Coding guidelines

Coding Golang

Coding in Go

We code in Go™ and try to follow the best practices and style outlined in Effective Go and the supplemental rules from the Go Code Review Comments wiki.

We also recommend new contributors review the following before submitting pull requests:

- Practical Go
- Go Proverbs

The following tools are executed against all pull requests. Any errors flagged by these tools must be addressed before the code will be merged:

- gofmt -s
- goimports
- go vet

Testing

Unit tests are expected to accompany all production code changes. These tests should be fast, provide very good coverage for new and modified code, and support parallel execution.

Two matching libraries are commonly used in our tests. When modifying code, please use the matching library that has already been chosen for the package.

- gomega
- testify/assert

Any fixtures or data required by tests should generated or placed under version control. When fixtures are generated, they must be placed in a temporary directory created by `ioutil.TempDir` and cleaned up when the test terminates. When fixtures are placed under version control, they should be created inside a `testdata` folder; documentation that describes how to regenerate the fixtures should be provided in the tests or a `README.txt`. Sharing fixtures across packages is strongly discouraged.

When fakes or mocks are needed, they must be generated. Bespoke, hand-coded mocks are a maintenance burden and tend to include simulations that inevitably diverge from reality. Within Fabric, we use `go generate` directives to manage the generation with the following tools:

- counterfeiter

- mockery

API Documentation

The API documentation for Hyperledger Fabric's Golang APIs is available in GoDoc.

The API documentation for Hyperledger Fabric's Go APIs is available in GoDoc.

Adding or updating Go packages

Generating gRPC code

Hyperledger Fabric uses go modules to manage and vendor its dependencies. This means that all of the external packages required to build our binaries reside in the `vendor` folder at the top of the repository. Go uses the packages in this folder instead of the module cache when `go` commands are executed.

If you modify any `.proto` files, run the following command to generate/update the respective `.pb.go` files.

If a code change results in a new or updated dependency, please be sure to run `go mod tidy` and `go mod vendor` to keep the `vendor` folder and dependency metadata up to date.

See the Go Modules Wiki for additional information.

```
cd $GOPATH/src/github.com/hyperledger/fabric make protos
```

Adding or updating Go packages

Hyperledger Fabric vendors dependencies. This means that all required packages reside in the `$GOPATH/src/github.com/hyperledger/fabric/vendor` folder. Go will use packages in this folder instead of the GOPATH when the `go install` or `go build` commands are executed. To manage the packages in the `vendor` folder, we use dep.