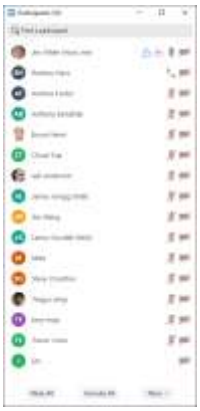
Agenda - Hanoi Pre-wire

March 31-April 1



Logistics (and intros if needed): meeting today 7am – 10am PDT; tomorrow 12pm-3pm PDT

Goals today:

1) Perhaps, establish some new tenets that help guide Hanoi and future release work
2) From all the possible features, tasks, tech debt or work we can do for Hanoi, what are possibly in scope for Hanoi (what is possible in and what is definitely not in scoping exercise)
    a. For that work that is possibly in, do we need some preliminary research/prototyping, etc. done before Hanoi planning conference to make a decision? Are there competing ideas that need to be compared/contrasted and presented by teams?
    b. Of all the possible work, are there items on our long list that should be removed – forever? Can we use this exercise to cleanup our backlog of things no longer to be considered?

Some immediate issues to resolve for Geneva

- Race condition testing. Tony has discovered that when running with race detector flag on, all sorts of stuff is coming out.
    - Is this a concern?
        - Not sure that it is for sure issues but valid warnings
        - Some DS timestamp issues (2281 bug) that may be related to race condition and suggests we have many such issues. Could this be handled by configuration?
    - What are others seeing?
        - Whenever it has been turned on it has given CI/CD to problems; container doesn't handle it well
    - What actions do we need to take prior to Geneva release?
        - Internal to a service – race conditions
        - Service interop – race between ordering ops from multiple services (not detected by race flag)
    - Is this something we should be doing regularly through DevOps/Testing?
        - In makefile in edgex-go its on (race detector flag)

            - It will fail unit tests if race flag on and it discovers issues
        - Not being done in SDK makes (and services those SDKs make – DS, App Services)
        - Not being done in all the modules
        - Defer – race detection for all unit testing for Hanoi
            - Some research necessary on why it was crashing in CI/CD
    - Actions
        - Mike & Iain – make files for SDKs and related services
        - Tony to look at other services for races
- Config seed PR – do we need mini-freeze for Lenny's PR?
    - Lenny keeping up with rebasing
    - Some bugs still being worked

Architecture Tenet Discussion

Before we begin our exercise to preliminarily scope Hanoi – or at least scope what is part of the Hanoi planning conversation, there is before us some fundamental questions

1) Where does EdgeX run? Is it a platform that could run in the cloud and in what capacity? Is it just a "thin edge" platform – running on things like a gateway in single instance mode?
2) Do we or will we allow parts of EdgeX need to be able to be distributed across hosts? If so, what parts? How would we (or would we) secure the communications (generally – not details)?
3) Would parts of EdgeX need to run in some form of high availability mode? Abide by 12 factor apps? Would all services need to be HA?
4) What does the architecture of EdgeX look like at some point in the future? What parts are message bus based? Where is there persistence of sensor data at the edge (still with core or support data)?
    a. Yes – potentially for lots of services; both data and control plane
    b. Yes – but start with data plane and see how it looks
    c. Yes – but add service by service
    d. If we do we need to think about generic events
    e. Enable the services with messages on a use-case by use-case circumstance; need to be individually considered.
5) Are there services we want to combine?
    a. Goal: simplicity and performance (inter-process communication reduction);
    b. All the core in one – perhaps core data out since it may be support-core
    c. Is it a configuration/deployment operation vs hard coded single exe

Dell Perspective – more computing resources coming down to the edge; more HA farther out to the edge. Thin edge makes sense at device level. DS need to run as stand alone agents – on constrained device potentially. DS as single instance service that roles up to core (and behind it) in a cluster. More monitoring in the cluster that gives visibility to the platform. Stop thinking about EdgeX as a set of micro services on a Raspberry Pi. Everything but data ingestion needs to be scalable.

Some agent to restart things if it fails

Steve O – failover needs to be throughout the system – to include device service.

Beechwoods – still interested in resource constrained platforms, but not necessarily monolithic EdgeX install on a Pi, but services spread out over collection of platforms. HA features – in favor; but still need to operate on resource constrained platforms. This is more about distribution of services – but not opposed to HA. We can't presume that security has to be in the services (versus around the monolithic platform) since they can be distributed.

Tony – for Hanoi, should focus on core platform and not HA. Focus on DS only in distributed mode.

Is HA a goal for all services (except DS)? - leaning to HA, but need more real customer input and careful not to lean away from very thin environments; on hold for now

Is service distribution a goal for all services? TBD

Distribution for DS?

If we can align on these basic questions, there is a feeling that this should start to guide our opinion on scope for Hanoi and even beyond.

Hanoi planning – per doc at https://wiki.edgexfoundry.org/pages/viewpage.action?pageId=37912800

Put the elements in our list in one of these categories:

- Possibly In
- Probably out of scope for Hanoi
- Get rid of it – never going to happen or OBE

General / Cross Cutting Concern

1.3

New feature

- Go 1.15? But evaluate whether we take advantages of some of the new features; 1.13 wouldn't be supported if 1.15 comes out – Anthony Bonafide – research features we want and that could impact us. (consider 1.14 as well?)
- Message bus between select services (DS – Appl Service)
    - Core – as a optional subscriber on bus to persist (stretch??)
    - Requiring broker or going brokerless or your choice – Lenny to present broad stroke arch at planning meeting
    - Supporting synchronous, asynchronous or both
- Support running a service on a different host than the other services. Specifically making it easier to run a device service on a different host that the rest of Edge. As a stretch to this goal, allow for select service high availability.
    - DS – as stand alone – distributed services (security ?'s, orchestration ?'s) – every one should understand Byron's Option B proposal (Bryon / Tony– to send out slide deck to TSC for review; refining to look at DS specifics)
        - Security relative to DS to core and command to DS – part of Bryon's research on what other impacts

- Improving EdgeX resiliency in face of issues or non-availability of some resources/services/etc. (typically for core and above services and not device services)
- Ensure all micro services follow 12 factor app methodology (see https://12factor.net/)
    - Allow services to be load balanced
    - Allow services to fail over
    - Allow for dynamic workload allocation
    - Allow services to live anywhere and be moved without lots of configuration to be changed in other services
    - Allow services to be distributed across hosts - and across API gateways (requiring service to service communication protection)
- Support truly distributed microservices
    - Allow services to run on multiple host machines
    - Secure distributed EdgeX with reverse proxy
    - Cross EdgeX instance command actuation (ex: device X on EdgeX box A triggers action on device Y on EdgeX box B)
    - Front a collection of duplicate microservices with a load balancer (allow for the microservice copies to scale up or down based on load); allow multiple instances of any microservice (for future load balancing and scaling efforts - today only single instances are allowed)
- Develop a test environment/playground to test high-availability and distribute service functionality.
- Allow for new category of micro services: “Sharing Services” for East/West data exchange with non-EdgeX entities
- Meetup support
- Hackathon kits/support
- LTS on 1.x release

Tech Debt

- V2 API – IOTech readout (TAF testing)
- Deprecate Mongo
    - Jim to get with Andre to discuss Redis security issues
    - Do we mark Mongo as deprecated for Geneva – yes if Redis ready to go
- Removal of Value Descriptors in favor of data typing in the reading
    - Part of V1 v V2 discussion
    - Larger discussion about when V1 is deprecated/removed
    - Devices with embedded device profile and addressable removal from all EdgeX
- Implementation of generic error handling across services
    - How to wrap errors
    - How to sanitize errors
- Develop process for security vetting of 3rd party components – security WG (Tingyu/Diana with WG inputs) provide high level idea of process and what it would incur
    - Review of licensing as part of the considerations
- Licensing file distribution with artifacts
    - If we don’t cover in arch’s meeting

- Restructuring our compose files to take advantage of compose file overrides, which removed the duplication in all our compose files. See https://devilbox.readthedocs.io/en/latest/configuration-files/docker-compose-override-yml.html - Lenny to provide readout/demo

Core / Supporting

New feature

- Allow for device hierarchy in metadata model: a device could be a manager for another device while also collecting data itself. Sending a command to a managing device could mean sending a command to all associated devices.
- Command: in order to protect the device from harmful commands, there should be the possibility to set a Min and Max limit for the value that is accepted on every single command. in fact the command service today is rather a hollow simple proxy, but in the future we very much envisioned adding additional security, caching to avoid having to hit the DS when unnecessary, and even grouping command requests for better resource conservation (especially for devices like BLE that get woken up when you hit them).
    - Command should be a dumb pass through/proxy
    - It should be a device service check – not command service; move feature to DS consideration
- Change core-data to support data
    - Coupled to sending data to app services directly from DS
    - Could involve changing the position of core-support-data in the data plane flow
- Drop logging service
    - Removing remote logging by service
    - Use bootstrap to write to standard out and optionally file
    - Non-backward comp change
    - Mark deprecated in Geneva (and remove from compose) and remove whenever
    - UI concern – jim to check with UI group
- Support for alternate logging formats and/or more structured logging
    - Look at structure (logging type) – Log for Us exploration (https://github.com/sirupsen/logrus)
    - XML, CSV vs JSON
- Support automatic migration of Devices between Device Services
    - Architectural topic to explore (Jim action)
- Core Command support send commands to devices based on "label" (#1884 in edgex-go)
    - https://github.com/edgexfoundry/edgex-go/issues/1884
    - Actuate a whole group of devices

Tech Debt

- Tests should run cleanly when passed the "-race" argument
- core-command coupled to core-metadata database
    - could be covered by combining the two services (backward comp issue)
    - V2 issue
- ~~Restructure the Redis DBClient implementation for Event/Reading~~
    - Refactoring exercise

- https://github.com/edgexfoundry/edgex-go/issues/2166

Application Services / App Functions SDK / Analytics Integration

New feature

- Kuiper – next release
  - Decouple from EdgeX – go-mod-contracts and messaging
  - Walk stage
    - Handles multiple devices and filtering in the rules engine itself (discuss with Kuiper team)
    - Enhance REST to match CLI??
- Deprecate Drools rules engine
  - Drop Kuiper in compose now; and label Drools as deprecated; remove Drools in Hanoi
- ~~Support Cloud Event export~~
  - Done in Geneva
- Support additional northbound endpoints and protocol types. Examples include:
  - Examples we have today: Azure, AWS, IBM Watson, HTTP(S), MQTT(S)
  - Tencent
  - Alibaba
  - IoTivity
  - SAP HANA
  - DDS
  - AMQP
- Support enrichment functions (an EAI concept) in export services (or application services). Allow additional data or information to be added to the flow of sensor data to the northbound side. This might be information about the sensor/device that captured it or information about the commands to actuate back down on a sensor/device.
  - Pretty much covered with custom app service via SDK
    - Provide example of how to do this
  - Security issues
- Define metadata about on the "gateway" or host (identity, location, …)
  - Potentially make that information/metadata extensible
  - How can we then make metadata of gateway available with app service export
  - Goes beyond app service
  - Malini – action item to layout what this feature includes/high level design for planning
- Integrate to edge software/agents (on data plane)
  - AWS Greengrass
  - Microsoft IoT Edge
  - Could be of more value for control plane
- Support additional northbound formats
  - Haystack
  - OPC UA
- App Service Configurable – Allow same function twice
  - Waiting for stronger user requirement
- Fork the pipeline & Pipeline per topic

    - Support multiple topics in SDK - go mod messaging
    - Allow various paths for different sensors for example
    - Can be done with code internal to a function - but a bit kludgy; could be useful and nice
    - Lenny/Mike – to provide read out on design/scope level of priority
- Multiplexor to split out data from device services
    - Take multiple readings and refunnel them back as separate readings in core data
    - As built in transform; not having to do it on your own via custom app service built with SDK
- 0MQ export function

Tech Debt

- Improve binary data support
    - Local edge analytics may be fed binary data and create intelligence from it to allow for actuation at the edge based on the binary data (example: examine an image an detect the presence of a person of interest).
    - Better support of CBOR – detect exception
- Support alternate message formats for service-to-service exchange (Protobuf, XML, etc.)
- Update SDK to use new persistence service

Device Services / Device Service SDKs

New feature

- Protect the device from harmful commands, there should be the possibility to set a Min and Max limit (or other profile checks to protect the device).
    - Preexisting issue for this
- Support Cloud Event import
    - Could this be done with SDK and REST or MQTT DS?
    - Provide example vs actual service
- Data filter design between DS and Core Data
    - Provide a design about how to implement this before implementing.
    - If possible, can the filter functions be shared across App Services and D.S. (w/ App WG)
        - Iain, Tony, Steve & Mike, Lenny to chat before planning meeting about possible alignment, use of app funct pipeline architecture?
- Support additional southside connectors
    - Profinet/Profibus
    - CANBus
    - LORA
    - IoTivity
    - Zigbee
    - ZWave
- Better support mesh network protocols (Zigbee, BACNet, etc.) where devices know and sometimes manage other devices
- Downsampling: It is mentioned that the device service may receive from the device new unattended readings (e.g. in a pub/sub type of scenario). In this case, there should be a setting to specify whether we accept all readings or we decide to downsample because the source is

pumping data too fast. This is actually a very common scenario when you deal with high frequency sensor packages.
    - Just for async readings? Auto events?
    - What are use cases?
    - More on device service to core data basis
    - Does gorilla mux deal with any load exceptions??
- Bound checking
    - Number of operations that can be done
    - Max request size (that lends to DoS, etc.)
    - Could be more globally applied – a REST QoS
    - Each WG should explore implications and design
    - Articulate the problem better; limit scope a bit; design target for Hanoi??
- ~~Cache fairly static information (device profiles, device information, etc.)~~
    - Done
- ~~Allow for easier device removal~~

Tech Debt

- SDK alignment – can / should the DS and Application Functions SDKs be more aligned (design, usage, etc.)?
    - Filtering exploration would be first step
    - We are doing it where it makes sense
    - Take on a case by case identification of an issue

Security

New feature

· Create a hardware secret storage design
o HW secure storage abstraction layer
o How to protect the Vault Master Key
· Sign docker images
    How would we check that those services running are signed? OH would do this in their env.
    We would rely on deployment/orchestration system to do the checking
    Wait for LF Edge to provide guidance and tools for CI/CD pipe to do this

Let's some user feedback and stats on what do most commercial products do in Docker Hub – Malini / Bryon to do some research on what VMWare/Intel do

· ~~Identification management for EdgeX /Enable Vault identify secret engine~~
    ~~Security between services~~
· ~~Enable Vault PKI secrets engine~~

Covered under service to service comms with Bryon readout

· Enable user-specified Kong proxy certificate
    BYO Cert for external only
· Streamline proxy certificate upload flow

· Secret store unsealing daemon
    Part of secret store setup/initialization today
    Potential refactor exercise? Internal to security WG consideration

Tech debt

- Configure TLS encryption for kong+postgres channel (one way TLS)
  - Cert generation for services. Could be handled with the item below
  - Special case of service of service TLS (Postgress vs our service)

- ~~Secure service to service communications~~
  - ~~Cert generation for services, will depend on the decision of architecture~~

Tech Debt

Secure Consul - No communication protection with Consul
- No authentication/authorization when seed into consul

Malini / Tingyu to provide readout on scope, high level how to on this

- ~~Create and use a per service Vault token in the security services~~
  - To be done for Geneva
  - May need some examples in docs

Blackbox tests of APIs through the API gateway
- Design work

Part of Test/QA TAF plan

Do we need "negative testing"

Tingyu – to find out what we have and what we might need

- ~~How to implement HTTPS in EdgeX (that is, how to protect all service endpoints with HTTPS)~~

part of service to service comms discussion

How to implement role-based security across our all EdgeX services.

Container security (running of the reference images via compose file)
- Set no-new-privileges option on containers
- Make docker container images read-only
- Make docker container images run as non-root users

- Eliminate remaining dependency on curl and jq in scripts/security-proxy-setup-checker.sh
  - Need utility that does this in a slightly different way

- EdgeX too tightly coupled with configuration
  - Tied into bootstrapping with Consul

- Secure Kong admin port with TLS
- Enable CORS (for Javascript type request across domains) for API Gateway
  - More feedback necessary?
- ~~Rewrite security-secret-setup to remove legacy cruft~~
  - Refactoring as part of PKI issue
- Pluggable random database password generation

- Add method to retrieve metadata inserted by file-token-provider
  No use right now
- Check the consul/registry for the configurable settings in security-secrets-setup
  Bryon says no gonna happen

ADR – for secrets provider (abstraction for go-mod-secrets) Lenny
Allow us to have secure and non-secure config
Implemented in App Services SDK; use this design in other services (through go-mod-secrets)

New feature

- Add hooks for hardware-assisted protection of Vault Master Key to security-secretstore-setup
- Streamline proxy certificate upload flow
- Create a hardware secret storage design
  - HW secure storage abstraction layer
  - How to protect the Vault Master Key
- Ensuring the services running are those expected and authorized (w/ DevOps assistance)
- Enable Vault PKI secrets engine
- Enable user-specified Kong proxy certificate
- Secret store unsealing daemon
- Configure TLS encryption for kong+postgres channel
- Secure service to service communications
  - Yes to DS to Core/Appl Service when distributed
  - All?? Possibly – Bryon readout forth coming

Tech Debt

- Create and use a per service Vault token in the security services
- Blackbox tests of APIs through the API gateway
- Design work
  - How to implement HTTPS in EdgeX (that is, how to protect all service endpoints with HTTPS)
  - How to implement role-based security across our all EdgeX services.
- Eliminate remaining dependency on curl and jq in scripts/security-proxy-setup-checker.sh
- Make docker container images read-only
- Set no-new-privileges option on containers
- Develop process for security vetting of 3rd party components
- Make docker container images run as non-root users
- Streamline proxy certificate upload flow
- Secure Kong admin port with TLS
- Enable CORS for API Gateway
- Rewrite security-secret-setup to remove legacy cruft
- Pluggable random database password generation
- Add method to retrieve metadata inserted by file-token-provider
- Check the consul/registry for the configurable settings in security-secrets-setup

System Management

New feature

- CLI improvements (do we have idea of what these might be)
- Metrics collection per ADR 0006
    - Add go-mod-messaging
    - Service to push
    - Configuration
    - Instrumentation in every service to collect
- System management - storing system metrics locally
- System management - actuation based on metric change (a "rules engine" for control plane data)
- System management - add alerts and notifications (service down, metric above threshold, etc.)
- SMA support for other control plane protocols
- Kubernetes “facilitation”
    - Jim

Tech Debt

- Improving system management capabilities to include providing an asynchronous set of APIs, offering more EdgeX specific APIs, storing management metrics (for store and forward) and exporting management data to 3rd party systems??
- Look at Driving "Default Services List" via Configuration??
- Refactor to enable operations against agent via executor??
- Capture CPU metrics data for macOS??

DevOps

New feature

- Sharpen our use of SonarCloud and provide developer education around it
- ~~Automatic code formatting in CI/CD pipe~~
- ~~Include a data filter between DS and Core Data (align with and share App Service filter function if possible)~~
- ~~Produce deployment artifacts that contain multiple services (e.g. a single core service docker container versus core, metadata and command services)~~
- ~~Produce service executables that combine services (e.g. create a single core executable that is core, metadata and command all in one) via the build/make process.~~
- Set up codecov for edgex-global-pipelines

Tech Debt

- Improve performance of pipelines
- Snap build improvements
    - Snap builds should use unbuffer for better output logging
    - Update snapcraft inside docker to use new setup from snapcraft
    - Make stage-snap jobs more smart about pushing metadata
- ~~Root cause analysis of the ARM failures for blackbox-testing - Infrastructure failures~~

Certification

??

Vertical Solutions

??

Test / QA / Documentation

New feature

- User guidance on platform needs
    - More performance statistics
    - # of devices/per recommendations
    - Providing EdgeX users guidance on platform needs, sensor data throughput and deployment based on performance metrics. Specifically, with the Geneva performance testing apparatus, the EdgeX community will be able to answer these questions for the user:
        - Will EdgeX fit on my system? - size of EdgeX services, infrastructure, etc. and hardware/platform requirements
        - What is the speed of data through the system? - from device service sensor data ingestion to the rules engine and back down through command to another device service to trigger a put command, how long does this take?
        - How many "things" can be processed at a time? – with caveats on the type of thing, type of data, etc.
        - These questions need to be answered on real hardware (both Intel and ARM)

Tech Debt

- System integration / interoperability tests - Device Service read data -> Core Data -> Rules Engine or Application/Export Service -> Command
- Add unit tests/testing for global libraries. (w/ DevOps help)
- Blackbox tests to edgex-go repo?
- Blackbox tests against snaps
- Configuration testing – testing non-default and dynamic config changes

Open Horizons

- Should be a separate project in spring 2020.
- Roadmapping/direction is self determined
- Any additional next steps for EdgeX integration?