

OpenTelemetry Python Configuration

Authors: Owais Lone <olone@splunk.com>

The OpenTelemetry Python project consists of a number of components such as the SDK, auto-instrumentor/bootstrapper and an ever growing number of auto-instrumentations. Most of these components need to provide the user some way to configure them. At the moment the OpenTelemetry Python project provides a configuration manager that allows users to pass in config options via environment variables. The configuration manager can be found here: https://github.com/open-telemetry/opentelemetry-python/blob/master/opentelemetry-api/src/opentelemetry/configuration/__init__.py

The config manager does not enforce any config schema and accepts any environment variables that are,

1. Prefixed with the word `OTEL\_` or `OTEL\_PYTHON\_`.
2. Only contain alphanumeric and underscore characters.

For example if a user sets an environment variable `OTEL\_SERVICE\_NAME` to `my-awesome-service` then any component can import the configuration manager and read the value as `Configuration.SERVICE\_NAME`. For example,

```
from opentelemetry import configuration
```

```
service_name = configuration.Configuration().service_name
```

This is actually a very simple and efficient system, and works very well in practice. However, there has been some desire to allow a config file based system in addition to or instead of an environment variable based one. This document proposes a simple way to support both at the same time.

I propose to keep existing support for env vars as is while extending the Configuration manager to support config files.

Goals

- Support file based configuration.
- Keep existing support for env var based configuration.
- Users should be able to define configuration through both env vars and config files for the same application.

- 100% backward compatible:
 - Components that use the configuration manager today must not break.
- Transparent to components:
 - Components should not have to care how or where a config value was defined. They should not have to worry about reading env vars, files or any other type of config source.
- Ability to extend and add more sources in future.

Non-Goals

- Have per-component config file/source.
- Make components provide any config schema/validation although components are free to do so.

Proposal

The proposal is to extend the configuration manager to read config values from a file in addition to environment variables. The configuration manager API should remain the same as it is today.

Config format

Config manager can support reading config from any number of config formats. I propose we start with YAML and add more formats if/when needed.

The config manager should read a config file and flatten the document into a flat dictionary. The manager should join nested keys with an underscore to flatten the document.

Example:

Given a config file below

```
service_name: svc-cfg
trace_exporter: opencensus
exporter:
  otlp:
    span_endpoint: localhost:5580
flask:
  distributed_tracing: false
```

And the following environment variables:

```
OTEL_SERVICE_NAME=svc-env  
OTEL_REQUESTS_BLACKLIST_URLS=example.com
```

The config manager would make the following config options available:

```
service_name: svc-cfg  
trace_exporter: otlp  
exporter_olp_span_endpoint: localhost:5580  
flask_distributed_tracing: False  
requests_blacklist_urls: example.com
```

And could be read by components as:

```
from opentelemetry import configuration  
  
service_name = configuration.Configuration().service_name
```

If a config value is found in both sources, the value from the configuration file should take precedence.

Specifying the config file

Config file can be specified as a CLI argument or an environment variable. The following options are equivalent

```
opentelemetry-instrumentation --config config.yaml program.py
```

```
OTEL_CONFIG_FILE=config.yaml opentelemetry-instrumentation program.py
```

Prior art

https://docs.google.com/document/d/10GW_yfi0yHozZlpUdi3t7Vg3XLftlD1brnznzPwWK-Is