+++ title = “Plugin protocol” keywords = [“grafana”, “plugins”, “backend”, “plugin”, “backend-plugins”, “documentation”] +++

Plugin protocol

There’s a physical wire protocol that Grafana server uses to communicate with backend plugins. This is the contract between Grafana and backend plugins, that must be agreed upon for Grafana and a backend plugin to be able to communicate with each other. The plugin protocol is built on gRPC and is defined in Protocol Buffers (a.k.a., protobuf).

We advise for backend plugins to not be implemented directly against this protocol. Instead, prefer to use the [Grafana Plugin SDK for Go]({{< relref “grafana-plugin-sdk-for-go.md” >}}) that implements this protocol and provides higher level APIs.

The plugin protocol is available in the GitHub repository. The plugin protocol lives in the [Grafana Plugin SDK for Go]({{< relref “grafana-plugin-sdk-for-go.md” >}}) since Grafana itself uses parts of the SDK as a dependency.

Versioning

Additions of services, messages and fields in the latest version of the plugin protocol are expected to happen, but should not introduce any breaking changes. If breaking changes to the plugin protocol is needed, a new major version of the plugin protocol will be created and released together with a new major Grafana release. Grafana will then support both the old and the new plugin protocol for some time to make sure existing backend plugins continue to work.

Because Grafana maintains the plugin protocol, the plugin protocol attempts to follow Grafana’s versioning, However, that doesn’t automatically mean that a new major version of the plugin protocol is created when a new major release of Grafana is released.

Writing plugins without Go

If you want to write a backend plugin in another language than Go, then it’s possible as long as the language supports gRPC. However, writing a plugin in Go is recommended and has several advantages that should be carefully taken into account before proceeding:

- There’s an official [SDK]({{< relref “grafana-plugin-sdk-for-go.md” >}}) available.
- Single binary as the compiled output.
- Building and compiling for multiple platforms is easy.
- A statically compiled binary (in most cases) doesn’t require any additional dependencies installed on the target platform enabling it to run

"everywhere".
- Small footprint in regards to binary size and resource usage.