Versioning

This document describes the versioning policy for this repository. This policy is designed so the following goals can be achieved.

Users are provided a codebase of value that is stable and secure.

Policy

- Versioning of this project will be idiomatic of a Go project using Go modules.
  - Semantic import versioning will be used.
    - Versions will comply with semver 2.0.
    - If a module is version `v2` or higher, the major version of the module must be included as a `/vN` at the end of the module paths used in `go.mod` files (e.g., `module go.opentelemetry.io/otel/v2`, `require go.opentelemetry.io/otel/v2 v2.0.1`) and in the package import path (e.g., `import "go.opentelemetry.io/otel/v2/trace"`). This includes the paths used in `go get` commands (e.g., `go get go.opentelemetry.io/otel/v2@v2.0.1`. Note there is both a `/v2` and a `@v2.0.1` in that example. One way to think about it is that the module name now includes the `/v2`, so include `/v2` whenever you are using the module name).
    - If a module is version `v0` or `v1`, do not include the major version in either the module path or the import path.
  - Modules will be used to encapsulate signals and components.
    - Experimental modules still under active development will be versioned at `v0` to imply the stability guarantee defined by semver.

      > *Major version zero (0.y.z) is for initial development. Anything MAY change at any time. The public API SHOULD NOT be considered stable.*

    - Mature modules for which we guarantee a stable public API will be versioned with a major version greater than `v0`.
      - The decision to make a module stable will be made on a case-by-case basis by the maintainers of this project.
    - Experimental modules will start their versioning at `v0.0.0` and will increment their minor version when backwards incompatible changes are released and increment their patch version when backwards compatible changes are released.
    - All stable modules that use the same major version number will use the same entire version number.
      - Stable modules may be released with an incremented minor or patch version even though that module has not been changed, but rather so that it will remain at the same version as other stable modules that did undergo change.
      - When an experimental module becomes stable a new stable module version will be released and will include this now stable module. The new stable module version will be an increment of the minor version number and will be applied to all existing stable modules as well as the newly stable module being released.

- Versioning of the associated contrib repository of this project will be idiomatic of a Go project using Go modules.
  - Semantic import versioning will be used.
    - Versions will comply with semver 2.0.
    - If a module is version `v2` or higher, the major version of the module must be included as a `/vN` at the end of the module paths used in `go.mod` files (e.g., `module go.opentelemetry.io/contrib/instrumentation/host/v2`, `require go.opentelemetry.io/contrib/instrumentation/host/v2 v2.0.1`) and in the package import path (e.g., `import "go.opentelemetry.io/contrib/instrumentation/host/v2"`). This includes the paths used in `go get` commands (e.g., `go get go.opentelemetry.io/contrib/instrumentation/host/v2@v2.0.1`. Note there is both a `/v2` and a `@v2.0.1` in that example. One way to think about it is that the module name now includes the `/v2`, so include `/v2` whenever you are using the module name).
    - If a module is version `v0` or `v1`, do not include the major version in either the module path or the import path.
  - In addition to public APIs, telemetry produced by stable instrumentation will remain stable and backwards compatible. This is to avoid breaking alerts and dashboard.
  - Modules will be used to encapsulate instrumentation, detectors, exporters, propagators, and any other independent sets of related components.
    - Experimental modules still under active development will be versioned at `v0` to imply the stability guarantee defined by semver.

      > *Major version zero (0.y.z) is for initial development. Anything MAY change at any time. The public API SHOULD NOT be considered stable.*

    - Mature modules for which we guarantee a stable public API and telemetry will be versioned with a major version greater than `v0`.

    - Experimental modules will start their versioning at `v0.0.0` and will increment their minor version when backwards incompatible changes are released and increment their patch version when backwards compatible changes are released.

    - Stable contrib modules cannot depend on experimental modules from this project.

    - All stable contrib modules of the same major version with this project will use the same entire version as this project.

      - Stable modules may be released with an incremented minor or patch version even though that module's code has not been changed. Instead the only change that will have been included is to have updated that modules dependency on this project's stable APIs.
      - When an experimental module in contrib becomes stable a new stable module version will be released and will include this now stable module. The new stable module version will be an increment of the minor version number and will be applied to all existing stable contrib modules, this project's modules, and the newly stable module being released.

  - Contrib modules will be kept up to date with this project's releases.

        - Due to the dependency contrib modules will implicitly have on this project's modules the release of stable contrib modules to match the released version number will be staggered after this project's release. There is no explicit time guarantee for how long after this projects release the contrib release will be. Effort should be made to keep them as close in time as possible.
        - No additional stable release in this project can be made until the contrib repository has a matching stable release.
        - No release can be made in the contrib repository after this project's stable release except for a stable release of the contrib repository.
- GitHub releases will be made for all releases.
- Go modules will be made available at Go package mirrors.

Example Versioning Lifecycle

To better understand the implementation of the above policy the following example is provided. This project is simplified to include only the following modules and their versions:

- `otel`: `v0.14.0`
- `otel/trace`: `v0.14.0`
- `otel/metric`: `v0.14.0`
- `otel/baggage`: `v0.14.0`
- `otel/sdk/trace`: `v0.14.0`
- `otel/sdk/metric`: `v0.14.0`

These modules have been developed to a point where the `otel/trace`, `otel/baggage`, and `otel/sdk/trace` modules have reached a point that they should be considered for a stable release. The `otel/metric` and `otel/sdk/metric` are still under active development and the `otel` module depends on both `otel/trace` and `otel/metric`.

The `otel` package is refactored to remove its dependencies on `otel/metric` so it can be released as stable as well. With that done the following release candidates are made:

- `otel`: `v1.0.0-rc.1`
- `otel/trace`: `v1.0.0-rc.1`
- `otel/baggage`: `v1.0.0-rc.1`
- `otel/sdk/trace`: `v1.0.0-rc.1`

The `otel/metric` and `otel/sdk/metric` modules remain at `v0.14.0`.

A few minor issues are discovered in the `otel/trace` package. These issues are resolved with some minor, but backwards incompatible, changes and are released as a second release candidate:

- `otel`: `v1.0.0-rc.2`
- `otel/trace`: `v1.0.0-rc.2`
- `otel/baggage`: `v1.0.0-rc.2`
- `otel/sdk/trace`: `v1.0.0-rc.2`

Notice that all module version numbers are incremented to adhere to our versioning policy.

After these release candidates have been evaluated to satisfaction, they are released as version `v1.0.0`.

- `otel`: `v1.0.0`

- `otel/trace` : `v1.0.0`
- `otel/baggage` : `v1.0.0`
- `otel/sdk/trace` : `v1.0.0`

Since both the `go` utility and the Go module system support [the semantic versioning definition of precedence](), this release will correctly be interpreted as the successor to the previous release candidates.

Active development of this project continues. The `otel/metric` module now has backwards incompatible changes to its API that need to be released and the `otel/baggage` module has a minor bug fix that needs to be released. The following release is made:

- `otel` : `v1.0.1`
- `otel/trace` : `v1.0.1`
- `otel/metric` : `v0.15.0`
- `otel/baggage` : `v1.0.1`
- `otel/sdk/trace` : `v1.0.1`
- `otel/sdk/metric` : `v0.15.0`

Notice that, again, all stable module versions are incremented in unison and the `otel/sdk/metric` package, which depends on the `otel/metric` package, also bumped its version. This bump of the `otel/sdk/metric` package makes sense given their coupling, though it is not explicitly required by our versioning policy.

As we progress, the `otel/metric` and `otel/sdk/metric` packages have reached a point where they should be evaluated for stability. The `otel` module is reintegrated with the `otel/metric` package and the following release is made:

- `otel` : `v1.1.0-rc.1`
- `otel/trace` : `v1.1.0-rc.1`
- `otel/metric` : `v1.1.0-rc.1`
- `otel/baggage` : `v1.1.0-rc.1`
- `otel/sdk/trace` : `v1.1.0-rc.1`
- `otel/sdk/metric` : `v1.1.0-rc.1`

All the modules are evaluated and determined to a viable stable release. They are then released as version `v1.1.0` (the minor version is incremented to indicate the addition of new signal).

- `otel` : `v1.1.0`
- `otel/trace` : `v1.1.0`
- `otel/metric` : `v1.1.0`
- `otel/baggage` : `v1.1.0`
- `otel/sdk/trace` : `v1.1.0`
- `otel/sdk/metric` : `v1.1.0`