Core Working Group Agenda 4/12/2018 meeting

Attended by: There may have been in attendance later in the meeting after the list of attendees were assembled.

Discussion and action items as a result of meeting in RED.

Old Business

DevOps issues/updates

- Automation of black box tests is priority 1
- Go & Arm CI (64/32 bit)
- Snap CI
- Build 0.6.0 of EdgeX: suggest April 20th for HM event

Documentation

- Now available for review. See PR #127. Soon to be merged.
- Feedback welcome (Andy – how do you want it?)
- Wiki page updates are trackable. Please be advised that changes you make in Wiki are not reflected in new docs yet, but will be.
- "Pretty" version available on IoTech Web site (when?? – Andy)
- Target early May cut over from Wiki to new documentation.

DS Requirements Discussion

- Open issues in DS requirements
    - Query All command results
    - logging/scheduling - *inprogress*
    - actuation commands RAML accuracy - *tbd* may be resolved
    - metadata updates (/callback) RAML incomplete - *tbd*
    - finalize appendix A on settings – does current settings/config work take care of this?
- New revision available??

License file issue

- Jeremy to implement for Docker files
- Issue submitted against export services for attribution file need (#120)
- Can we close this discussion?

Core Services Refactor

- Trevor to present what he's done for core data
- Planning to submit to edgexfoundry-holding for review and further comment

Service Naming, Availability and Configuration Bootstrapping

- Requesting approval to implement per https://wiki.edgexfoundry.org/download/attachments/7602423/ServiceNameDesign-v6.pdf?version=1&modificationDate=1523468903687&api=v2
- Major tenets of this document:
    - Config rules

- Bootstrap - 3 properties
  - -registry – use the registry set to either true or false (default false)
  - –confdir – path to local configuration file (default ./res)
  - –profile – configuration profile to use when loading the configuration (defaults to the default profile – which is the configuration file with no addition profile information attached – see below). This option would be used for microservices only when loading local configuration.
- Microservice Naming
  - The “name” of the service is not its unique identity but rather an indication of the API set it satisfies. The service name will identify the type or nature of the service but is not guaranteed to uniquely identify the actual instance.
  - A service will not change its name. The service name is really an indication of the API set it offers.
  - The service name for each service type is known to all of EdgeX. Therefore, the service name will no longer be provided by configuration (local or by the registry).
- Service to Service comms
  - When the registry is enabled then Service A should request the address of Service B from the registry by name (allows for load balancing or other redirection in the future)
  - When the registry is not used (as in development), then Service A will use the address of Service B provided by the local configuration settings for Service B
- Availability
  - When an EdgeX microservice starts, it should not be considered “available” until the following ordered events occur:
    - Service configuration is loaded (locally or from the registry as described above)
    - Required external services are initialized and/or verified to be available (that which makes the service truly “available” and ready to take on requests from the other services)
    - Start responding affirmatively (pong is the response today) to ping API requests. It should return an appropriate HTTP error response when responding negatively.
    - If the registry is enabled, register itself

- Implementation by Dell in core & support, Tony to add to make reference in SDK requirements and put in SDK, Janko to add to work tasks for export services

Contribution Guidance Top 3

1. Commit subject line messages should explain the problem that the commit is solving in 50 characters or less. “Fixing bug” or “adding comments” doesn’t explain the problem the commit is solving!.
2. Provide a message body to the commit to provide more detail explanation. When you do, keep lines in the body to 72 characters or less.

   You can add a commit subject line and multiple message bodies like this:

git commit –s –m “This is the subject; keep to 50 char” –m “This is the first line of the message body"$'\n'"This is the second line of the message body."$'\n'"Keep all message lines to 72 char.”

3. Commit your changes in logical chunks. Make it easier for the community to review, comment on, and accept your contribution. For example, create a separate commit for a big fix and an enhancement.
4. Locally merge (or rebase) the upstream development branch into your topic branch. Do not merge master into your local PR branch (this is the lazy approach to dealing with committed code locally in a branch, and then realizeing that your PR isn’t merge-able because HEAD has changed). See https://git-scm.com/book/en/v2/Git-Branching-Rebasing if you need more help.

- Tony was going to provide rewrite of #4
- With that, seek approval and post to applicable pages

New Business

ReadMe Docs Generation

- Jeremy/Andy discussion

Technical Debt and other issues we want to address at June F2F

- Next security features
- Drasko/Rodney issue from security WG (paragraph needed)
- First steps to distributed-EdgeX (services running on different host machines and impact to security reverse proxy, command calling, etc.)
- Load balancing – multiple instances of services
- ????