Geneva Planning Items

This is a high-level list of items on the table for the Geneva road map. They are not listed in order of priority. The actual list of deliverables to be included with Geneva will be determined at the upcoming face-to-face meetings in Phoenix, AZ.

- Dynamic Device Provisioning
  - Support the dynamic onboarding of new devices which conform to an existing device profile
  - This eliminates the need to itemize devices in the device service configuration.toml
  - Also could include the dynamic re-assignment of a device from one device service to another as in the case of a moving (wearable) device
- Error Handling Refactoring
  - https://github.com/edgexfoundry/edgex-go/issues/1734
  - Anthony Bonafide presented a possible approach in Core WG (22-Aug-2019)
- Explicit Request and Response types for API endpoints
  - These would be defined in go-mod-core-contracts
  - The current models used for request/response as found in go-mod-core-contracts was extracted from edgex-go so that other applications would not have to import the entirety of edgex-go in order to be interoperable.
  - There is unnecessary complexity involved when creating requests (see devices and device services) because of requirements applied to the model's integrity but not necessarily the request
  - All requests and responses for the Core/Supporting service APIs should have explicit request and response models.
    - Example: AddDeviceRequest, AddDeviceResponse, UpdateDeviceProfileRequest, UpdateDeviceProfileResponse
- If Geneva becomes V2, how to support V1?
  - Proposal – Rather than keeping the /v1 code in the various repos that rev to /v2, we should branch the most recent v1.x release into an LTS branch.
    - Subsequent supporting changes would be made there.
    - This LTS branch will never have any /v2 code.
  - The above removes all /v1 code from the /v2 codebase. Thus an application interacting with the /v2 codebase could not expect to call a /v1 endpoint and have it work.
- Elimination of Value Descriptors
  - The current ask is to include the typing metadata on the event readings themselves.
- Querystring support in the Event payload
  - The use case here is that certain device services require querystring parameters in order to obtain readings from one or more registers within a single device (such as Intel's RFID case)
  - Inclusion of the querystring params was proposed as a way to record what the source register was when the event was created
    - Just tracking it back to the device may not be granular enough
- Configuration.toml structural changes as warranted
  - Specifics will be added per discussion
  - Example – Align security service configuration structure to standardize with Core Services
- Core Services Monolithic deployment
  - During the planning of Fuji there was discussion related to the possibility of deploying Core/Supporting services as a monolith. Some prototyping was done. Do we want to pick this up?
- Messaging directly between Southside / Northside
  - Bypass core-data, etc
- Support some other messaging bus besides ZeroMQ
- Environment variable overrides in order to remove the docker configuration.toml
  - Use base configuration.toml, override docker hosts and other relevant settings via environment vars in docker-compose.
  - Applicable to docker deployments only
- Configuration consistency w/r/t cmd line, env variable override and file persistence
  - Order of precedence
  - Naming
  - Across all tiers in the platform (Device Services is outlier as of Edinburgh)