Architecture Issues and Decisions

- Open Discussion Items
  - General
  - Domain Identifiers
- Closed Discussion Items
  - General
  - Messaging technology
  - Go Lang Discussion Items
  - Configuration files
  - Service Identifiers
  - Device Services SDK Requirements
  - Edinburgh Release -- Configuration Registry Structure

Information here will be moved to the Design Decisions Project Board once approved by the TSC

The EdgeX community has and will continue to explore issues of design and architecture of EdgeX Foundry, its many micro services, deployment mechanisms, testing apparatus, build process and more.

This page contains the list of design/architecture items currently under review by the various working groups and committees. It also contains a list of the issues reviewed and now considered closed pending any new evidence or material in order to reopen the issue.

Readers are encouraged to review the EdgeX mailing list (in particular the edgex-devel@lists.edgexfoundry.org list) and working group meeting minutes (https://wiki.edgexfoundry.org/display/FA/Working+Groups) for more information and background on the issues.

Open Discussion Items

General

- Our RAML documents reference draft-03 of json-schema.org. Some questions:
  - draft-06 is the latest, should we move?
  - Do the current Java microservices use the specified schema for validating JSON?
  - Should we be using the schema for automated testing validation?
  - Should be be using the "enum" construct (supported by draft-03)?

Domain Identifiers

- How should concepts like Device or Addressable be uniquely identified? Currently using unique name strings
- How should persistent store identifiers be related/not related to these identifiers.

Closed Discussion Items

General

- Copyright header – stays the same (see contributor page) until LF has an alternative header and tool to provide
- Enums & JSON – use strings in place of any language specific enums or Iota in JSON messages and serialize/deserialize as necessary in your language of choice.

Messaging technology

- Immediate needs between Core and Export services: what are the appropriate options to avoid Java/Go issuses
- Longer term, what should be considered the leading service to service communications protocols (outside of REST) and why?

Consensus agreement that for now, we stick with ZeroMQ through California release. Dell team will look at interfacing the message elements such that 0MQ could more easily be replaced by a user.
Longer term, we do need to address the needs of the Windows developer and provide more ease of use for ARM. We'll relook at these and potentially offer alternatives post California release.
The community is encouraged to do some POC work with alternative technologies like those listed above to be able to provide more information for the relook after California.

Go Lang Discussion Items

- Minimum Go Lang version required is 1.9 (1.10 not used by Alpine yet and won't allow compile in Docker image, but performance on Arm with 1.10 is desired and should be moved to as soon as possible).
- Repos names will be in the form of <service>-<language>. For example core-domain-go
- Work group leads can make the determination about repositories for their services. The applications work group, for example, has decided that all Go export code/micro services will go in one repository (export-go). The decisions are passed to work group leads (working with their development leads) and on a language by language basis that best suits the needs of development and community.
- Glide will be our vendoring tool of choice (may want to relook with GoDep now released and in production - as of 3/21 Go Project Meeting)
- Line endings in code files are different for *nix versus Windows. In *nix, lines end with “\n”. In Windows, lines end with “\r\n” (carriage return, line feed). This creates issues when pulling/working with code created in different environments. The *nix line feed is the desired line ending for EdgeX Foundry source code. Windows developers need to configure tools to use and apply the appropriate line endings.

Configuration files

- Agreed to use TOML for specifying configuration information (for local config as well as to config-seed) going forward. Java code will not be altered at this time.
- Reasons for TOML (over JSON or YAML) is that white spacing is ignored and it may be validated in the future.
- In the future, EdgeX may support multiple types of configuration - allowing the user to pick the file format.

Service Identifiers

- How should micro services be identified uniquely and how should that relate to the address or host of the address?
- Proposed design doc on service naming, availability, configuration, etc.:
    - ServiceNameDesign.odt
    - ServiceNameDesign-v2.odt (inclusive of typos and clean ups and Trevor Conn comments)
    - ServiceNameDesign-v3.odt (convergence of Tony/Trevor comments - keeping comments in areas still in need of cleanup/resolution)
    - ServiceNameDesign-v4.odt
    - ServiceNameDesign-v5.odt
    - ServiceNameDesign-v6.pdf

Device Services SDK Requirements

- edgex-cali-delhi-device-requirements-v8.pdf

Edinburgh Release -- Configuration Registry Structure

- EdgeX-Config-Edinburgh-v3.pdf