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 The library has been archived and is no longer maintained.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

.NET application for Splunk Observability Cloud

The SignalFx Instrumentation for .NET automatically instruments .NET applications, Windows services running .NET applications, ASP.NET applications deployed on IIS, and Azure App Service applications.

To get started, follow the instructions to install the SignalFx Instrumentation for .NET manually.

Install the SignalFx Instrumentation for .NET manually

If you don't use the guided setup, follow these instructions to manually install the SignalFx Instrumentation for .NET:

- Instrument your .NET application
- Instrument your Windows service running a .NET application
- Instrument your ASP.NET application deployed on IIS
- Instrument your application in Azure App Service
- Instrument your background task in Azure App Service

Instrument your .NET application

Follow these steps to automatically instrument your application:

- Check that you meet the requirements. See .NET instrumentation compatibility and requirements.

- Download the latest release of the SignalFx Instrumentation for .NET for your operating system from the Releases page on GitHub .

3. Install the package for your operating system:

Windows (PowerShell)

```
msiexec /i signalfx-dotnet-tracing-<version-here>-x64.msi /quiet
```

Windows x64Windows x86

```
msiexec /i signalfx-dotnet-tracing-<version-here>-x86.msi /quiet
```

Linux rpm

```
rpm -ivh signalfx-dotnet-tracing-<version-here>.rpm
./opt/signalfx/createLogPath.sh # Optional
```

Linux deb

```
dpkg -i signalfx-dotnet-tracing-<version-here>.deb
./opt/signalfx/createLogPath.sh # Optional
```

Linux tar(glibc)

```
tar -xf signalfx-dotnet-tracing-<version-here>.tar.gz -C /opt/signalfx
./opt/signalfx/createLogPath.sh # Optional
```

4. Set the following environment variables:

Windows (PowerShell)

```
# Set the following variables in the process scope


$Env:COR_ENABLE_PROFILING = "1"
$Env:COR_PROFILER = "{B4C89B0F-9908-4F73-9F59-0D77C5A06874}"
$Env:CORECLR_ENABLE_PROFILING = "1"
$Env:CORECLR_PROFILER = "{B4C89B0F-9908-4F73-9F59-0D77C5A06874}"
$Env:SIGNALFX_SERVICE_NAME = "<my-service-name>"
$Env:SIGNALFX_ENV = "<your-environment>"
```

- Avoid setting the environment variables in the system or user scopes in Windows unless you require permanent autoinstrumentation. See Configure the SignalFx Instrumentation for .NET for more information on how to include or exclude processes for autoinstrumentation.

Linux

```
export CORECLR_ENABLE_PROFILING="1"

export CORECLR_PROFILER="{B4C89B0F-9908-4F73-9F59-0D77C5A06874}"

export CORECLR_PROFILER_PATH="/opt/signalfx/SignalFx.Tracing.ClrProfiler.Native.so"
```

```
export SIGNALFX_DOTNET_TRACER_HOME="/opt/signalfx"

export SIGNALFX_SERVICE_NAME="<my-service-name>"

export SIGNALFX_ENV="<your-environment>"
```

5. (Optional) To activate automatic metric collection, see Activate metrics collection.

6. Run your application.

If no data appears in APM, see Troubleshoot .NET instrumentation for Splunk Observability Cloud.

If you need to add custom attributes to spans or want to manually generate spans, instrument your .NET application or service manually. See Manually instrument .NET applications for Splunk Observability Cloud.

Activate AlwaysOn Profiling

To activate AlwaysOn Profiling, set the `SIGNALFX_PROFILER_ENABLED` environment variable to `true`.

To activate memory profiling, set the `SIGNALFX_PROFILER_MEMORY_ENABLED` environment variable to `true` after activating AlwaysOn Profiling.

See Get data into Splunk APM AlwaysOn Profiling for more information. For more settings, see .NET settings for AlwaysOn Profiling.

Activate metrics collection

To activate automatic metric collection, set the `SIGNALFX_TRACE_METRICS_ENABLED` environment variable to `true`.

To activate runtime metrics, set the `SIGNALFX_RUNTIME_METRICS_ENABLED` environment variable to `true`.

See Metrics collected by the SignalFx Instrumentation for .NET for more information about the metrics collected by the instrumentation. For more metric settings, see Metrics settings.

Note

Runtime metrics are always collected if AlwaysOn Profiling is activated.

Instrument your Windows service running a .NET application

To instrument a Windows service, install the instrumentation and set the following environment variables:

```
$svcName = "MySrv"    # Name of the Windows service you want to instrument
[string[]] $vars = @(
  "COR_ENABLE_PROFILING=1",                          # Activate .NET Framework Profiler
  "COR_PROFILER={B4C89B0F-9908-4F73-9F59-0D77C5A06874}",     # Select .NET Framework Profiler
  "CORECLR_ENABLE_PROFILING=1",                      # Activate .NET (Core) Profiler
  "CORECLR_PROFILER={B4C89B0F-9908-4F73-9F59-0D77C5A06874}", # Select .NET (Core) Profiler
  "SIGNALFX_SERVICE_NAME=<my-service-name>",         # Set service name
  "SIGNALFX_ENV=<environment-name>"                  # Set environment name
)
Set-ItemProperty HKLM:SYSTEM\CurrentControlSet\Services\$svcName -Name Environment -Value $vars
# Every time you start the service, it will be auto-instrumented.
```

For more information on the default service name, see Changing the default service name.

Instrument your ASP.NET application deployed on IIS

To instrument an ASP.NET application running on IIS, install the instrumentation and edit the web.config file to add the following settings. See Configuration methods for more information.

ASP.NET 4.x and higher

Add the following settings inside the `<appSettings>` block of your web.config file:

```
<add key="SIGNALFX_SERVICE_NAME" value="service-name" />
<add key="SIGNALFX_ENV" value="environment-name" />
```

After applying the changes to the web.config file, restart IIS by running the following command:

```
Start-Process "iisreset.exe" -NoNewWindow -Wait
```

In some cases, you might have to restart the machine.

ASP.NET Core

Add the following settings inside the `<aspNetCore>` block of your web.config file:

```
<environmentVariables>
  <environmentVariable name="CORECLR_ENABLE_PROFILING" value="1" />
```

```
  <environmentVariable name="CORECLR_PROFILER" value="{B4C89B0F-9908-4F73-9F59-0D77C5A06874}" />
  <environmentVariable name="SIGNALFX_SERVICE_NAME" value="service-name" />
  <environmentVariable name="SIGNALFX_ENV" value="environment-name" />
</environmentVariables>
```

After applying the changes to the web.config file, restart IIS by running the following command:

```
Start-Process "iisreset.exe" -NoNewWindow -Wait
```

In some cases, you might have to restart the machine.

Note

The ASP.NET Core instrumentation collects and obfuscates query strings by default. See Query string settings for more information.

Note

By default, the installer activates IIS instrumentation for .NET Framework by setting the `Environment` registry key for W3SVC and WAS services located in the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services` folder.

Instrument your application in Azure App Service

To instrument an application or service in Azure App Service, follow these steps:

1. Find and install the **SignalFx .NET Tracing** extension in your application. See Adding Extensions to Web Apps in Azure App Service in the Azure documentation for more information.

2. Add the following application settings. See Configure Apps in the Azure documentation for more information.

Name	Value

`SIGNALFX_ACCESS_TOKEN`	Your Splunk access token. To obtain an access token, see Retrieve and manage user API access tokens using Splunk Observability Cloud.
`SIGNALFX_REALM`	`realm` is the Splunk Observability Cloud realm, for example, `us0`. To find your Splunk realm, see Note about realms.
`SIGNALFX_SERVICE_NAME`	The name of your service or application.
`SIGNALFX_ENV`	The name of your environment where you're instrumenting the application.

3. Restart the application.

Note

To reduce latency and benefit from OTel Collector features, set the endpoint URL to a Collector instance running in Azure VM over an Azure VNet.

Instrument your background task in Azure App Service

When instrumenting an Azure WebJob in App Service, add the following settings. Replace `<extension-version>` in system paths with the version of the .NET instrumentation, for example, `v0.2.0`:

Name	Value
`SIGNALFX_ACCESS_TOKEN`	Your Splunk access token. To obtain an access token, see Retrieve and manage user API access tokens using Splunk Observability Cloud.

`SIGNALFX_REALM`	`realm` is the Splunk Observability Cloud realm, for example, `us0`. To find the realm name of your account, open the navigation menu in Splunk Observability Cloud, select **Settings**, and select your username. The realm name appears in the **Organizations** section.
`SIGNALFX_SERVICE_NAME`	The name of your service or application.
`SIGNALFX_ENV`	The name of your environment where you're instrumenting the application.
`COR_ENABLE_PROFILING`	`1`
`COR_PROFILER`	`{B4C89B0F-9908-4F73-9F59-0D77C5A06874}`
`COR_PROFILER_PATH`	`C:\home\signalfx\tracing\<extension-version>\win-x64\SignalFx.Tracing.ClrProfiler.Native.dll`
`COR_PROFILER_PATH_32`	`C:\home\signalfx\tracing\<extension-version>\win-x86\SignalFx.Tracing.ClrProfiler.Native.dll`
`COR_PROFILER_PATH_64`	`C:\home\signalfx\tracing\<extension-version>\win-x64\SignalFx.Tracing.ClrProfiler.Native.dll`
`CORECLR_ENABLE_PROFILING`	`1`
`CORECLR_PROFILER`	`{B4C89B0F-9908-4F73-9F59-0D77C5A06874}`
`CORECLR_PROFILER_PATH_32`	`C:\home\signalfx\tracing\<extension-version>\win-x86\SignalFx.Tracing.ClrProfiler.Native.dll`
`CORECLR_PROFILER_PATH_64`	`C:\home\signalfx\tracing\<extension-version>\win-x64\SignalFx.Tracing.ClrProfiler.Native.dll`

`SIGNALFX_DOTNET_TRACER_HOME`	`C:\home\signalfx\tracing\<extension-version>`
`SIGNALFX_PROFILER_EXCLUDE_PROCESSES`	`SnapshotUploader.exe;workerforwarder.exe`
`SIGNALFX_TRACE_LOG_PATH`	`C:\home\LogFiles\signalfx\tracing\<extension-version>\dotnet-profiler.log`
`SIGNALFX_AZURE_APP_SERVICES`	`0`

Caution

Set `SIGNALFX_AZURE_APP_SERVICES` to `0` when instrumenting WebJobs. Keep a separate App Service for the WebJob, so that you can use separate settings for your application and for the background service.

Activate AlwaysOn Profiling

AlwaysOn Profiling requires .NET 6.0 or higher.

Limited support is available for the following legacy versions of .NET:

- CPU profiling: .NET Core 3.1 and .NET 5.x
- Memory profiling: .NET Core 5.x

To activate AlwaysOn Profiling, do the following:

- Activate the profiler by setting the `SIGNALFX_PROFILER_ENABLED` environment variable to `true` for your .NET process.
- Activate memory profiling by setting the `SIGNALFX_PROFILER_MEMORY_ENABLED` environment variable to `true`.

For more configuration options, including setting a separate endpoint for profiling data, see .NET settings for AlwaysOn Profiling.

Deploy the .NET instrumentation in Kubernetes

To deploy the .NET instrumentation in Kubernetes, configure the Kubernetes Downward API to expose environment variables to Kubernetes resources.

The following example shows how to update a deployment to expose environment variables by adding the agent configuration under the `.spec.template.spec.containers.env` section:

```
apiVersion: apps/v1
kind: Deployment
spec:
  selector:
    matchLabels:
      app: your-application
  template:
    spec:
      containers:
        - name: myapp
          env:
            - name: SPLUNK_OTEL_AGENT
              valueFrom:
                fieldRef:
                  fieldPath: status.hostIP
            - name: SIGNALFX_ENDPOINT_URL
              value: "http://$(SPLUNK_OTEL_AGENT):9411/api/v2/spans"
            - name: SIGNALFX_SERVICE_NAME
              value: '<name-of-your-service>'
            - name: SIGNALFX_ENV
              value: '<name-of-your-environment>'
            - name: CORECLR_ENABLE_PROFILING
              value: "1"
            - name: CORECLR_PROFILER
              value: '{B4C89B0F-9908-4F73-9F59-0D77C5A06874}'
            - name: CORECLR_PROFILER_PATH
              value: '/opt/signalfx/SignalFx.Tracing.ClrProfiler.Native.so'
            - name: SIGNALFX_DOTNET_TRACER_HOME
              value: '/opt/signalfx'
```

Send data directly to Splunk Observability Cloud

By default, the instrumentation sends all telemetry to the local instance of the Splunk Distribution of OpenTelemetry Collector.

To bypass the OTel Collector and send data directly to Splunk Observability Cloud, set the following environment variables:

Windows PowerShell

```
$env:SIGNALFX_ACCESS_TOKEN=<access_token>
$env:SIGNALFX_REALM=<realm>
```

Linux

```
export SIGNALFX_ACCESS_TOKEN=<access_token>
export SIGNALFX_REALM=<realm>
```

To obtain an access token, see Retrieve and manage user API access tokens using Splunk Observability Cloud.

In the ingest endpoint URL, `realm` is the Splunk Observability Cloud realm, for example, `us0`. To find the realm name of your account, follow these steps:

1. Open the navigation menu in Splunk Observability Cloud.
2. Select **Settings**.
3. Select your username.

The realm name appears in the **Organizations** section.

For more information on the ingest API endpoints, see Send APM traces .

Caution

This procedure applies to spans and traces. To send AlwaysOn Profiling data, you must use the OTel Collector.

Specify the source host

To override the host used by the agent, use the environment variable `OTEL_RESOURCE_ATTRIBUTES` to set your host's name to the desired source:

Windows PowerShell

```
$env:OTEL_RESOURCE_ATTRIBUTES=host.name=<host_name>
```

Linux

```
export OTEL_RESOURCE_ATTRIBUTES=host.name=<host_name>
```