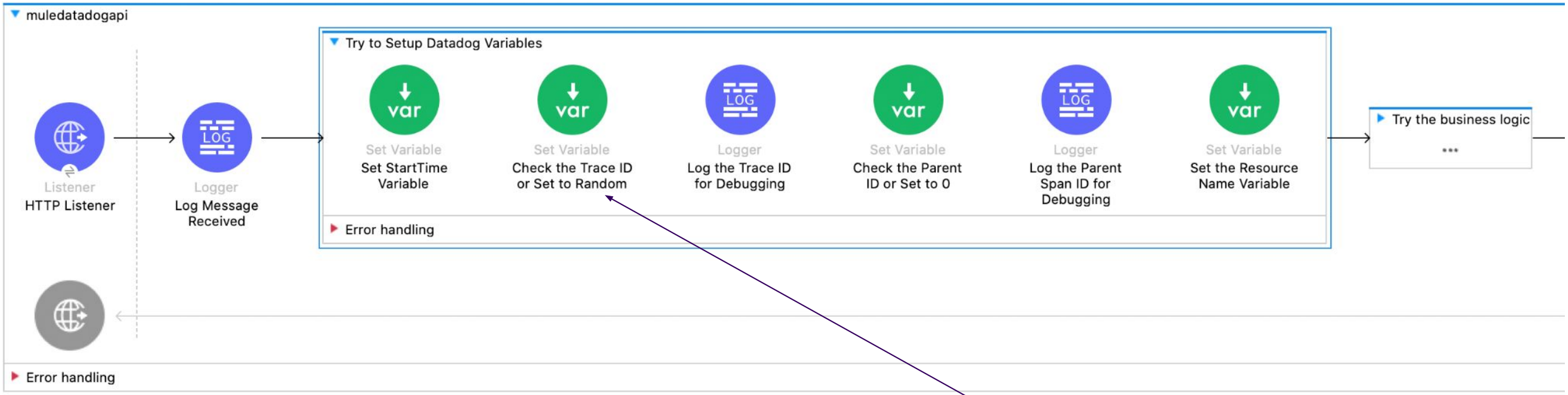
Settings

Name: startTime

Value: fx `#[ now() as Number {unit: "milliseconds"}`

DATADOG

Send Traces to Agent Via API

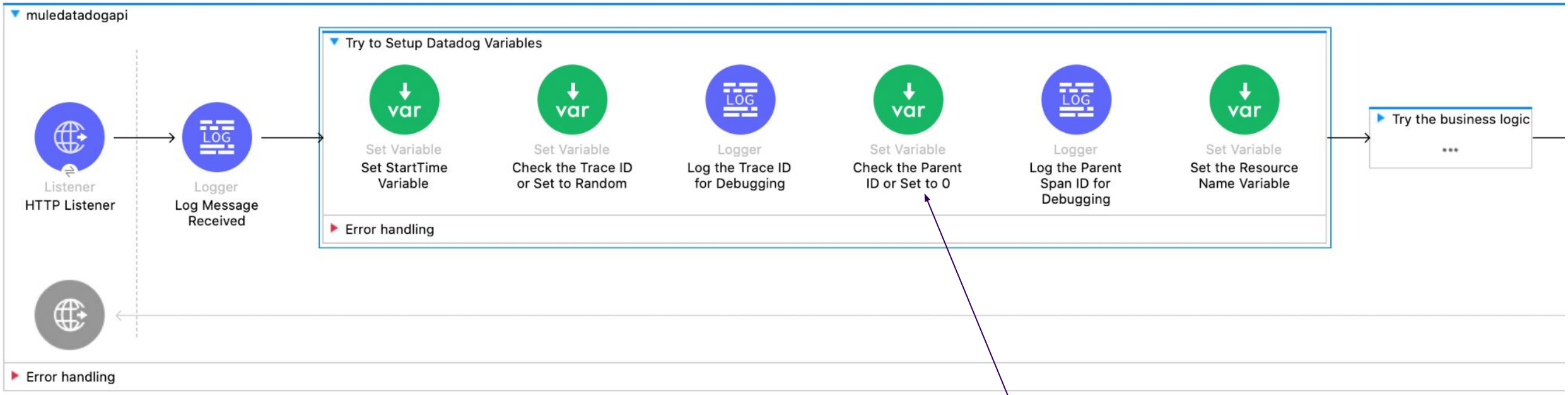


Check if there is an existing Datadog Trace ID. In this case, 'x-datadog-trace-id' in the HTTP header or generate a random one.

```
%dw 2.0
output application/java
---
if (attributes.headers['x-datadog-trace-id'] == null)
round(random() * 9223372036854775807) as String
else
attributes.headers['x-datadog-trace-id']
```

DATADOG

Send Traces to Agent Via API

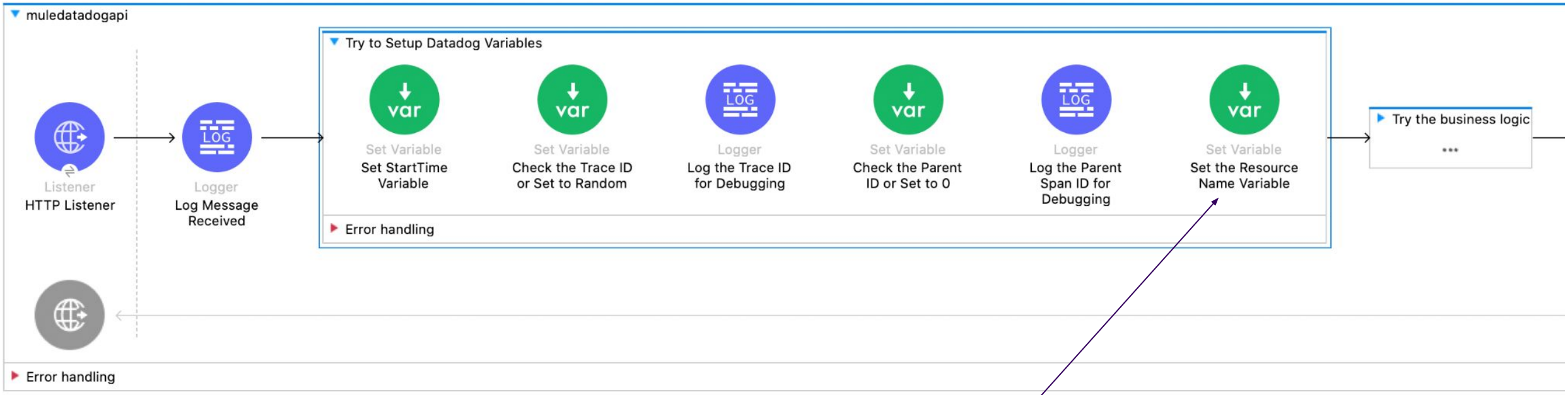



```
%dw 2.0
output application/java
---
if (attributes.headers['x-datadog-parent-id'] == null)
0 as String
else
attributes.headers['x-datadog-parent-id']
```

Also, check if there is an existing Datadog Parent ID. In this case, ‘x-datadog-parent-id’ in the HTTP header or use 0.

DATADOG

Send Traces to Agent Via API



Usually, the resource name is based on the URL. You can use automatically populated variables in MuleSoft to set the name.

Display Name: Set the Resource Name Variable

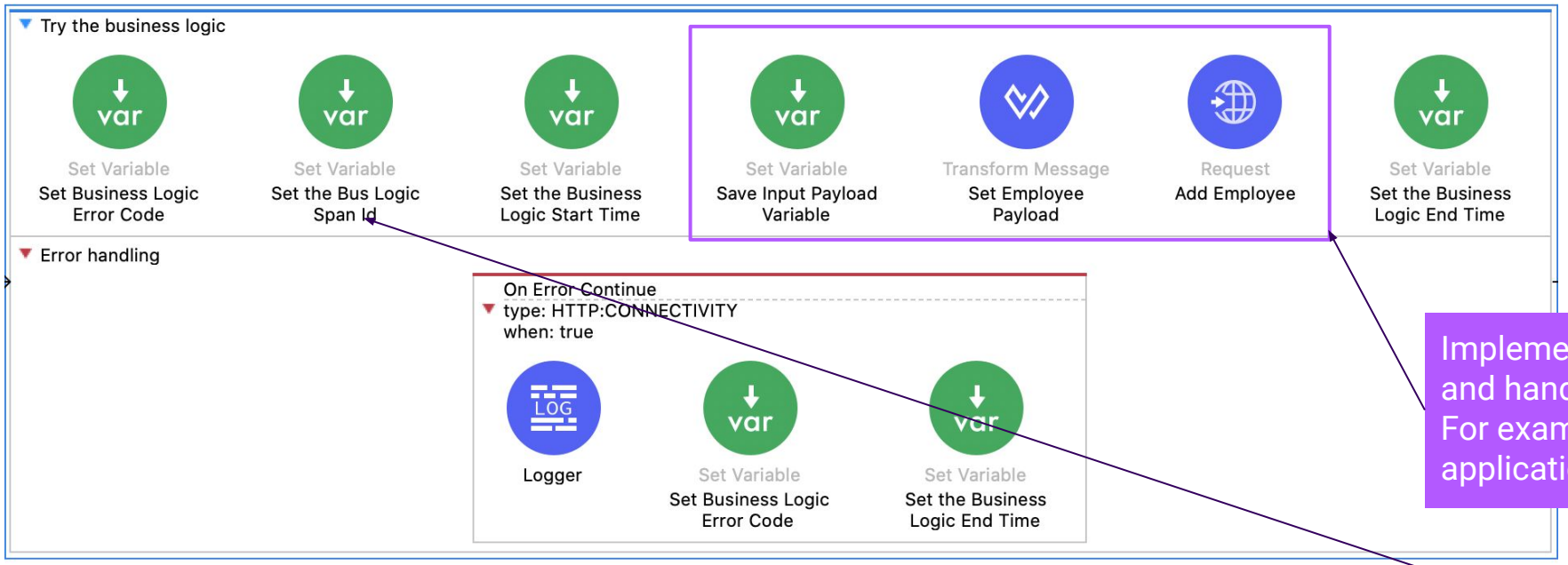
Settings

Name: resource

Value: `#[ attributes.method ++ ' ' ++ attributes.listenerPath`

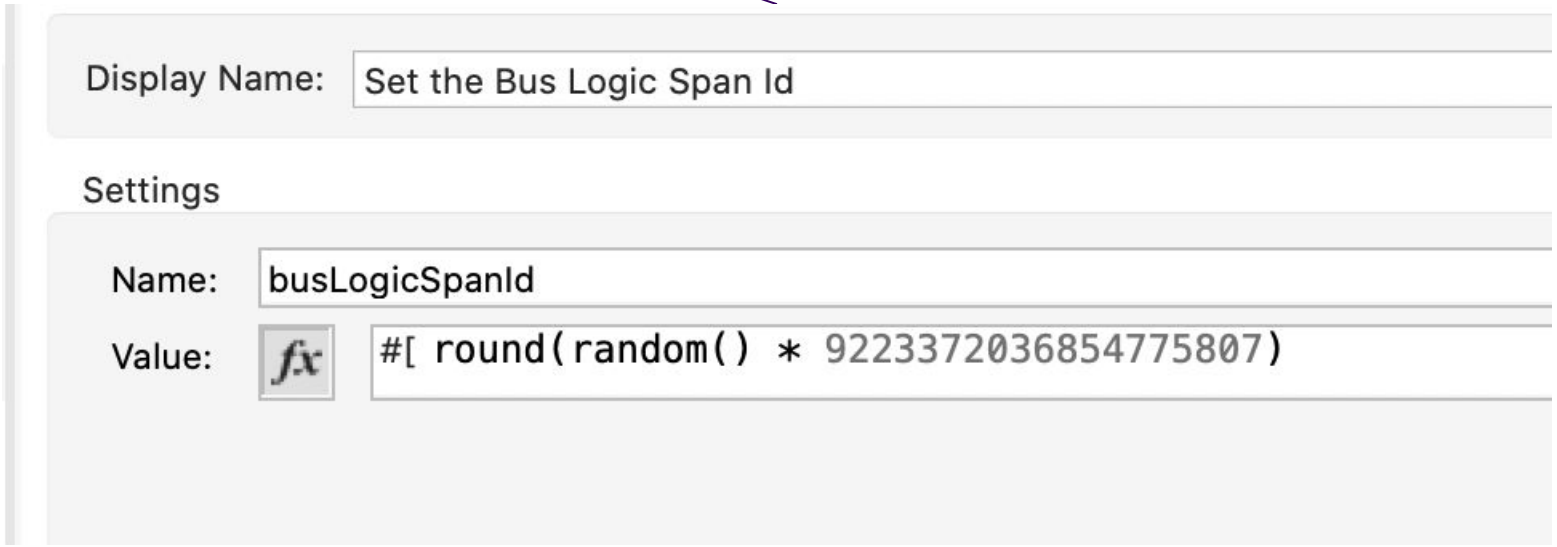
DATADOG

Send Traces to Agent Via API



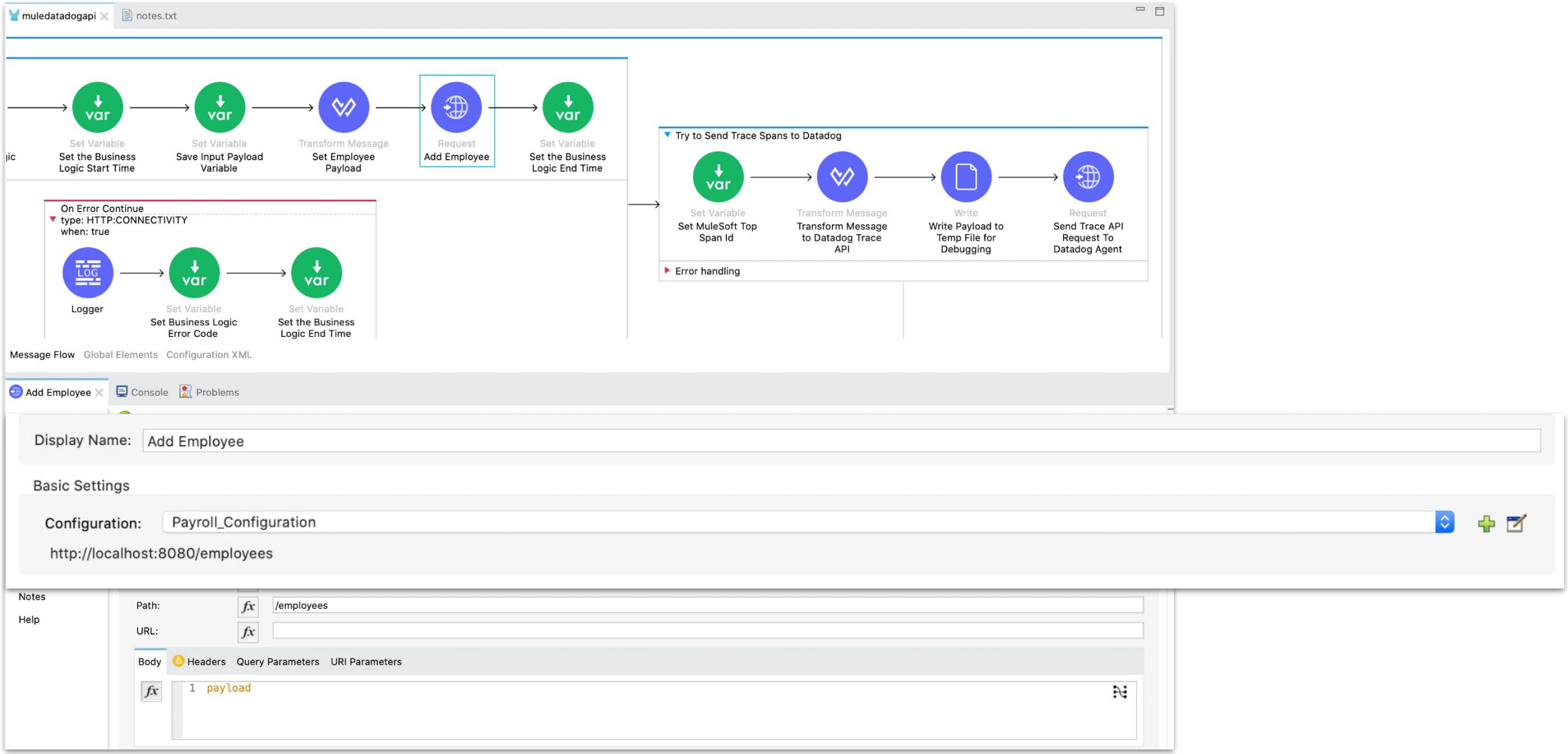
Implement your business logic as usual and handle errors.
For example, calling out to an employee application.

With the business logic, you need to generate a random Datadog Span ID, set the start time and do error handling that you should always do anyway except you will set an error code for the span. Otherwise, do all Business Logic as usual. You do this separately for each major section of business logic that you want to trace.



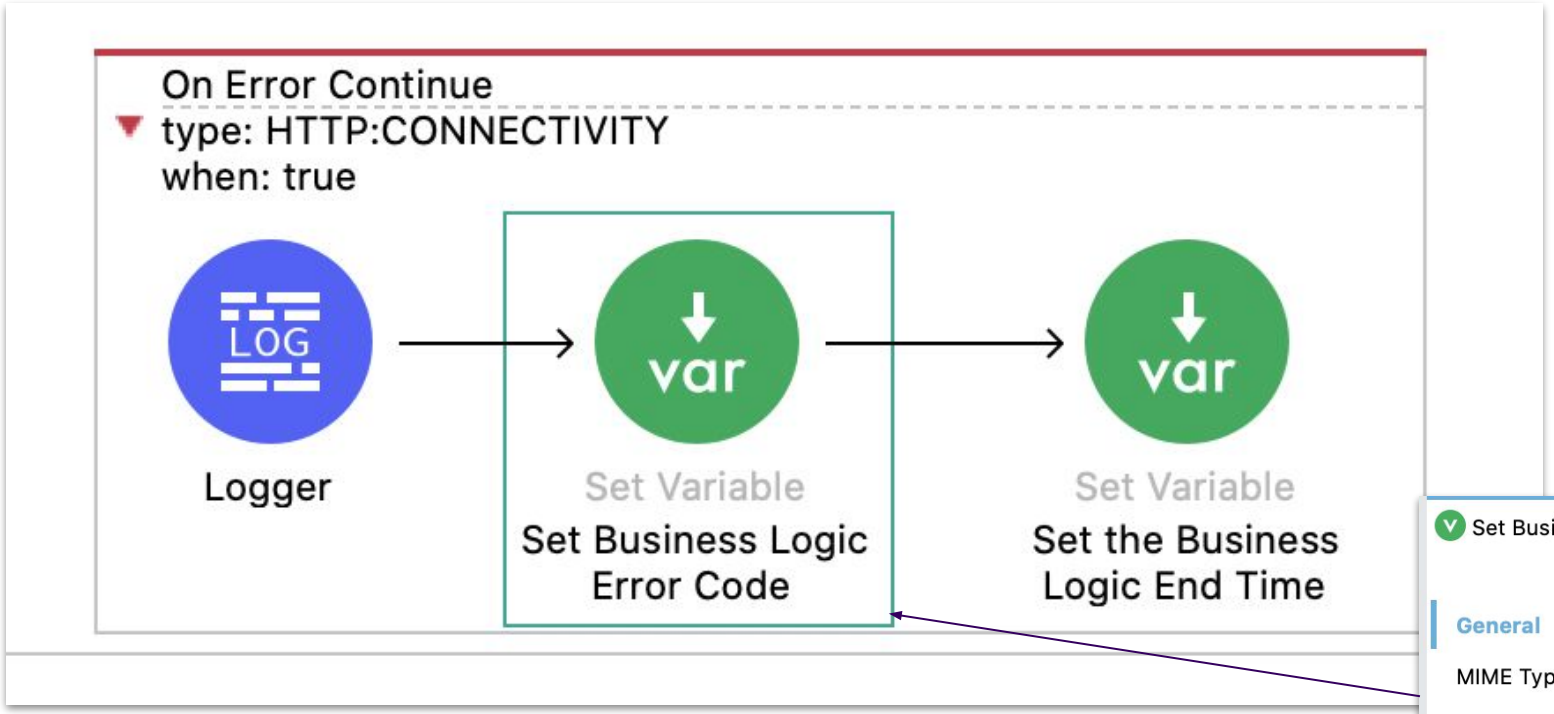
DATADOG

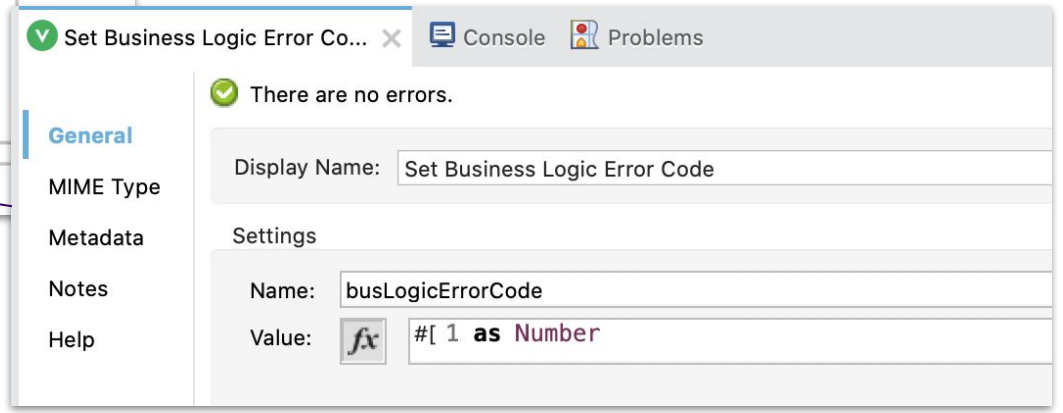
Send Traces to Agent Via API



DATADOG

Send Traces to Agent Via API

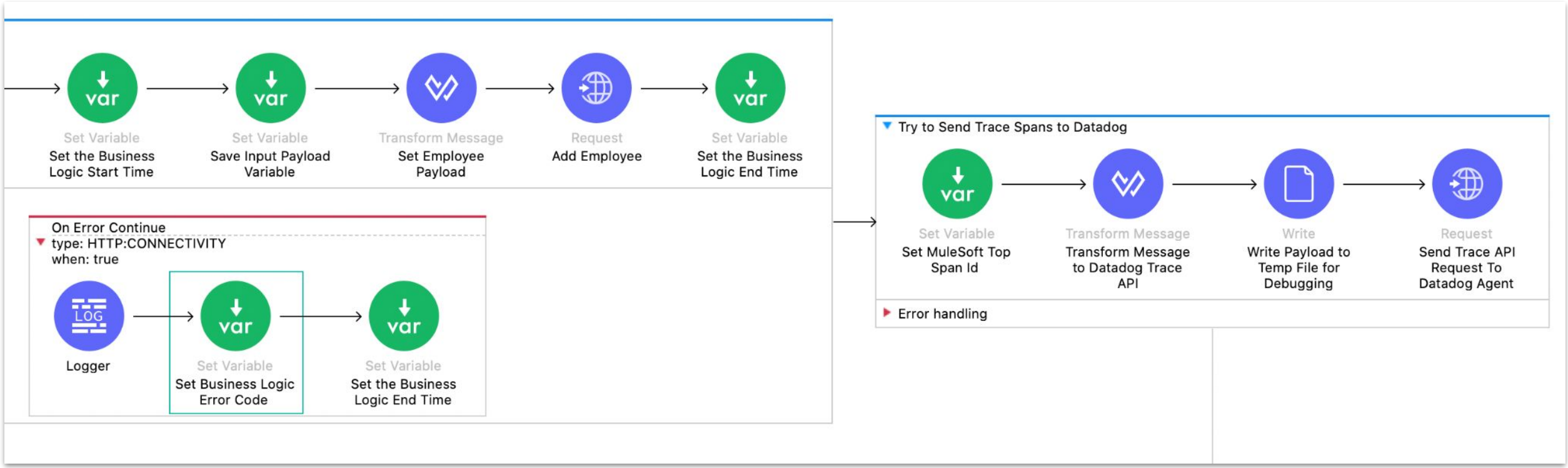




If there is an error, you want to set that in a variable so that it can be passed in the data for that Datadog “Span”. 1 = Error

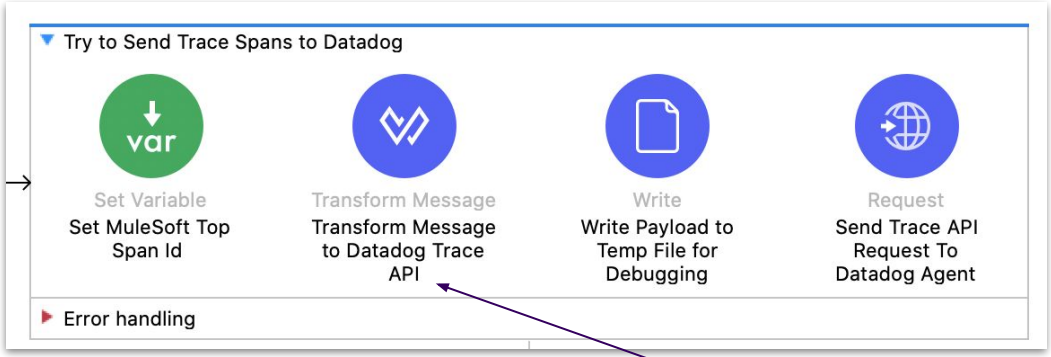
`error`	int32	Set this value to 1 to indicate if an error occurred. If an error occurs, you should pass additional information, such as the error message, type and stack information in the meta property.

Send Traces to Agent Via API



DATADOG

Send Traces to Agent Via API



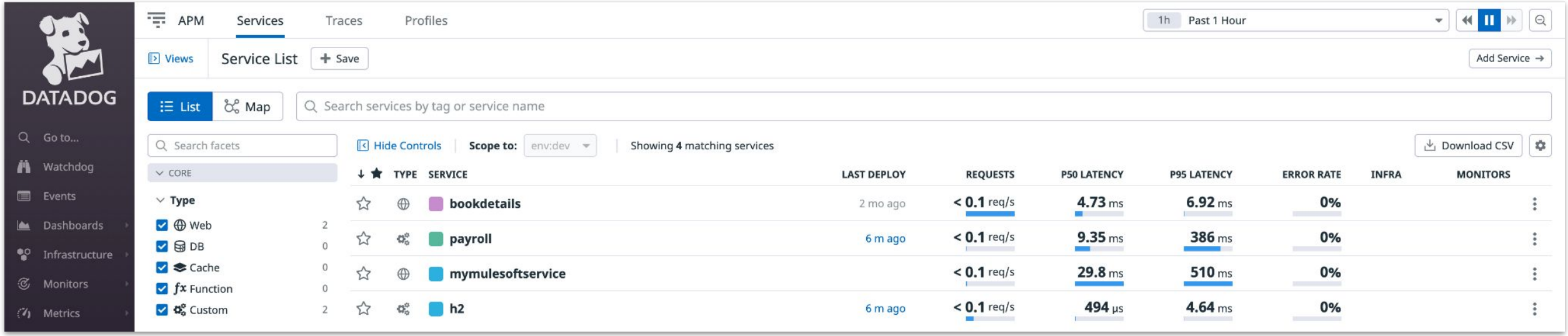
Then put together the trace and spans to send to Datadog. Convert times to nanoseconds and set any other business related metrics that you want to show on the trace.

Output Payload ▾ Preview

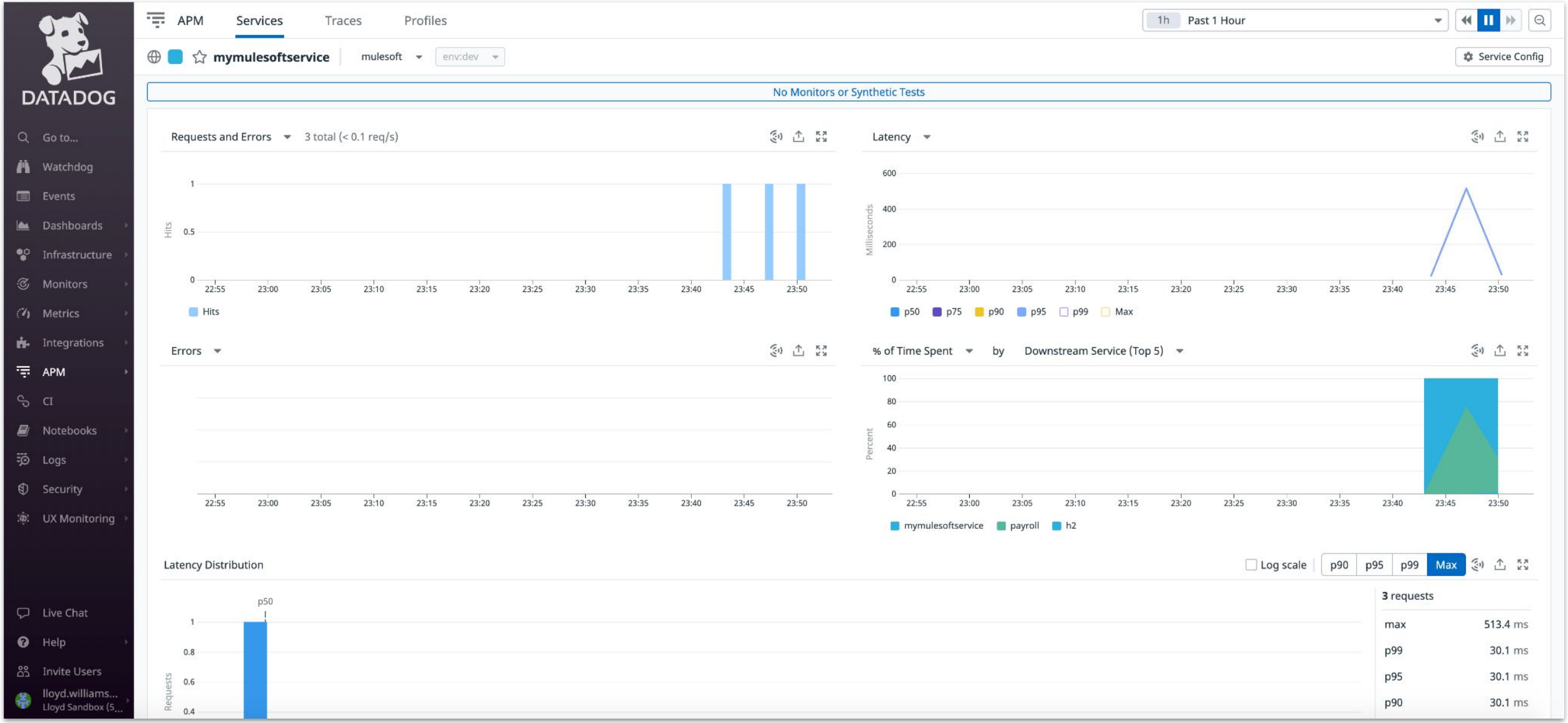
```
%dw 2.0
output application/json
---
[
    [
        {
            "duration": ((now() as Number {unit: "milliseconds"}) - vars.startTime) * 1000000,
            "error": 0,
            "meta": vars.inputPayload,
            "metrics": {
                "cost": 22.05
            },
            "name": "mulesoft",
            "parent_id": vars.parentSpanId as Number,
            "resource": vars.resource,
            "service": "mymulesoftservice",
            "span_id": vars.topSpanId,
            "start": vars.startTime * 1000000,
            "trace_id": vars.traceid as Number,
            "type": "web"
        }
    ],
    [
        {
            "duration": (vars.busLogicEnd - vars.busLogicStart) * 1000000,
            "error": vars.busLogicErrorCode,
            "meta": {
                "more": "my business data"
            },
            "metrics": {
                "dollars": 212.95
            },
            "name": "businesslogic",
            "parent_id": vars.topSpanId,
            "resource": "POST /employees",
            "service": "mymulesoftservice",
            "span_id": vars.busLogicSpanId,
            "start": vars.busLogicStart * 1000000,
            "trace_id": vars.traceid as Number,
            "type": "web"
        }
    ]
]
```

DATADOG

Send Traces to Agent Via API

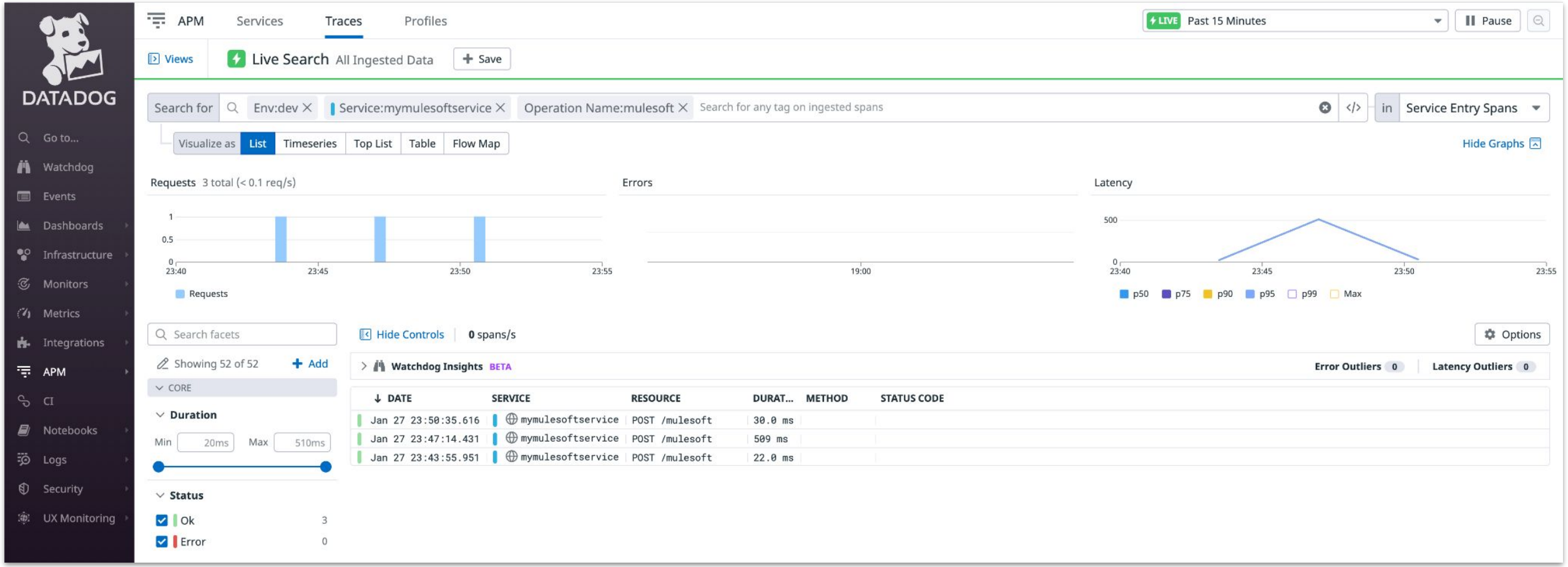


Send Traces to Agent Via API

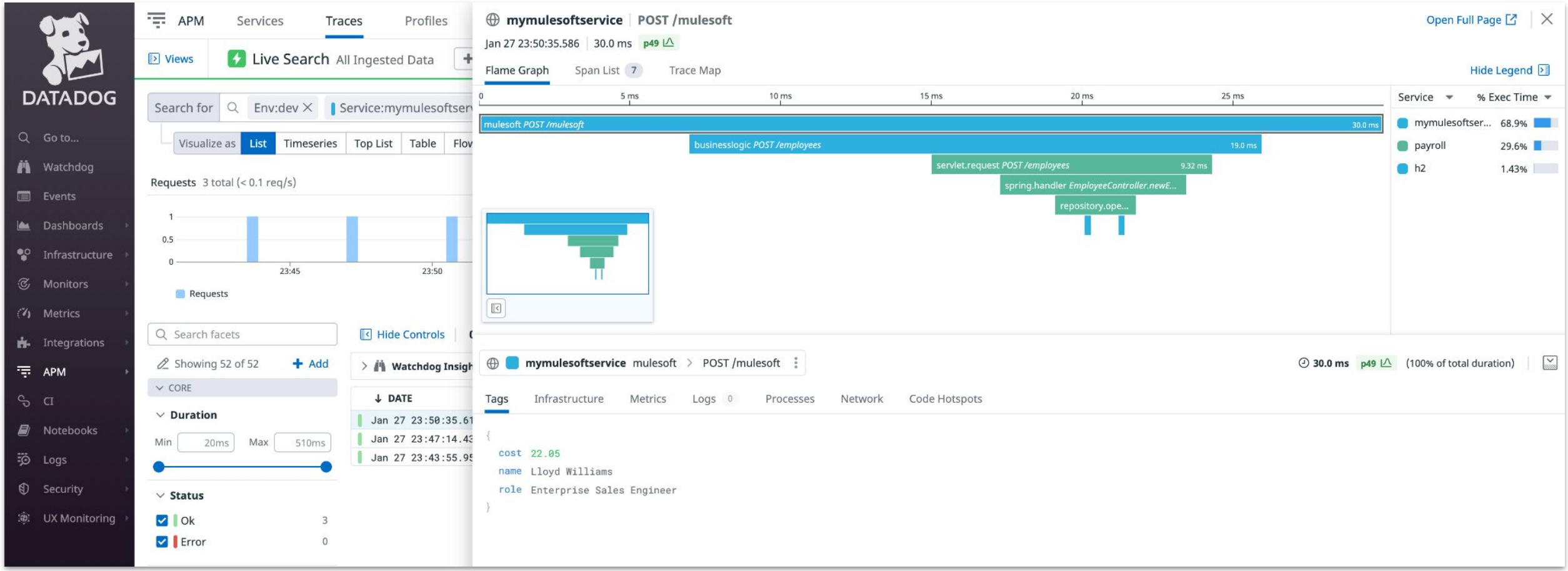


DATADOG

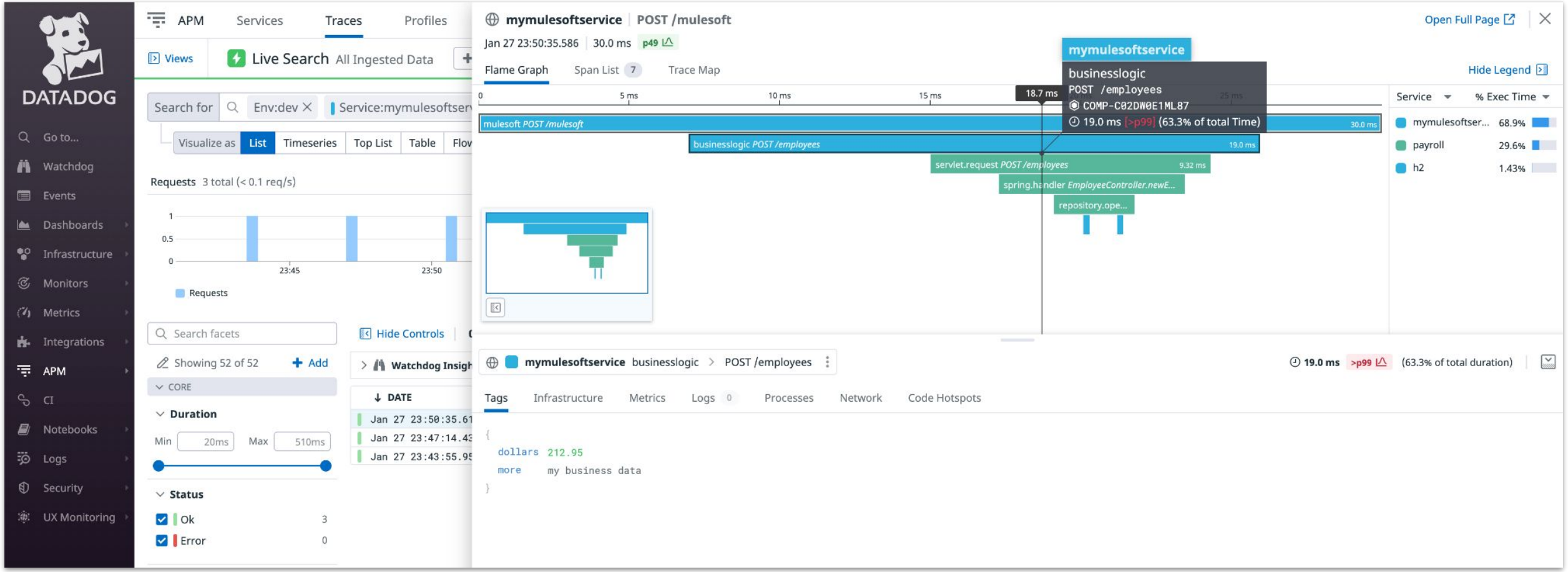
Send Traces to Agent Via API

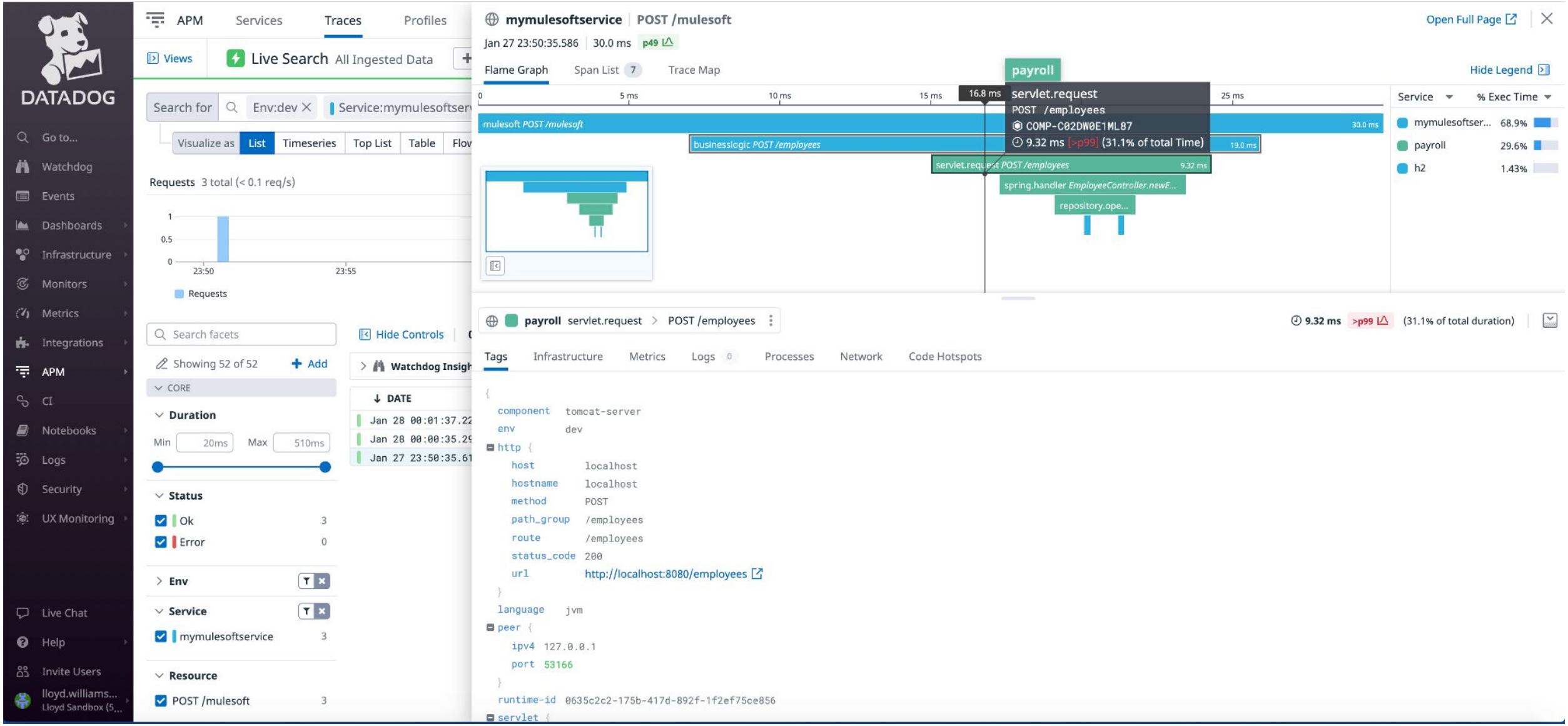


Send Traces to Agent Via API



Send Traces to Agent Via API






```
#!/bin/sh
java -javaagent:/Users/lloyd.williams/u01/datadog/dd-java-agent.jar -Ddd.profiling.enabled=false -Ddd.logs.injection=true -Ddd.service=payroll -Ddd.env=dev -Ddd.version=0.0.2 -jar target/payroll-0.0.2.jar
```

DATADOG

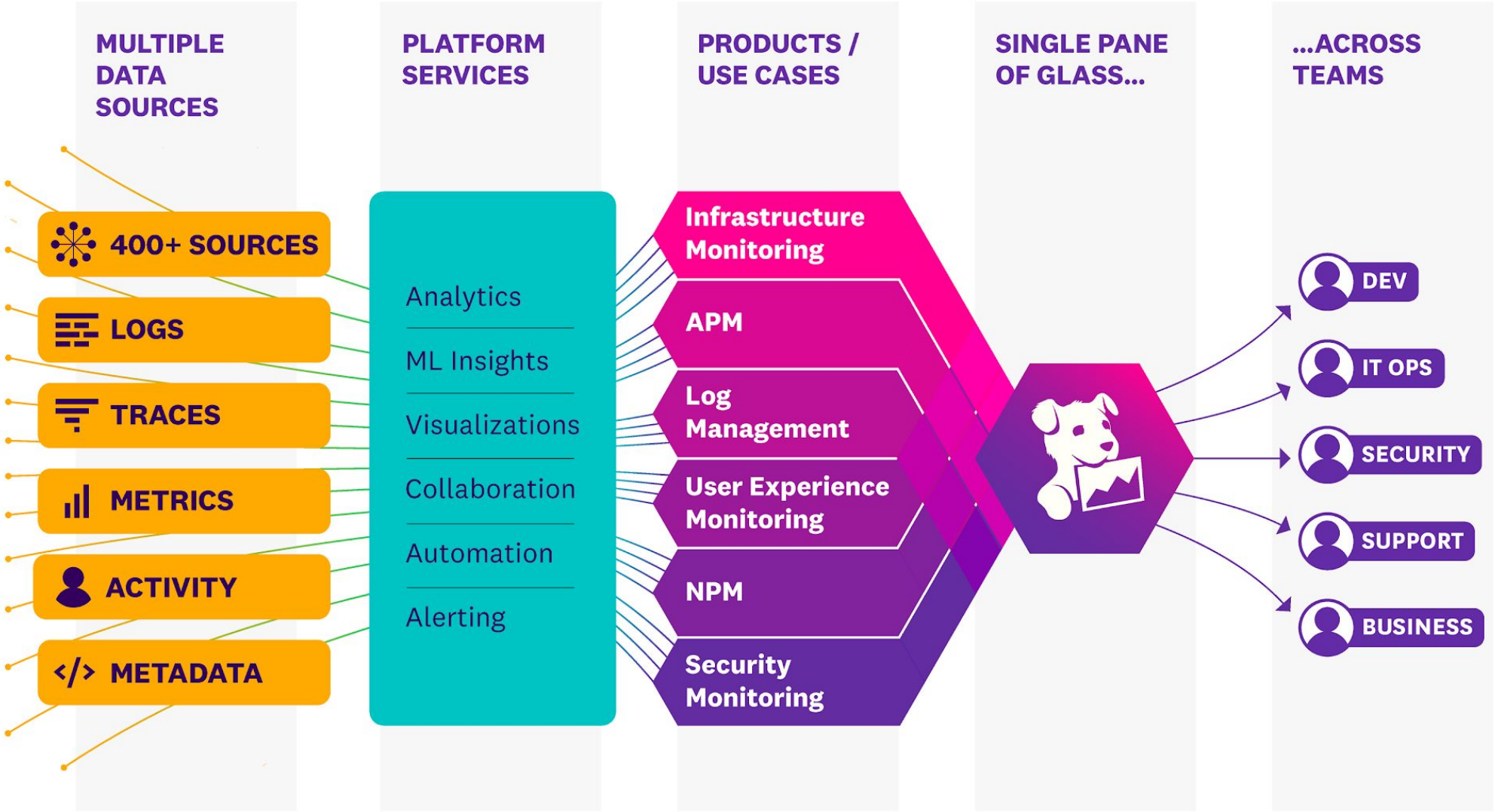
"Payroll" called from MuleSoft is a Java Spring Boot microservice instrumented with the dd-trace-agent that also calls a database.

Datadog Application Performance Monitoring (APM)

- Datadog Application Performance Monitoring (APM) provides end-to-end distributed tracing from frontend devices to databases—with no sampling.
- If you want to sample, you can do it more intelligently with Tracing without Limits™.
- Tagging is built into the core of the Datadog architecture and makes it a platform that uses your language for searching, filtering and correlating different types of monitoring together.
- Datadog is a market leader (Gartner MQ) in Application Performance Monitoring (APM) for tracing requests through distributed applications.
- The Datadog APM SDK allows you to easily add custom metrics to a trace with very little code.
- Follow the User Experience (UX) Real User Monitoring (RUM) session to its related back-end traces.
- Trace through messaging layers like Kafka and view calls (SQL) into databases.

DATADOG

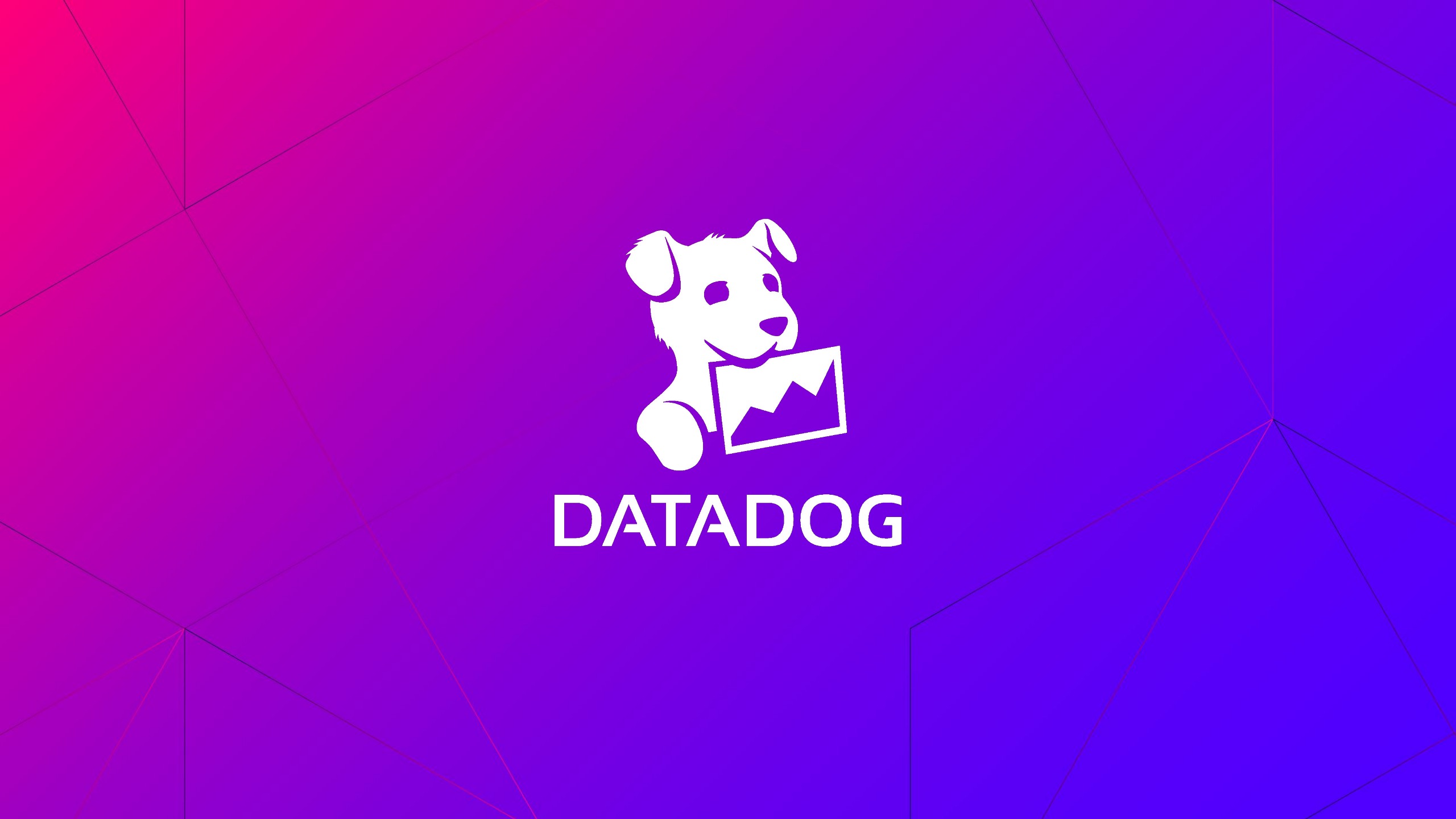
Datadog — Unified SaaS Observability, Monitoring and Analytics Platform



DATADOG

Why Datadog?

- Unified observability - Integrated platform, used by everyone
- Cloud-agnostic - Built for the modern stack, deployed everywhere
- Ease of use - OOTB seamless correlation, no manual querying
- Logging without Limits - ingest all logs cost-effectively & analyze in real-time
- Superior granularity and retention - 15 secs for 15 months, no rollups
- Data-driven, actionable alerting - Reduce alert fatigue with machine learning


DATADOG

Datadog solves complexity