
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 The library has been archived and is no longer maintained.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.

Troubleshoot .NET instrumentation for Splunk Observability Cloud

When you instrument a .NET application using the SignalFx Instrumentation for .NET and you don't see your data in Splunk Observability Cloud, follow these troubleshooting steps.

General troubleshooting

Follow these steps to troubleshoot general instrumentation issues:

1. Check that you've configured all settings according to your needs. See Configure the SignalFx Instrumentation for .NET.
2. Check what environment variables apply to your process using tools such as Process Explorer. On Linux, run `cat /proc/<pid>/environ` where `<pid>` is the process ID.

3. Make sure that all environment variables are configured use the following commands:

Windows (PowerShell)

 # Run a tool like Process Explorer or execute the following:

[System.Diagnostics.Process]::GetProcessById(<pid>).StartInfo.EnvironmentVariables

Linux

cat /proc/<pid>/environ *# where <pid> is the process ID*

Activate debug logging

The SignalFx Instrumentation for .NET logs its configuration using `INF` log messages at startup.

You can activate debug logging to obtain more information about the issue:

1. Set the `SIGNALFX_TRACE_DEBUG` environment variable to `true` before starting your instrumented application.
2. Run your application or service and generate some activity.
3. Collect the debug logs. By default, log files are in the following locations:
    - Windows: `%ProgramData%\SignalFx .NET Tracing\logs\`
    - Linux: `/var/log/signalfx/dotnet/`. If it doesn't exist, run `/opt/signalfx/createLogPath.sh`.

You can change the default location by updating the `SIGNALFX_TRACE_LOG_DIRECTORY` environment variable. See Diagnostic logging settings for more information and settings.

Note

If you've enabled debug logging and restarted the application, but you don't find any logs in the previous locations, check that other APM agents aren't running or aren't installed on the host. Multiple APM agents might prevent the SignalFx Instrumentation for .NET from instrumenting the application.

Caution

Activate debug logging only when needed. Debug mode requires more resources.

Traces don't appear in Splunk Observability Cloud

If traces from your instrumented application or service are not available in Splunk Observability Cloud, verify the OpenTelemetry Collector configuration:

- Make sure that the Splunk Distribution of OpenTelemetry Collector is running.
- Make sure that a `zipkin` receiver and an `otlp` exporter are configured.
- Make sure that the `access_token` and `endpoint` fields are configured.
- Check that the traces pipeline is configured to use the `zipkin` receiver and `otlp` exporter.

Metrics don’t appear in Splunk Observability Cloud

If metrics from your instrumented application or service are not available in Splunk Observability Cloud, make sure that the following conditions are true:

- The Splunk Distribution of OpenTelemetry Collector is running.
- A `signalfx` receiver and a `signalfx` exporter are configured.
- The `access_token` and `realm` fields are configured.
- The metrics pipeline is configured to use the `signalfx` receiver and `signalfx` exporter.

.NET instrumentation not working on Linux

Installing the instrumentation on Linux might fail if you use an incompatible package.

Make sure that you’re using an installation package that is compatible with your Linux distribution. To find out your distribution or package manager, run the following commands:

```
lsb_release -a
cat /etc/*release
cat /etc/issue*
cat /proc/version
```

High CPU usage

By default, the SignalFx Instrumentation for .NET instruments all .NET processes running on the host automatically. This might significantly increase CPU usage if you’ve activated the instrumentation in the system or user scope. Make sure that the instrumentation’s environment variables are always set in the process or terminal scope.

To restrict global instrumentation to a set of processes, use the `SIGNALFX_PROFILER_PROCESSES` and `SIGNALFX_PROFILER_EXCLUDE_PROCESSES` environment variables, which include and exclude processes for instrumentation. See Configure the SignalFx Instrumentation for .NET for more information.

Troubleshoot AlwaysOn Profiling for .NET

See the following common issues and fixes for AlwaysOn Profiling:

Check that AlwaysOn Profiling is activated

The .NET instrumentation logs the string `AlwaysOnProfiler::MemoryProfiling` started at `info` log level. To check whether AlwaysOn Profiling is activated, search your logs for strings similar to the following:

```
10/12/22 12:10:31.962 PM [12096|22036] [info] AlwaysOnProfiler::MemoryProfiling started.
```

If no string appears, make sure that you've activated the profiler by setting the `SIGNALFX_PROFILER_ENABLED` environment variable to `true`. See .NET settings for AlwaysOn Profiling.

If you've activated the CPU profiler or the memory profiler on an unsupported runtime version, entries similar to the following entry appear in the logs:

```
2022-10-12 12:37:18.640 +02:00 [WRN] Cpu profiling activated but not supported.
2022-10-12 12:37:18.640 +02:00 [WRN] Memory profiling activated but not supported.
```

Check the AlwaysOn Profiling configuration

If AlwaysOn Profiling is not working as intended, check the configuration settings. The .NET instrumentation logs AlwaysOn Profiling settings using INF messages at startup. Search for the string `TRACER CONFIGURATION`.

Unsupported .NET version

To use AlwaysOn Profiling, upgrade your .NET version to .NET Core 3.1 or .NET 5.0 and higher. Memory profiling requires .NET 5.0 and higher, as `ICorProfilerInfo10` must be available in the runtime.

None of the .NET Framework versions is supported.

AlwaysOn Profiling data and logs don't appear in Splunk Observability Cloud

Collector configuration issues might prevent AlwaysOn Profiling data and logs from appearing in Splunk Observability Cloud.

To solve this issue, do the following:

1. Check the configuration of the SignalFx Instrumentation for .NET, especially `SIGNALFX_PROFILER_LOGS_ENDPOINT`.
2. Verify that the Splunk Distribution of OpenTelemetry Collector is running at the expected endpoint and that the application host or container can resolve the host name and connect to the OTLP port.
3. Make sure that you're running the Splunk Distribution of OpenTelemetry Collector and that the version is 0.34 or higher. The required version for memory profiling is 0.44. Other collector distributions might not be able to route the log data that contains profiling data.
4. A custom configuration might override settings that let the collector handle profiling data. Make sure to configure an `otlp` receiver and a `splunk_hec` exporter with correct token and endpoint fields. The `profiling` pipeline must use the OTLP receiver and Splunk HEC exporter you've configured. See Splunk HEC exporter for more information.

The following snippet contains a sample `profiling` pipeline:

```
receivers:
  otlp:
    protocols:
      grpc:

exporters:
  # Profiling
  splunk_hec/profiling:
    token: "${SPLUNK_ACCESS_TOKEN}"
    endpoint: "${SPLUNK_INGEST_URL}/v1/log"
    log_data_enabled: false

processors:
  batch:
  memory_limiter:
    check_interval: 2s
    limit_mib: ${SPLUNK_MEMORY_LIMIT_MIB}

service:
  pipelines:
    logs/profiling:
      receivers: [otlp]
      processors: [memory_limiter, batch]
      exporters: [splunk_hec, splunk_hec/profiling]
```

Loss of profiling data or gaps in profiling data

When the instrumentation can't send data to Splunk OpenTelemetry Collector due to full buffers, AlwaysOn Profiling activates the escape hatch, which drops all logs with profiling data until the buffers are empty.

If the escape hatch activates, it logs the following message:

Skipping a thread sample period, buffers are full.

You can also look for the `** THIS WILL RESULT IN LOSS OF PROFILING DATA **.` message.

The thread sampler resumes its activity when any of the buffers is empty.

To avoid the loss of profiling data due to full buffers, check the configuration and the communication layer between your process and the Splunk Distribution of OpenTelemetry Collector.

Uninstall the SignalFx Instrumentation for .NET

To remove the SignalFx Instrumentation for .NET, follow the instructions for each operating system.

Windows

Follow these steps to remove the SignalFx Instrumentation for .NET:

1. Stop all instrumented services or applications.
2. Remove all environment variables you might have set for the instrumentation.
3. Uninstall **SignalFx .NET Tracing** from the **Programs and Features** control panel.

Linux

Follow these steps to remove the SignalFx Instrumentation for .NET:

1. Stop all instrumented services or applications.
2. Remove all environment variables you might have set for the instrumentation.
3. Remove `signalfx-dotnet-tracing` using your package manager or delete the files from `/opt/signalfx` if you installed the instrumentation using the tar file.

If you are a Splunk Observability Cloud customer and are not able to see your data in Splunk Observability Cloud, you can get help in the following ways.

Available to Splunk Observability Cloud customers

- Submit a case in the Splunk Support Portal .
- Contact Splunk Support .

Available to prospective customers and free trial users

- Ask a question and get answers through community support at Splunk Answers .
- Join the Splunk #observability user group Slack channel to communicate with customers, partners, and Splunk employees worldwide. To join, see Chat groups in the *Get Started with Splunk Community* manual.