Caution

The SignalFx Instrumentation for .NET reached End of Support on February 21, 2025. The library has been archived and is no longer maintained.

New customers instrumenting the .NET ecosystem should use the Splunk Distribution of OpenTelemetry .NET. Existing customers should consider migrating to Splunk Distribution of OpenTelemetry .NET which offers similar capabilities. To learn how to migrate, see Migrate from the SignalFx .NET Instrumentation.

Connect .NET trace data with logs for Splunk Observability Cloud

You can configure logging libraries to include tracing attributes provided automatically by the SignalFx Instrumentation for .NET. Use the metadata to correlate traces with log events and explore logs in Splunk.

Check compatibility and requirements

The SignalFx Instrumentation supports the following logging libraries:

Library	Versions	Layouts
ILogger	2.5.0 to 6.x.x	• JSON format: `json` from the `NetEscapades.Extensions.Logging` package
Log4Net	1.0.0 to 2.x.x	• JSON format: `SerializedLayout` from the `log4net.Ext.Json` package • Raw format: `PatternLayout`

NLog	1.0.0.505 to 4.x.x	• JSON format: `JsonLayout` • Raw format: Custom layout
Serilog	1.4.0 to 2.x.x	• JSON format: `JsonFormatter` or `CompactJsonFormatter` from the `Serilog.Formatting.Compact` package • Raw format: Output template

Activate log correlation

To activate log correlation, set the `SIGNALFX_LOGS_INJECTION` environment variable to `true` before running your instrumented application.

Include trace metadata in log statements

The SignalFx Instrumentation for .NET provides the following attributes for logging libraries:

- `trace_id`
- `span_id`
- `service.name`, as defined by the `SIGNALFX_SERVICE_NAME` environment variable
- `service.version`, as defined by the `SIGNALFX_VERSION` environment variable
- `deployment.environment`, as defined by the `SIGNALFX_ENV` environment variable

If the logging library uses JSON, the tracing library automatically handles data injection.

If the logger uses a raw format, you must configure your logger to include the trace metadata. The following sections show how to configure the supported loggers to include trace metadata.

Log4Net

When using the `SerializedLayout` layout you can add all contextual properties by adding the properties member. For example:

```
<layout type='log4net.Layout.SerializedLayout, log4net.Ext.Json'>
  <!-- existing configuration -->
  <member value='properties'/> <!-- addition -->
```

```
</layout>
```

You can also add the context fields explicitly. For example:

```
<layout type='log4net.Layout.SerializedLayout, log4net.Ext.Json'>
  <!-- existing configuration -->
  <member value='trace_id' />
  <member value='span_id' />
  <member value='service.name' />
  <member value='service.version' />
  <member value='deployment.environment' />
</layout>
```

When using the `PatternLayout` layout, add the context fields manually. You must wrap the values in quotation marks. For example:

```
<layout type="log4net.Layout.PatternLayout">
  <conversionPattern value="%date [%thread] %level %logger {trace_id=&quot;%property{trace_id}&quot;,
span_id=&quot;%property{span_id}&quot;, service.name=&quot;%property{service.name}&quot;,
service.version=&quot;%property{service.version}&quot;,
deployment.environment=&quot;%property{deployment.environment}&quot;} - %message%newline" />
</layout>
```

NLog

When using the `JsonLayout` layout and NLog version 4.4.10 and higher, you can add all contextual properties by setting the `includeMdlc` attribute to `true`. For example:

```
<layout xsi:type="JsonLayout" includeMdlc="true"> <!-- includeMdlc property available in NLog 4.4.10+ -->
  <!-- existing configuration -->
</layout>
```

You can also add the context fields explicitly. For example:

```
<layout xsi:type="JsonLayout">
  <!-- existing configuration -->
  <attribute name="trace_id" layout="${mdc:item=trace_id}"/>
  <attribute name="span_id" layout="${mdc:item=span_id}"/>
  <attribute name="service.name" layout="${mdc:item=service.name}"/>
  <attribute name="service.version" layout="${mdc:item=service.version}"/>
  <attribute name="deployment.environment" layout="${mdc:item=deployment.environment}"/>
</layout>
```

When using the custom layout, add the context fields manually. Values must be wrapped in quotation marks. For example:

```
<target
// existing configuration
layout="${longdate}|${uppercase:${level}}|${logger}|{trace_id=&quot;${mdc:item=trace_id}&quot;,span_id=&quot;${mdc:item=span_id}&quot;,service.name=&quot;${mdc:item=service.name}&quot;,service.version=&quot;${mdc:item=service.version}&quot;,deployment.environment=&quot;${mdc:item=deployment.environment}&quot;}|${message}"
/>
```

Serilog

To extract the trace context that you want to inject, enrich the `LoggerConfiguration` instance using the log context:

```
var loggerConfiguration = new LoggerConfiguration()
   .Enrich.FromLogContext() // addition
```

When using the output template, you can either use the `{Properties}` placeholder to print all contextual properties or add context fields manually.

When adding context fields manually, wrap the values in quotation marks. For example:

```
"{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} [{Level:u3}] trace_id=\"{trace_id}\" span_id=\"{span_id}\"
service.name=\"{service_name}\" service.version=\"{service_version}\"
deployment.environment=\"{deployment_environment}\"{NewLine}{Message:lj}{NewLine}{Exception}"
```

The instrumentation uses the underscore character as separator for field names (`_`), as Serilog doesn't support property names that use the dot separator (`.`). To ingest log data, define the following conversion rules:

- `service_name` to `service.name`
- `service_version` to `service.version`
- `deployment_environment` to `deployment.environment`

ILogger

When using the `NetEscapades.Extensions.Logging.RollingFile` package, activate the `IncludeScopes` option and use the `json` formatter. For example:

```
Host.ConfigureLogging(builder =>
  builder.AddFile(opts =>
  {
    opts.FileName = "logs";
    opts.Extension = "json";
    opts.FormatterName = "json"; // supported formatter
    opts.IncludeScopes = true; // addition
  })
);
```

Note

SignalFx Instrumentation for .NET only supports ILogger 2.5.0 or higher.

Log correlation also works when ILogger is wrapping other supported loggers.

Sample applications

To download several sample applications that show how to configure log correlation, see https://github.com/signalfx/signalfx-dotnet-tracing/tree/main/tracer/samples/AutomaticTraceIdInjection on GitHub.