• Go 1.13
- Per documented release goals for Geneva
- Update your local environments if you haven’t already

The rest of our topics today are related to the API V2 Skeleton PR. For reference: [https://github.com/edgexfoundry/edgex-go/pull/2285](https://github.com/edgexfoundry/edgex-go/pull/2285)

1.) Go-Kit
   a. Some recent questions around how this framework overlaps in intent with what we’re doing in API V2. Are we re-inventing what already exists?
   b. Brief overview of our project’s investigation into Go-Kit
   c. Why didn’t we pick it?
      i. Rewrite wasn’t feasible for V1 API workstream
      ii. Dependencies
         1. [https://github.com/go-kit/kit/blob/master/go.sum](https://github.com/go-kit/kit/blob/master/go.sum) (488 lines)
      iii. No serious consideration at the time for alt. transports or HA
   d. Useful links
      i. Non-goals of Go-Kit
         1. [https://github.com/go-kit/kit#non-goals](https://github.com/go-kit/kit#non-goals)
      ii. Layering
         1. [https://gokit.io/faq/#services-mdash-what-is-a-go-kit-service](https://gokit.io/faq/#services-mdash-what-is-a-go-kit-service)

2.) Open question on /version response from PR 2285
   a. [https://github.com/edgexfoundry/edgex-go/pull/2285#discussion_r373252402](https://github.com/edgexfoundry/edgex-go/pull/2285#discussion_r373252402)
   b. For Geneva, remove V2/version
      i. Just use existing /version endpoint which will return the service version (e.g. v1.2.0)
      ii. For formal V2 release re-implement /version according to V2 patterns

3.) Pending review comments from Canonical (Tony / Ian J)
   a. Still pending

4.) Recap of action item from TSC meeting yesterday
   a. General agreement that it would be valuable to have a diagram, something visual, that shows the decoupled elements required in order to implement a use case (AddDevice, AddEvent, etc)
      i. Lexicon for common terminology
         1. [https://github.com/michaelestrin/edgex-go/tree/issue-2277/internal/pkg/v2](https://github.com/michaelestrin/edgex-go/tree/issue-2277/internal/pkg/v2)
      ii. Layering diagram w/package references
         1. Refer to ReadMe
      iii. Decoupled elements (w/relevant interfaces?) that should be implemented for assembly into a feature.
         1. Visual diagram + 1
   b. Example implementation endpoints that exist
      i. Metrics, test, version, ping
   c. Missing elements for docs / comms
i. Service-specific use-case implementation (AddAddressable, etc)

ii. Will this help or hinder the review process if there’s a baseline eval of complexity

d. Pending action items
   i. Implement use-case for metadata/Addressable
   ii. ADR
   iii. Docs for new dev onboarding

e. Alignment of 3 EdgeX contexts w/V2 impl patterns (DS, Core, AppFunc)
   i. Architecturally aren’t the 3 contexts already dis-similar?
   ii. Jim W. expressed desire to align the three architecturally
   iii. DS, Core and AppFunc all targeting go-mod-bootstrap integration for Geneva which moves us in an aligned direction
   iv. Should alignment be on the V2 spec or the underlying implementation in addition to the spec?
      1. Agreement on target to implement V2 integration by Hanoi (DS, Core, AppFunc)
      2. Still question about underlying implementation approach

v. Clarify “adoption of V2 spec”
   1. Usage of Core/V2 by DS and AppFunc?
   2. DS and AppFunc providing their own V2 endpoints?
   3. DS and AppFunc using the same implementation for V2?

vi. Timeline
   1. Geneva – Impl V2 in Core
   2. Hanoi – Usage of V2/Core by DS, AppFunc
      a. AppFunc / DS impl (define?) their own V2 spec
      b. Ireland – Impl to follow above definition?
         i. Lenny says lightweight impl of V2 AppFunc in Hanoi

vii. Follow-up meeting
   1. Define “what’s to come out of the V2 API”
   2. Do we have common goals?
   3. What are we trying to target?
   4. Should the implementation / architecture be the same across all contexts?
   5. How does this benefit certification?