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 The library has been archived and is no longer maintained.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

Configure the SignalFx Instrumentation for .NET

You can configure the SignalFx Instrumentation for .NET to suit your instrumentation needs. In most cases, modifying the basic configuration is enough to get started. More advanced settings are also available.

Configuration methods

You can change the settings of the SignalFx Instrumentation for .NET in the following ways:

1. Set environment variables. On Windows, set them in the process scope unless you want to activate autoinstrumentation globally for all .NET applications.

Edit the web.config or app.config file. For example:

```
<configuration>
  <appSettings>
    <add key="SIGNALFX_SERVICE_NAME" value="my-service-name" />
  </appSettings>
</configuration>
```

2.

Generate a JSON configuration file and set the `SIGNALFX_TRACE_CONFIG_FILE` environment variable to the path of the file. You can define settings as key-value pairs:

```
{
  "SIGNALFX_SERVICE_NAME": "my-service-name"
}
```

3.

Note

Settings defined using environment variables override settings in XML and JSON configuration files.

General settings

The following settings are common to most instrumentation scenarios:

Setting	Description
`SIGNALFX_ENV`	The value for the `deployment.environment` tag added to all spans.
`SIGNALFX_SERVICE_NAME`	The name of the application or service. If not set, the instrumentation looks for a suitable default name. See Changing the default service name.
`SIGNALFX_VERSION`	The version of the application. When set, it adds the `version` tag to all spans.
`SIGNALFX_PROFILER_PROCESSES`	Names of the executable files that the profiler can instrument. Supports multiple semicolon-separated values, for example: `MyApp.exe;dotnet.exe`.
`SIGNALFX_PROFILER_EXCLUDE_PROCESSES`	Names of the executable files that the profiler cannot instrument. Supports multiple semicolon-separated values, for example: `ReservedProcess.exe;powershell.exe`.

`SIGNALFX_TRACE_CONFIG_FILE`	Path of the JSON configuration file. Set this environment variable if you're configuring the instrumentation using a JSON file. See Configuration methods for more information.
`SIGNALFX_TRACE_ENABLED`	Set to `false` to deactivate the tracer. The default value is `true`.
`SIGNALFX_AZURE_APP_SERVICES`	Set to `true` to indicate that the profiler is running in the context of Azure App Services. The default value is `false`.
`SIGNALFX_DOTNET_TRACER_HOME`	Location of the installed instrumentation. Must be set manually to `/opt/signalfx` when instrumenting applications on Linux or background services in Azure App Service. By default, the Windows installer automatically uses the `C:\Program Files\SignalFx\.NET Tracing` directory.

Exporter settings

The following settings control trace exporters and their endpoints:

Setting	Description
`SIGNALFX_ACCESS_TOKEN`	Splunk Observability Cloud access token for your organization. The token activates sending traces directly to the Splunk Observability Cloud ingest endpoint. To obtain an access token, see Retrieve and manage user API access tokens using Splunk Observability Cloud.
`SIGNALFX_REALM`	The name of your organization's realm, for example, `us0`. When you set the realm, metrics are sent to `https://ingest.<realm>.signalfx.com/v2/datapoint`

	and traces are sent to `https://ingest.<realm>.signalfx.com/v2/trace`.
`SIGNALFX_ENDPOINT_URL`	The URL to where the trace exporter sends traces. The default value is `http://localhost:9411/api/v2/spans`. Setting a value overrides the `SIGNALFX_REALM` environment variable.
`SIGNALFX_METRICS_ENDPOINT_URL`	The URL to where the metrics exporter sends metrics. The default value is `http://localhost:9943/v2/datapoint`. Setting a value overrides the `SIGNALFX_REALM` environment variable.
`SIGNALFX_TRACE_PARTIAL_FLUSH_ENABLED`	Activate to export traces that contain a minimum number of closed spans, as defined by `SIGNALFX_TRACE_PARTIAL_FLUSH_MIN_SPANS`. The default value is `false`.
`SIGNALFX_TRACE_PARTIAL_FLUSH_MIN_SPANS`	Minimum number of closed spans in a trace before it's exported. The default value is `500`. Requires the value of the `SIGNALFX_TRACE_PARTIAL_FLUSH_ENABLED` environment variable to be `true`.

Trace propagation settings

The following settings control trace propagation:

Setting	**Description**
`SIGNALFX_PROPAGATORS`	Comma-separated list of propagators for the tracer. The available propagators are `B3` and `W3C`, which correspond to the `b3multi` and `tracecontext` propagators in the OpenTelemetry SDK. The default value is `B3,W3C`.

.NET settings for AlwaysOn Profiling

The following settings control the AlwaysOn Profiling feature for the .NET instrumentation:

Environment variable	Description
`SIGNALFX_PROFILER_ENABLED`	Activates AlwaysOn Profiling. The default value is `false`.
`SIGNALFX_PROFILER_MEMORY_ENABLED`	Activates memory profiling. The default value is `false`.
`SIGNALFX_PROFILER_LOGS_ENDPOINT`	The collector endpoint for profiler logs. The default value is `http://localhost:4318/v1/logs`.
`SIGNALFX_PROFILER_CALL_STACK_INTERVAL`	Frequency with which call stacks are sampled, in milliseconds. The default value is `10000` milliseconds.

Note

For more information on AlwaysOn Profiling, see Introduction to AlwaysOn Profiling for Splunk APM.

Metrics settings

The following settings control metric collection:

Setting	Description
`SIGNALFX_METRICS_{0}_ENABLED`	Configuration pattern for activating or deactivating a specific metrics group. For example, to activate `NetRuntime` metrics, set `SIGNALFX_METRICS_NetRuntime_ENABLED=true`. Supported metrics are `NetRuntime`, `Process`, `AspNetCore`, and `Traces`. The default value is `false`. See Metrics collected by the SignalFx Instrumentation for .NET for more information.

Note

NetRuntime metrics are always collected if memory profiling is activated.

Instrumentation settings

The following settings control instrumentations and tracing behavior:

Setting	Description
`SIGNALFX_GLOBAL_TAGS`	Comma-separated list of key-value pairs that specify global span tags. For example: `key1:val1,key2:val2`.
`SIGNALFX_RECORDED_VALUE_MAX_LENGTH`	Maximum length of the value of an attribute. Values longer than this value are truncated. Values are discarded entirely when set to `0`, and ignored when set to a negative value. The default value is `12000`.
`SIGNALFX_DISABLED_INTEGRATIONS`	Comma-separated list of library instrumentations you want to deactivate. Each value must match an internal instrumentation ID. See Supported libraries for a list of integration identifiers.
`SIGNALFX_TRACE_{0}_ENABLED`	Activates or deactivates a specific instrumentation library. For example, to deactivate the Kafka instrumentation, set `SIGNALFX_TRACE_Kafka_ENABLED` to `false`. The value must match an internal instrumentation ID. See Supported libraries for a list of integration identifiers.

Library-specific instrumentation settings

The following settings control the behavior of specific instrumentations:

Setting	Description

`SIGNALFX_HTTP_CLIENT_ERROR_STATUSES`	Comma-separated list of HTTP client response statuses or ranges for which the spans are set as errors, for example: `300,400-499`. The default value is `400-599`.
`SIGNALFX_HTTP_SERVER_ERROR_STATUSES`	Comma-separated list of HTTP server response statuses or ranges for which the spans are set as errors, for example: `300,400-599`. The default value is `500-599`.
`SIGNALFX_INSTRUMENTATION_ELASTICSEARCH_TAG_QUERIES`	Activates the tagging of a `PostData` command as `db.statement`. It might introduce overhead for direct streaming users. The default value is `true`.
`SIGNALFX_INSTRUMENTATION_MONGODB_TAG_COMMANDS`	Activates the tagging of a `BsonDocument` command as `db.statement`. The default value is `true`.
`SIGNALFX_INSTRUMENTATION_REDIS_TAG_COMMANDS`	Activates the tagging of Redis commands as `db.statement`. The default value is `true`.
`SIGNALFX_TRACE_DELAY_WCF_INSTRUMENTATION_ENABLED`	Activates the updated WCF instrumentation, which delays execution until later in the WCF pipeline when the WCF server exception handling is established. The default value is `false`.
`SIGNALFX_TRACE_HEADER_TAGS`	Comma-separated map of HTTP header keys to tag names, automatically applied as tags on traces. For example: `x-my-header:my-tag,header2:tag2`.

`SIGNALFX_TRACE_HTTP_CLIENT_EXCLUDED_URL_SUBSTRINGS`	Comma-separated list of URL substrings. Matching URLs are ignored by the tracer. For example, `subdomain,xyz,login,download`.
`SIGNALFX_TRACE_KAFKA_CREATE_CONSUMER_SCOPE_ENABLED`	Activate to close consumer scope upon entering a method and starting a new one on method exit. The default value is `true`.
`SIGNALFX_TRACE_ROUTE_TEMPLATE_RESOURCE_NAMES_ENABLED`	Activate to base ASP.NET span and resource names on routing configuration, if applicable. The default value is `true`.

Server trace information

To connect Real User Monitoring (RUM) requests from mobile and web applications with server trace data, trace response headers are activated by default. The instrumentation adds the following response headers to HTTP responses:

```
Access-Control-Expose-Headers: Server-Timing
Server-Timing: traceparent;desc="00-<serverTraceId>-<serverSpanId>-01"
```

The `Server-Timing` header contains the `traceId` and `spanId` parameters in `traceparent` format. W3C tracecontext and W3C baggage context propagation is activated by default. For more information, see the Server-Timing and traceparent documentation on the W3C website.

Note

If you need to deactivate trace response headers, set `SIGNALFX_TRACE_RESPONSE_HEADER_ENABLED` to `false`.

Query string settings

Note

This feature is only available when instrumenting ASP.NET Core applications.

The following settings control the inclusion of query strings in the `http.url` tag for ASP.NET Core instrumented applications.

Setting	Description
`SIGNALFX_HTTP_SERVER_TAG_QUERY_STRING`	Activates or deactivates query string inclusion in the `http.url` tag for ASP.NET Core applications. The default value is `true`.
`SIGNALFX_TRACE_OBFUSCATION_QUERY_STRING_REGEXP`	Custom regular expression to obfuscate query strings. The default value is shown in the example.
`SIGNALFX_TRACE_OBFUSCATION_QUERY_STRING_REGEXP_TIMEOUT`	Timeout to the execution of the query string obfuscation pattern defined in `SIGNALFX_TRACE_OBFUSCATION_QUERY_STRING_REGEXP`, in milliseconds. The default value is `200`.

Obfuscating query string prevents your applications from sending sensitive data to Splunk.

The default regular expression for query obfuscation is the following:

((?i)(?:p(?:ass)?w(?:or)?d|pass(?:_?phrase)?|secret|(?:api_?|private_?|public_?|access_?|secret_?)key(?:_?id)?|token|consumer_?(?:id|key|secret)|sign(?:ed|ature)?|auth(?:entication|orization)?)(?:(?:\s|%20)*(?:=|%3D)[^&]+|(?:"|%22)(?:\s|%20)*(?::|%3A)(?:\s|%20)*(?:"|%22)(?:%2[^2]|%[^2]|[^"%])+(?:"|%22))|bearer(?:\s|%20)+[a-z0-9\._\-]|token(?::|%3A)[a-z0-9]{13}|gh[opsu]_[0-9a-zA-Z]{36}|ey[I-L](?:[\w=-]|%3D)+\.ey[I-L](?:[\w=-]|%3D)+(?:\.(?:[\w.+\/=-]|%3D|%2F|%2B)+)?|[\-]{5}BEGIN(?:[a-z\s]|%20)+PRIVATE(?:\s|%20)KEY[\-]{5}[^\-]+[\-]{5}END(?:[a-z\s]|%20)+PRIVATE(?:\s|%20)KEY|ssh-rsa(?:\s|%20)*(?:[a-z0-9\/\.+]|%2F|%5C|%2B){100,})`

Diagnostic logging settings

The following settings control the internal logging of the SignalFx Instrumentation for .NET:

Setting	Description

`SIGNALFX_DIAGNOSTIC_SOURCE_ENABLED`	Activate to generate troubleshooting logs using the `System.Diagnostics.DiagnosticSource` class. The default value is `true`.
`SIGNALFX_FILE_LOG_ENABLED`	Activates file logging. The default value is `true`.
`SIGNALFX_MAX_LOGFILE_SIZE`	The maximum size for tracer log files, in bytes. The default value is `245760`, or 10 megabytes.
`SIGNALFX_STDOUT_LOG_ENABLED`	Activates `stdout` logging. The default value is `false`.
`SIGNALFX_STDOUT_LOG_TEMPLATE`	Configures the `stdout` log template using the Serilog formatting conventions. The default value is `[{Level:u3}] {Message:lj} {NewLine}{Exception}{NewLine}`.
`SIGNALFX_TRACE_DEBUG`	Activate to activate debugging mode for the tracer. The default value is `false`.
`SIGNALFX_TRACE_LOG_DIRECTORY`	Directory of the .NET tracer logs. Overrides the value in `SIGNALFX_TRACE_LOG_PATH` if present. The default value is `/var/log/signalfx/dotnet/` for Linux and `%ProgramData%\SignalFx .NET Tracing\logs\` for Windows.
`SIGNALFX_TRACE_LOGGING_RATE`	The number of seconds between identical log messages for tracer log files. Setting this environment variable to `0` deactivates rate limiting. The default value is `60`.
`SIGNALFX_TRACE_STARTUP_LOGS`	Activate to activate diagnostic logs at startup. The default value is `true`.

Changing the default service name

By default, the SignalFx Instrumentation for .NET retrieves the service name by trying the following steps until it succeeds:

1. For the SignalFx .NET Tracing Azure Site Extension, the default service name is the site name as defined by the `WEBSITE_SITE_NAME` environment variable.
2. For ASP.NET applications, the default service name is `SiteName[/VirtualPath]`.
3. For other applications, the default service name is the name of the entry assembly. For example, the name of your .NET project file.
4. If the entry assembly is not available, the instrumentation tries to use the current process name. The process name can be `dotnet` if launched directly using an assembly. For example, `dotnet InstrumentedApp.dll`.

If all the steps fail, the service name defaults to `UnknownService`.

To override the default service name, set the `SIGNALFX_SERVICE_NAME` environment variable.