Core Working Group Agenda (14-June-2018)

Attendees:

- Janko asked that we revisit “glide” replacement w/"dep"
  - History on this – PR #256 ([https://github.com/edgexfoundry/edgex-go/pull/256](https://github.com/edgexfoundry/edgex-go/pull/256))
  - Vgo to be official in 1.12
    - [https://research.swtch.com/vgo-accepted](https://research.swtch.com/vgo-accepted)
  - If schedule holds, will be Feb 2019 ([https://golang.org/doc/devel/release.html](https://golang.org/doc/devel/release.html))
  - Discussion centered on what particular issue dep would solve as opposed to glide. Timing of switch, possible to consider move to dep if vgo gets moved out.
    - Or possibly evaluate other tools
  - For now, leave as is.
- Add glide.lock as part of California branch to ensure versioning and consistent builds
  - Jim, Jeremy and Trevor
New Business

- Readiness of export connectors (Janko / Mainflux)
  - New contributions being reviewed
    - https://github.com/edgexfoundry/edgex-go/pull/297
  - For California? QA?
  - Needs more testing / verification – Andy / Keith?
    - Requesting tools / accounts for public cloud vendors to enable testing
    - Approval has been obtained from the Board.
  - Platforms
    - Google IoT
    - Azure IoT
    - Janko to make full list and submit to Keith

- Code Quality Pipeline
  - For example: https://medium.com/@jgautheron/quality-pipeline-for-go-projects-497e34d6567
  - Suggested tools include
    - SonarQube
    - CodeClimate
    - Coverity (Keith)
  - Minimal desired functionality?
    - Configurable rules, clear reporting, integration with source control and CI/CD.
  - Clarify pricing – run in CI/CD vs per seat. Latter isn’t really doable.
  - Ensure code quality tools run when the PR is created
    - Add as required check before PR can be approved.
  - Concern about devs not being able to run checks locally before submitting PRs
  - Tools should support multi-language
  - Does LF have a recommendation based on other projects?
    - Identify key Go-based projects and do a survey (Brett / Jeremy)

- Structured Log Format
  - EdgeX already imports uber-go/zap
    - https://github.com/uber-go/zap
    - Used by export services
  - Need to decide on format of the message – fields and values
  - Logs to file system by default
    - How do we route to support-logging?
    - Performance impact of same.
  - Can log level be configurable while application is running?
    - https://godoc.org/go.uber.org/zap#AtomicLevel.ServeHTTP
  - Do we need to consider an output formatter?
- JSON
- Delimited String
- Custom
  - Need to flesh out
    - Eric / Trevor – Document need and criteria (possibly targeting distributed environment)
    - Tony – Document need and criteria (targeting smaller, constrained environment)
    - How do we implement a solution that works in both of these domains?
    - Rodney to follow up with more info based on known solutions

Future items of discussion include
  - Tracing capabilities
  - Alternates to JSON serialization
    - (Jim) Move up in priority due to Hitachi deadline
    - (Tony) Probably need an event schema change to support byte array
      (Reading specifically)
  - Replace RAML with Swagger
    - Ask Hitachi for assistance with conversion
    - No decision yet on moving to Swagger, hoping for the above.
  - (Keith) Restart discussion on UUIDs
    - Perhaps dependent on Mongo replacement