gRPC-Go

The Go implementation of gRPC: A high performance, open source, general RPC framework that puts mobile and HTTP/2 first. For more information see the Go gRPC docs, or jump directly into the quick start.

Prerequisites

- **Go**: any one of the **three latest major** releases.

Installation

With Go module support (Go 1.11+), simply add the following import

```
import "google.golang.org/grpc"
```

to your code, and then `go [build|run|test]` will automatically fetch the necessary dependencies.

Otherwise, to install the `grpc-go` package, run the following command:

```
$ go get -u google.golang.org/grpc
```

> ***Note:*** *If you are trying to access* `grpc-go` *from* ***China****, see the FAQ below.*

Learn more

- Go gRPC docs, which include a quick start and API reference among other resources
- Low-level technical docs from this repository
- Performance benchmark
- Examples

FAQ

I/O Timeout Errors

The `golang.org` domain may be blocked from some countries. `go get` usually produces an error like the following when this happens:

```
$ go get -u google.golang.org/grpc
package google.golang.org/grpc: unrecognized import path "google.golang.org/grpc"
(https fetch: Get https://google.golang.org/grpc?go-get=1: dial tcp
216.239.37.1:443: i/o timeout)
```

To build Go code, there are several options:

- Set up a VPN and access google.golang.org through that.
- Without Go module support: `git clone` the repo manually:

```
git clone https://github.com/grpc/grpc-go.git
$GOPATH/src/google.golang.org/grpc
```

You will need to do the same for all of grpc's dependencies in `golang.org`, e.g. `golang.org/x/net`.

- With Go module support: it is possible to use the `replace` feature of `go mod` to create aliases for golang.org packages. In your project's directory:

  ```
  go mod edit -replace=google.golang.org/grpc=github.com/grpc/grpc-go@latest
  go mod tidy
  go mod vendor
  go build -mod=vendor
  ```

  Again, this will need to be done for all transitive dependencies hosted on golang.org as well. For details, refer to golang/go issue #28652.

Compiling error, undefined: grpc.SupportPackageIsVersion

If you are using Go modules:

Ensure your gRPC-Go version is `require`d at the appropriate version in the same module containing the generated `.pb.go` files. For example, `SupportPackageIsVersion6` needs `v1.27.0`, so in your `go.mod` file:

```
module <your module name>

require (
    google.golang.org/grpc v1.27.0
)
```

If you are *not* using Go modules:

Update the `proto` package, gRPC package, and rebuild the `.proto` files:

```
go get -u github.com/golang/protobuf/{proto,protoc-gen-go}
go get -u google.golang.org/grpc
protoc --go_out=plugins=grpc:. *.proto
```

How to turn on logging

The default logger is controlled by environment variables. Turn everything on like this:

```
$ export GRPC_GO_LOG_VERBOSITY_LEVEL=99
$ export GRPC_GO_LOG_SEVERITY_LEVEL=info
```

The RPC failed with error `"code = Unavailable desc = transport is closing"`

This error means the connection the RPC is using was closed, and there are many possible reasons, including:

1. mis-configured transport credentials, connection failed on handshaking

2. bytes disrupted, possibly by a proxy in between
3. server shutdown
4. Keepalive parameters caused connection shutdown, for example if you have configured your server to terminate connections regularly to trigger DNS lookups. If this is the case, you may want to increase your MaxConnectionAgeGrace, to allow longer RPC calls to finish.

It can be tricky to debug this because the error happens on the client side but the root cause of the connection being closed is on the server side. Turn on logging on **both client and server**, and see if there are any transport errors.