Go Module strategy with regard to releases

In the new Go world, all Go repositories contain Go modules.

The go-mod-contracts, go-mod-registry, go-mod-messaging, device-sdk-go, and app-functions-sdk are all considered "library modules". The library modules can be versioned and tagged independently of deployable EdgeX artifacts. A tag applied to a library module indicates its version under the requirements of "semantic import versioning". For example, if go-mod-registry is at version 1.1.0, then the tag would be "v1.1.0". Authority to version and tag a library module are at the discretion of the applicable working group chairpersons with release manager coordination.

Deployable artifacts – such as the EdgeX core services and the device services like device-mqtt-go – are versioned together in accordance with the EdgeX release strategy. For example, when the services in the edgex-go repo are deployed as part of a named release, the version assigned to that release (v1.1.0) is assigned to the edgex-go repo via a Git tag. It may also be desirable to apply additional tags, such as "Edinburgh" or "LTS". But to satisfy dependency management via Go modules, a version tag (v1.1.0) must exist.

Freeze

At the point of a freeze date, a new release branch is created from the master branch. All work on the release branch of both library modules and project modules will stop, except for bug fixes or unless authorized by the TSC.

Release

At the point of release

- the application/sdk repositories will be branched. The repository branch will be named for the release (example Edinburgh, Delhi, etc.). The master branch will always contain the latest code.
- the application/sdk repositories will be tagged with the "v" + version number.
- the VERSION files within the application/sdk repositories on the master will be updated to the expected version of the next release (this may be major or minor – see here).
- application repositories that reference / depend on SDKs will be updated on the master branch to reference the new versions

Library Module Release

As the library modules are developed, maintained and released separately from the applications/sdks, and are not deployable units, they do not need to follow a versioning cadence that aligns to a release. The current proposal is that they rarely branch and tags proceed in sequence on master for the most part.

It is possible that a given version of a library module is included in an LTS release. Here are two successive scenarios for how I propose we would manage the versioning / branching in this case.

Scenario 1

Go-mod-core-contracts v1.5.0 is part of the Edinburgh release. We find a bug in it and fix it on master. Since no minor revision increment has been tagged, we can simply tag this commit with "v1.5.1" and consume it in the Edinburgh dot release.

Scenario 2

Following from the above go-mod-core-contracts v1.5.1 is part of the Edinburgh dot release. We find a bug in it and fix it on master. At the point where the bug was fixed, we've just deployed the Hanoi release and our go-mod-core-contracts library is now at v1.7.x on master. In order to deploy the fix to the 1.5.x stream, we do the following:

- Create a "backport/1.5" branch at the point of latest 1.5.x tag (in this case v1.5.1)
- Add a commit containing the fix to this branch tagged as v.1.5.2
- Edinburgh dot2 release now consumes this version.

The above is based on how Consul manages their versioning

https://github.com/hashicorp/consul/releases/tag/v1.2.0 (1.2.0 version on master 25-Jun-2018)

https://github.com/hashicorp/consul/releases/tag/v1.2.3 (1.2.3 version still on master 13-Sep-2018)

https://github.com/hashicorp/consul/releases/tag/v1.4.0 (1.4.0 version on master 14-Nov-2018)

https://github.com/hashicorp/consul/releases/tag/v1.2.4 (1.2.4 version on backport/1.2.x branch 20-Nov-2018)