EdgeX Planning Conference
Final Scope, Action Items, To-Dos

Levski Release

May 16-19, 2022

Kamakura Release

- Still be done
  - C SDK
  - C Device Services
  - LLRP Device Service and App Service (working through snap support)
  - Docs
    - https://github.com/orgs/edgexfoundry/projects/46
    - What doesn't get done falls to Levski
  - Performance metrics report (Blog/marketing efforts associated with it)
- Issues/Concerns
  - None to date

Jakarta LTS Release

- Version 2.1.1
- Jun 8 target
- See board for what’s in https://github.com/orgs/edgexfoundry/projects/47

EdgeX Project Boards

- Work Groups to prep/transition boards to Levski by May 25th
  - Remove the Dones
  - Clean out In progress, QA Review, etc.
  - Move appropriate Icebox items back to the Backlog (leave issues not going to be addressed in the Icebox)
- Start a new Levski Documentation updates (or rename Kamakura)
  - Jim to set up Levski Docs and Kamakura Issues boards
    - ✓ Created both (as new projects)
- Do we want to explore the new style of boards
  - Multi-view
  - More customizations
  - Better issue repository access
  - ✓ Jim to research how to transition from old to new – email forthcoming
  - WG Chair decision on which board type to use this release cycle

EDGE X FOUNDRY™

Manual Testing To Do

Test	Owner/Actions
Test any configuration not already handled by new TAF tests - Test without Core Data (done) - With various Message Bus implementations (MQTT, Redis) (Done) - Without scheduler, notifications, rules engine	Jim on test without scheduler, notifications, rules engine
Test reboot of the system. Make sure EdgeX can come back up	Lenny
Test device services (with real hardware where possible)	Jim – GPIO, Modbus, SNMP Intel – LLRP (DS and AS)
Working with service authors to identify minimal tests	
Test different combinations/permutations of app functions in app services	Already covered by TAF tests
~~Test on Windows~~	
Test running with multiple services of same type (device service and app services)	Jim – device service Lenny – app service (done)

Next release cycle – handle prior to release
Add to lessons learned

Levksi Release

Make sure mention of metrics as Beta in Docs
✓ Lenny submitted a PR

- Version 2.3
  - Must still be backward compatible
  - Can add, but not remove deprecated items yet
  - Not LTS, but must consider we have LTS now on the books
- Major additions under consideration
  - Unit of measure implementation
  - More metrics
  - North to south message bus implementation (#1)
  - CPE implementation (#1)
  - EdgeX micro service authentication prototype
    - prototype OpenZiti-based solution

Maybes
- Lighter weight services
- Compile out features to make them lighter weight

Business Tasks

- Developer Evangelism
  - EdgeX Ready
    - Expanding to cover device profiles for 2.x, and from any protocol covered by EdgeX (vs just MQTT and REST)
    - EdgeX Ready badge for individuals (in addition to companies) to build “members” and attention
    - Creation of Device Profile, custom Device Service -> individuals to earn EdgeX Ready badge?
      - Requires some training prior
    - Additions/expansion of program to be covered in Outreach
  - Tech Talks (May-June)
    - Add others as topics/needs arise
  - Hackathon (fall?) July 1st go/nogo
  - Return to events in 2022 (Fall?)
- Next Level of OpenSSF badge?
  - We have entry badge; we are 30% on the way to next level (silver)
  - 4000 projects – 750 pass basic; 32 pass silver criteria; 11 at gold; https://bestpractices.coreinfrastructure.org/en/projects/1226?criteria_level=1
  - This release - gap analysis & go/nogo and prioritization (due diligence) of tasks toward silver level
    - Badge itself is byproduct but not the driver
  - Drive this task under Security WG (target due dil/gap analysis this release cycle)
    - Next Planning meeting Go/NOGo and work tasks to take on
- Do we still need an Outreach WG (and TSC voting position)?
  - Don’t combine with LF Edge (need independence)
  - Perhaps move the work into TSC
  - Bring up at TSC for further discussion – how best to accomplish Outreach work (are more effective ways to conduct the mission)

Web Site Activity the Release Cycle

- Tasks in priority order
  - Renew China Website Contract
    - Update certificates
  - Address Security Audit issues
    - Implement the CSP (content security policy – defend against scripting attacks)
    - Upgrade TLS
    - securityheaders.com and ssllabs.com both pass with an A grade
    - Part of OpenSSF audit – must pass to earn Silver
  - SEO audit/review – SEO improvements
    - Making sure China search results in cn.edgexfoundry.org
  - Review/update content
    - Review/update architecture diagrams
    - Community can do some of this work in order to keep costs down
  - Contact DB for outbound marketing
    - Register interest; get updates and newsletter
- We have a budget of $10K with $6,500 unallocated
  - Outreach WG to explore budget and tasks and work to complete as much as possible

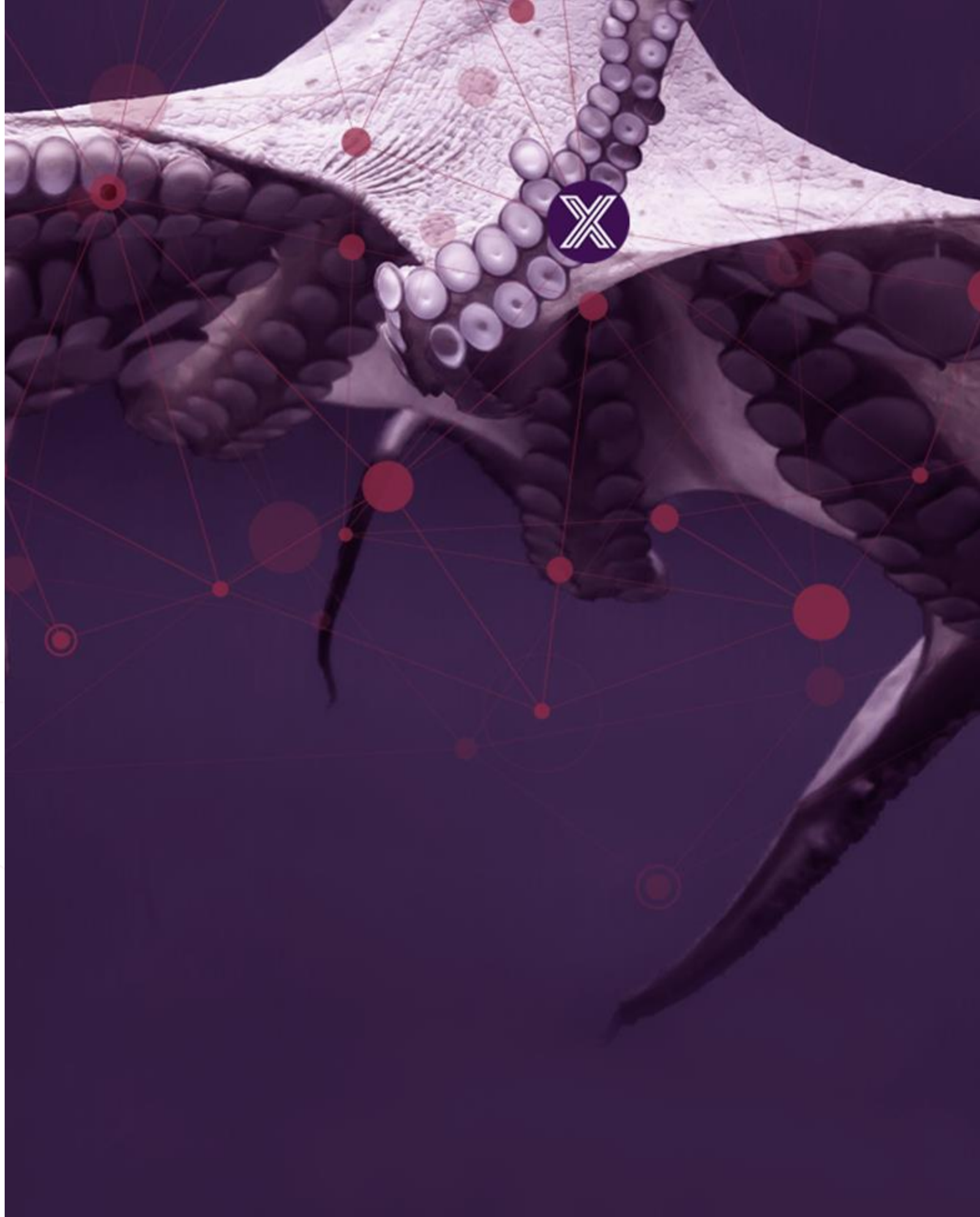
2022 EdgeX Award Winners

- Innovation Award Winners
  - Jim Wang & Bryon Nevis
  - Anthony Casagrande & Marc Fuller
  - Emilio Reyes
- Contribution Award Winners
  - Iain Anderson
  - Siggi Skulason

Award ceremony in June when trophies have been distributed

EDGE X FOUNDRY™

Cadence & Release Naming



edgexfoundry.org | @edgexfoundry

EDGE X FOUNDRY™

Cadence Check

- April & Oct/Nov remain target release months
  - Kamakura release – May 2022
  - Levski release – Nov 2022
  - Minnesota release – May 2023
  - Napa release – Nov 2023
  - Odessa release - May 2024
- Venue for next F2F Meeting (Nov 2022)
  - With travel restrictions lifted – everyone in favor or a F2F
  - Volunteers, suggestions on where to hold this meeting?
- Conferences – marketing committee update
  - Target events to return to and when

O release named by Mengyi and Farshid

Release Timing – No changes

- Formal release/planning schedule is now:
- Generally attempt to select release date about 2 months in advance of the release
  - Typically early March and early September for spring/fall releases
  - Release date preferred to have be a Wednesday
  - Adjust per circumstances and TSC review
- Release schedule for minor release
  - Freeze date 2 weeks in advance of release date
  - Prewire, Thursday prior to freeze date
- Release schedule for major or LTS release
  - Freeze date 3 weeks in advance of release date
  - Prewire, Thursday prior to freeze date
- Planning meeting – generally the week following the release
  - Virtual meeting: Monday – Thursday with Friday for training/non-conference
  - In person: Tuesday-Thursday with Thursday afternoon for training/non-conference

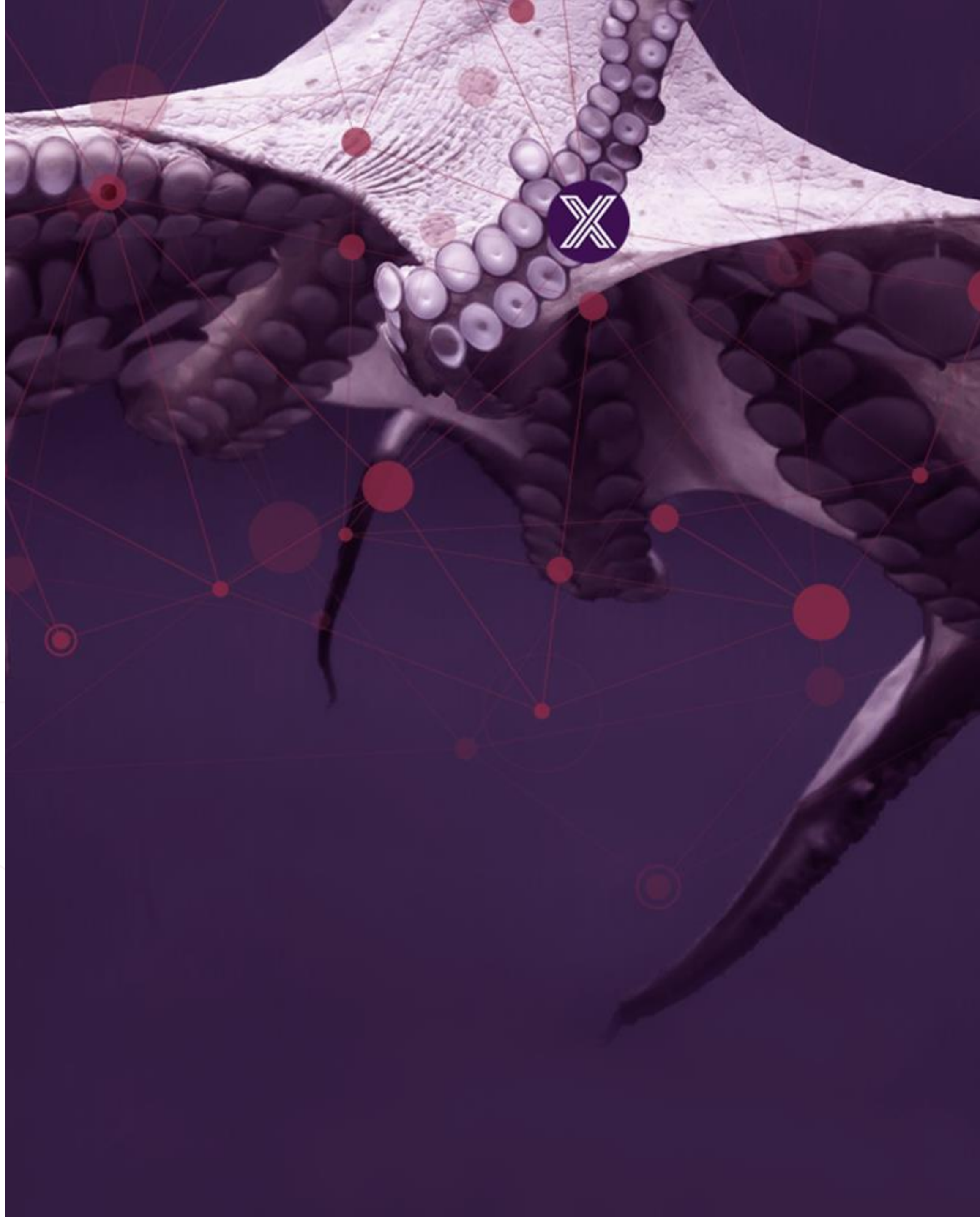
IF travel necessary – planning meeting may need to be a week out

Include manual testing in the release schedule for Levski

EdgeX Meeting Times – No Changes

- Mondays
  - Device Services @ 8am PDT
  - Outreach/Marketing @ 9am PDT alternate weeks
  - Architect's meeting once a month @ 10am PDT
- Tuesdays
  - Occasional Adopter series or tech talk meeting @ 8am PDT
  - Application Services WG @ 3:30PM PDT alternate weeks
- Wednesdays
  - TSC @ 8am PDT
  - Security @ 9am PDT
- Thursdays
  - Core @ 8am PDT
  - DevOps @ 9am PDT
- Friday
  - No meetings

Process Improvements



ADR Process

- Is it working for us?
    - Does it need improvements?
- Farshid's recommendations:
    - Need to cover requirements better
    - Separate requirements and design
- Need to avoid getting into implementation details (how?)
- Decision – using Farshid's suggestions:
    - Create a 3 step process: document use case/requirements, document design (ADR), implement
    - Farshid to draft use case/requirements template
    - Jim to update ADR process documents in the Wiki
    - Jim to create ADR template
    - Update Docs
        - Add versions and change log to ADRs
        - Add UC/Requirements section to docs
    - Use the new process in Levski

Real Hardware Testing

- How can EdgeX validate the platform with real hardware?
- How can the project get real hardware experiences/feedback?
- Can we establish a "framework" for outside testing with real hardware?
- Can we identify a champion per device service?
- Can we leverage "plug-fest" idea (ex: W3C) from other projects?
- Decision: Device Service WG – take up as a discussion item and potentially formalize a proposal to the community

3rd party library support

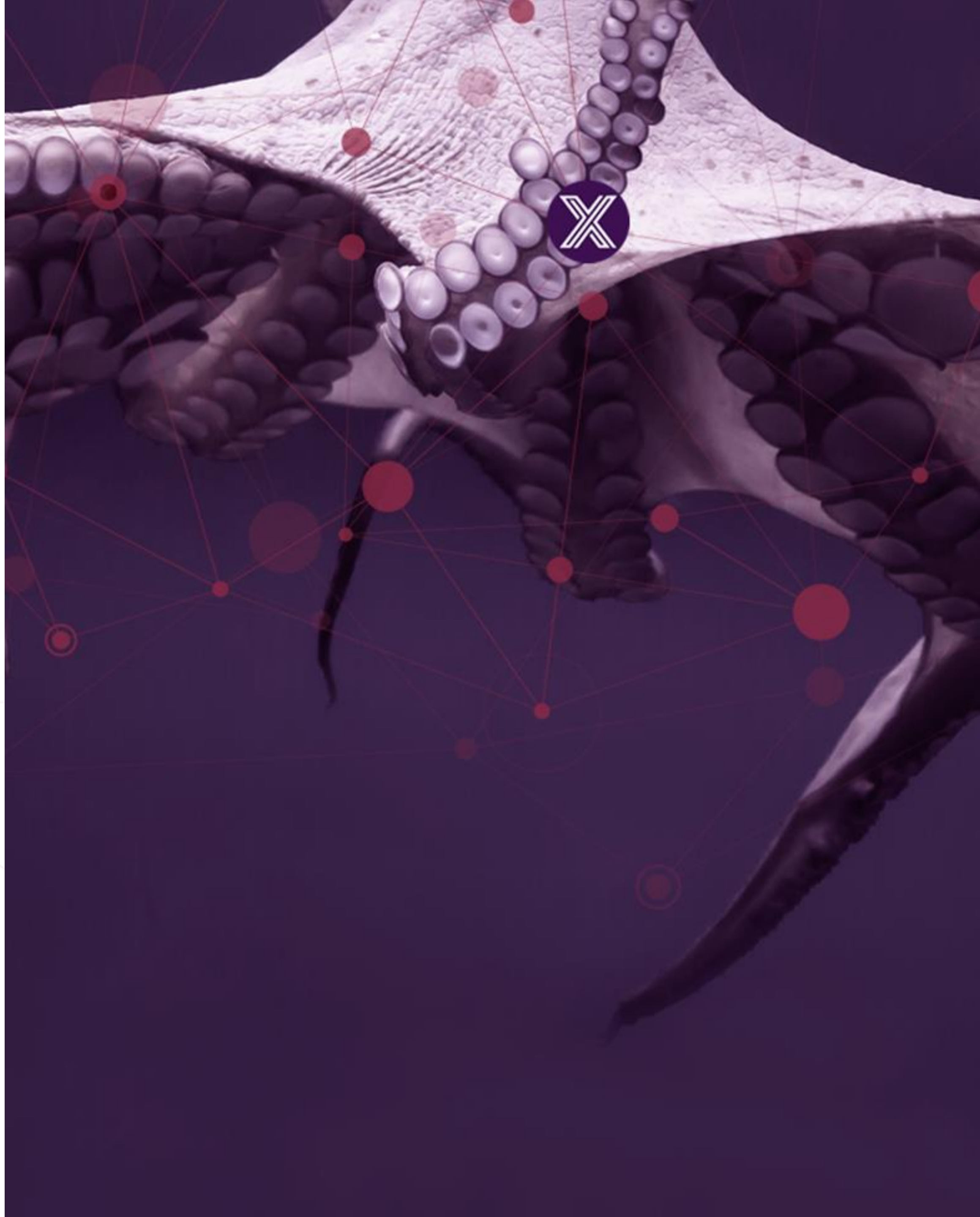
- We have a process in place to review and vet 3rd party libraries being used by the project
  - We perform a “paper study” to explore the use, ongoing development status, upkeep, and contributors to a library
- In many cases, 3rd party libraries have less than ideal maintenance records, use, etc.
  - But often, there are few or no alternatives to using a library we might otherwise reject
- What to do about this going forward?
  - Fork and maintain libraries (at least those with potential support issues) in EdgeX
    - Only do this when the library team is not responding to an issue
  - Try to provide assistance to these library projects
    - This is our main recourse for issues
  - Do nothing – hope for the best; react to issues as they arise
  - We have and maintain our vendor copy
    - Default – as long as there are no issues
- Decision: apply more scrutiny to non-version libs but not a hardline to refuse them

Github Discussions

- Use Github Discussions (https://docs.github.com/en/discussions) for Q&A. Slack is good for community engagement and announcements. But when it comes to Q&A, there are currently a large number of questions with great answers, sadly lost in Slack history. Slack is hard to access and the content isn't indexed by search engines as far as I know. There is no easy way to organize or search efficiently per topic. On the other hand, Github Discussions which is similar to Stack Overflow but attached to the source code, allows creating a good knowledge base which the community can quickly search and benefit from.
- Example projects using
  - https://github.com/redis/redis/discussions
  - https://github.com/lf-edge/ekuiper/discussions
  - https://github.com/Kong/kong/discussions
- Decision: setup github discussion for device-service-go trial balloon through this release cycle.
  - Post message on Slack to tell people to go to new Github Discussions
- Future: consider consolidated “discussion repo” and move all Q&A to this repo discussion board

EDGE X FOUNDRY™

Architectural Discussions/Decisions



In Scope

To be worked on as specified in the notes during the Levski release cycle

★ Add task work to appropriate WG scope

Metrics Collection

- ADR - https://github.com/edgexfoundry/edgex-docs/pull/268
  - DONE!
- What telemetry do we collect?
  - Impact to size
  - Laundry list in the ADR
  - Revisit scope and what additions we make in Levski (if any)
- Decision:
  - Each Work Group to review metric list in ADR– determine which go in this release
    - Can add other metrics
  - List determination to be accomplished before the end of June
  - Implementation – in scope for this release
  - Move this feature out of Beta with Levski

Add task work to appropriate WG scope

EDGE X FOUNDRY™

Last Connected & Last Reported

- Bugs discovered : (1151, 1150) these fields on device service and device have not been populated since ? (perhaps Barcelona but at least Edinburgh)
  - Obviously the empty fields need to be addressed
  - Options:
    - Just deprecate those fields (not recommended)
    - Use Levski release to fix these
- Concerns about "chattiness" on updating Device Service timestamps
- Is there a useful distinction between these dates – is this used?
- Decision
  - Use new metrics telemetry to track last connected and last reported (possible named differently) for device (not for device service)
  - When CPE is designed, it should be possible for a device service developer to add the trigger of a CPE when a device is reported/connected in device service code
  - Deprecate lastConnected and lastReported fields in both device service and device

★ Add task work to appropriate WG scope

GUI and calls to SMA

- Today, the GUI calls on the SMA for service metrics, configuration, etc.
  - SMA is deprecated (although won't be removed until EdgeX 3)
  - Do we want GUI to use a different means to get metrics/config/etc.?
  - Per https://github.com/edgexfoundry/edgex-ui-go/issues/486
- Calling on each service directly presents some challenges
  - Service name from the registry isn't directly mapped to metrics APIs
  - GUI doesn't call on all services today (one of the reasons for the SMA)
  - Addition of new service, GUI would have to be notified or do a refresh of service list periodically
  - GUI wouldn't know the proxy path when Kong is deployed/used (GUI is outside of Kong)
- Suggested
  - Service registration includes URL to get metrics and other SMA related data for each service
- Decision
  - Mark "System" page deprecated in the GUI
  - Stretch goal: have new page that provides new service metrics telemetry data (at least a start) for Levski
  - Long term, future release – what is our reference implementation of external control tools of choice for start/stop/restart
    - Options: use Portainer, Docker CLI, snap CLI, … and provide documentation/examples of how-to on these
    - Jim to add to backlog; explore with GUI team some options

NATs (back again?)

- Add NATS implementation for MessageClient interface
  - https://github.com/edgexfoundry/go-mod-messaging/issues/37
  - https://github.com/edgexfoundry/go-mod-messaging/pull/137
  - Size concerns – add size to the binaries
    - Conditional compilation "trick" may be a way address this (this is EdgeX Lite concept)
    - Size hit: 1.5MB hit to add per service (5-20%?)
  - Rationale
    - lightweight protocol with potential for on-device use
    - ability to use common messaging infrastructure at all levels of stack
    - protocol-level metadata allows 'native' conveyance of the EdgeX message envelope.
    - flexible deployment options for HA in some settings
  - Requested by at least one adopter and discussed in the Kamakura planning meeting (others by Slack more recently)
  - NATS JetStream?? How does this apply/differ from NATS at large?
  - You are going to want to secure it – what does that look like?
  - Biggest benefits
    - Elimination of header decode/encode with our current implementation
    - MQTT 5 – provide the bridging features and some of what NATs does
    - Side note: look at MQTT 5 features for future releases (research needed) – Jim to add to backlog
- Decision
  - Make experimental feature – build and enable (like 0MQ)
  - Redis PubSub and MQTT still there by default
  - We should collect some performance metrics comparing options as release comes out
  - Future release (EdgeX 3.0) – explore NATS or MQTT5 as base /default implementation
  - Cross cutting concern (alt build files, etc.)

EDGE X FOUNDRY™

Documentation

- Should they be oriented around use of Docker/Docker Compose or native services
  - Impacted areas
    - Getting Started (most)
    - Security (most)
    - Microservices (some)
- First – what's the right approach – think like an adopter
- Second – how do we get it done
- Would we need additional docs to then address Docker/Docker Compose?
  - Separate doc sets?
  - Multiple releases to do? EdgeX 3.0 issue?

Documentation Discussion

- What's the "easy button" – quick start
  - Docker/Docker Compose lends itself well to that (if they are comfortable with Docker)
  - Some like the native command line
  - What does EdgeX do and what does it look like? How do you interact with it? Docker Compose worked for that. Then people say "let's do something else with it."
- Decision:
  - Security docs are currently all about Docker Compose; let's add documentation to help with security outside of Docker Compose
  - Using snaps with Docker is a reality; let's add some help to provide for this hybrid environment
  - Use this cycle to identify those areas where we need more non-Compose options
  - The hybrid development mode is important and how most developers work
    - Some are "gotchas" in this area
    - We need to identify those and provide documentation help

Add task work to appropriate WG scope

EDGE X FOUNDRY

Issues associated with Last Connected & Last Reported

- If we are going into the lastConnected and lastReported section of the metadata and DS code, might there be some other tech debt we want to address?
  - Command chaining
    - device-sdk-go #26 and device-sdk-c #3
    - The fields specifying these operations were dropped in the profile
    - Rework for v2 so there's an implication there that these will be WONTFIX
  - Configurable size limits in device service
    - MaxCmdValueLen is a configuration setting in the Go SDK that does nothing. It is labelled as "MaxCmdValueLen is the maximum string length of a command parameter or result (including the value descriptor name) that can be returned by a Driver."
    - In the C SDK this is called MaxCmdResultLen and it doesn't do anything either
- Decisions:
  - We won't support command chaining. Close this issue and make any code changes necessary to end this option
  - On size limits, these should be used going forward:
    - MaxRequestSize – limits request size
    - MaxEventSize – on device service top level config setting; make this consistent across SDKs; clean up other config

★ Add task work to appropriate WG scope

Build Time Issues

- "make docker" takes a lot of time due to continual rebuilding of go module cache between dockers
- With SPIFFE/SPIRE and other related items, builds are taking a long time
  - 6 minutes for Intel builds
  - Approaching an hour on ARM builds
- Future discussions about how to do other builds for other needs (with/without security, etc.) could make this worse in the future
- Are there things we can do or try to reduce build times
- Decision: in scope for Levski
  - Short term – having our own SPIFFE/SPIRE Nexus image for use in builds
    Long term – use SPIFFE/SPIRE artifact that is coming soon
  - DevOps scope item, with a t-shirt size of medium

go-mod dependency “pains” in build locally

- Need developer build common cache for Docker images
- Needs some research/education assistance
- Possible – implications to our Makefile/Dockerfile??
- Task for DevOps with presentation in WG meeting to help

★ Add task work to appropriate WG scope

EDGE X FOUNDRY™

Securing the internal message bus – specific to MQTT

- Communications between services – message bus
- How to generate credentials and tell the services about these
- How to initialize the MQTT secrets for the internal message bus comms
  - This is just about initializing the broker and putting the credentials can get at the in Vault
  - Question: why is an alternative secure implementation for MQTT – we already have secure Redis Pub/Sub? Should we consider deprecating Redis Pub/Sub if it is not meeting needs?
  - MQTT does provide more "knobs" (QoS, durability, keep alive, etc.) for using MQTT vs Redis today
- Decision:
  - Crawl - Use existing mechanisms (as we do for Redis) to push (inject) known secrets (MQTT user/pass) to each service and the broker. And start the broker started with security enabled. Broker is selected by EdgeX - Mosquitto.
  - Future – wait for more user requirements. Possible work with CA cert in addition to user/pass
  - In scope for Levski; support already there; just getting the secret in place Medium in Security

Add task work to appropriate WG scope

NATs

- Add NATS implementation for MessageClient interface
- https://github.com/edgexfoundry/go-mod-messaging/issues/37
- https://github.com/edgexfoundry/go-mod-messaging/pull/137
- Size concerns – add size to the binaries [Size hit: 1.5MB hit to add per service (5-20%?)]
  - Conditional compilation "trick" may be a way address this (this is EdgeX Lite concept)
- Rationale
  - lightweight protocol with potential for on-device use
  - ability to use common messaging infrastructure at all levels of stack
  - protocol-level metadata allows 'native' conveyance of the EdgeX message envelope.
  - flexible deployment options for HA in some settings
  - Biggest benefit – elimination of header decode/encode with our current impl
    - MQTT 5 – provide the bridging features and some of what NATs does
    - Side note: look at MQTT 5 features for future releases (research for Levski)
- Requested by at least one adopter and discussed in the Kamakura planning meeting (others by Slack more recently)
- NATS JetStream?? How does this apply/differ from NATS at large?
- You are going to want to secure it – what does that look like?
- Decision: in scope for Levski
  - Make experimental feature – build and enable (like 0MQ)
  - RedisPubSub and MQTT still their by default
  - We should collect some performance metrics comparing options as release comes out
  - Future release (EdgeX 3.0) – explore NATS or MQTT5 as base /default implementation
  - Cross cutting concern (alt build files, etc.)

★ Add task work to appropriate WG scope

EDGE X FOUNDRY™

Ziti for underlying service-service security?

- Per Ziti presentation at Security WG meeting
- https://openziti.github.io/
- Potential alignment with NATs
- Decision:
  - Create a prototype with OpenZiti during this release cycle

Lite Builds

- Provide capability to build EdgeX services w/o certain capabilities too have a smaller footprint
  - Same as we did with Delay Start capability and ZMQ
  - Mainly focus on Security capabilities that are currently built into all the services that cause their footprint to increase
  - Gives adopters ability to build a lite version when they are not using secure mode.
  - Can be done for other capabilities also that we identify as contributing to a larger footprint that some low resource deployments don't need.
  - Could do this with the NATS implementation. Out by default, but could be added via build tag. Requires all services (which use MessageBus) that an adopter wants to use to be rebuilt with NATS included.
- Decision:
  - Defer until we have stats and input from adopters
  - Explore goweight

Out of Scope for Levski

Returned to the backlog

Global Configuration

- https://github.com/edgexfoundry/edgex-docs/issues/580
- Is it time to consider some sort of global configuration?
  - We have a lot of duplication
  - Requires a lot of config to be touched when an adopter wants to change one item
    - Examples: topic name, logging
    - Turn on/off metrics collection in the future
- Difficulty – where would the configuration go?
- Considerations: with or without Consul, with or without security (and Vault)
- Hard to solve – one of the downsides of microservices
- EdgeX 3.0 consideration – would likely be non-backward compatible
- Architecturally significant – calls for an ADR
- Has to be an enhancement to config provider abstraction/interface
- Need to start design before EdgeX 3.0, but doesn't need to start with Levski
- Idea: common could be loaded from a file

URI reference to point to config/profile/etc.

- Came up as part of UoM ADR discussion
- https://github.com/orgs/edgexfoundry/projects/48#card-77954532
- We are looking to use a URI to specify an external configuration file in the future
  - Allowing for file, HTTP, HTTPS or other protocol access support.
  - URI mechanism may be used to point to device profiles, configuration files, UoM and other "configuration" information in the future.
  - This would even allow multiple EdgeX instances to use the same configuration or profile (multiple EdgeX instances using the same URI to use a shared profile for example).
- Helps as EdgeX instances are scaled
- May need EdgeX 3.0 to implement (backward compatibility)
  - It could be done in a backward compatible way
  - Need requirements/use case better documented
- Requirements/ADR needed
- Global use – falls under discussion and use as part of Global Configuration
- Just using it for UoM – do it as an add-on after UoM is implemented (stretch goal for UoM impl)

Bring your own Vault

- Allowing users to potentially run Vault in a cloud
- One customer did run-your-own; it is physically possible
  - Had to rip open secret store setup to make it happen (lot of work)
  - Bootstrapping is the complex part
  - This is really technical debt refactoring (not interfaced/abstracted enough right now)
- “We should perhaps be thinking about enabling the "bring-your-own Vault" approach, similar to what I proposed for SPIFFE/SPIRE support. This would allow users to potentially run the Vault instance in a cloud where it is more secure.” – Bryon N

Device Discovery

- Worth a relook and / or additions
- Protocol specific today
- Implementation the responsibility of individual device service providers
- Do we need additional hooks to facilitate secure device onboarding?
- We didn’t do a good job of how to do this that provide assistance to developers (LLRP as example)
    - Better examples
    - Use lessons learned
    - Camera services are doing
- Chance to do a better job than what we did in the first go-around
    - Tech debt
- More input from developers that have gone through it
- Allow for more non-static (vs static) capability of provisioning
- Unintuitive approach right now – minimally need more documentation (in scope)
    - Is there a better design that makes it more intuitive – an EdgeX 3.0 issue
    - Is there something with ProvisionWatchers that would help with the design
- Push design to a later release

I18n/L10n metadata

- https://github.com/edgexfoundry/edgex-go/issues/3935
- An adopter has requested that things like device names be allowed to contain other language characters. Specifically, in this instance, Vietnamese characters. Presumably, much more of EdgeX could be internationalized or localized.
- May be applicable to all services working with user defined JSON or YAML (like device profiles).
- Validation of “names” and such would have to be updated to allow for I18n/l10n
  - Think this might be in a single place
- How would this work with URLs, topic names, etc.?
- Input from China – how about a name to character of choice translation kept somewhere that can be used for UX purposes?
  - Otherwise I18N/L10N not a big issue or need right now
- Add this to the backlog, but no action today

Redis Time Series

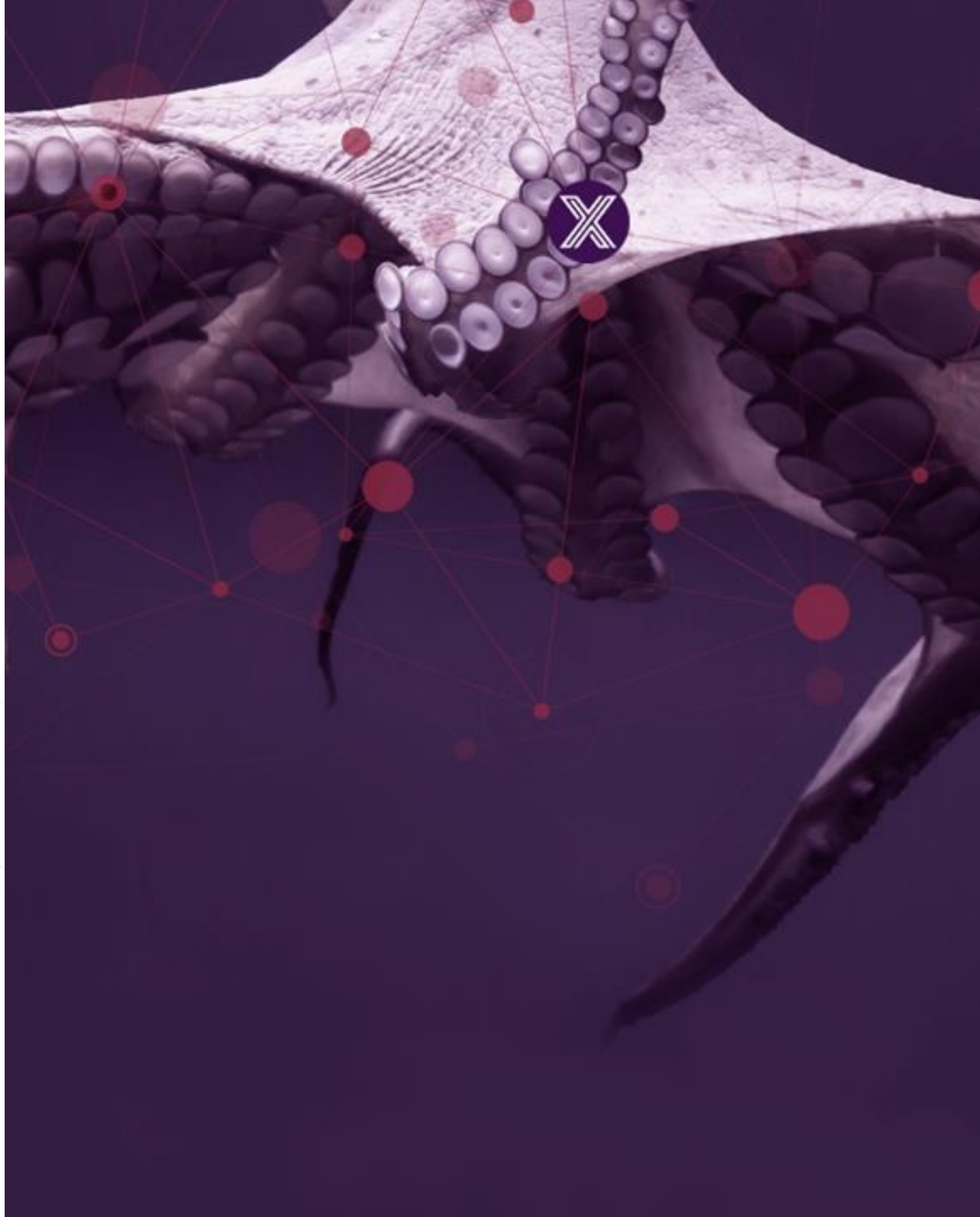
- Use of Redis TimeSeries (https://redis.io/docs/stack/timeseries/) instead of key value as core-data backend.
  - This will undoubtedly improve performance.
  - Also, it would potentially allow Grafana's Redis plugin (https://grafana.com/grafana/plugins/redis-datasource/) to visualize EdgeX's data without having to rely on other components.
- Put in the backlog; welcome contribution on the part of the community
  - Consideration: which TimeSeries is best choice? And how many DBs do you need/want?

Remote Device Service

- Device services receive commands over HTTP. This works well when the device service is deployed along core services, but starts to get complicated when it is deployed on another node.
  - We all know that in many use cases, device services will have to be distributed.
  - How to access the HTTP endpoints remotely and securely? SSH tunnels are one way but they are rather a workaround than a robust solution (this example uses two SSH tunnels https://docs.edgexfoundry.org/2.2/security/Ch-RemoteDeviceServices/).
  - The Spiffe ADR (https://docs.edgexfoundry.org/2.2/design/adr/security/0020-spiffe/) mentions the we don't "support distributed services" and introduced a mechanism to authenticate remove services.
  - While the authentication part is resolved, we still have to rely on SSH tunnels to make the EdgeX connectivity work, as if the device was deployed locally.
  - From personal experience, connectivity issues (e.g. cellular 3G connection) will make SSH tunnels a very unreliable solution and nothing suitable for production. Opening the ports to public via a proxy that can perform access control is another way but a complicated one, assuming that core-command can authenticate with the secured DS endpoint when making requests.
  - How about replacing the functionality described as "Receive and react to REST based actuation commands" (https://docs.edgexfoundry.org/2.2/microservices/device/Ch-DeviceServices/#device-service-functionality) with a pull model, where DS subscribes to commands without having to to expose any HTTP endpoint at all? An MQTT-based message bus can add QoS and make this a robust and versatile way of connecting remote device services with EdgeX. I think that's when SPIFFE will be able to play an important role. (edited)
- Discussion
  - We have solved delayed start but not distributed services (with SPIFFE/SPIRE)
  - We are evolving toward this; but we could do this if DS was talking MQTT (provided MQTT bus comms is secured – which it is not)
  - Could be configurable (turning off HTTP/REST)
  - Ziti could solve this
  - Decision: use baby steps toward this and then also wait and see the outcomes of Ziti prototype

EDGE X FOUNDRY™

Levski Scope



General / Cross Cutting

- Upgrade Go (1.18, 1.19?)
  - Generics important
  - 1.19 August – wait and see
  - 1.18 as soon as possible
  - Also bump the latest versions of Kong, Consul, Redis, Vault, eKuiper, Postgres – with research and test in each case (some of these are major issues)
- UoM implementation
  - Small / Medium (we think this is isolated to metadata)
  - Stretch goal: use of URI just for unit of measure references
- North-South message bus implementation
  - Large (complexity – command, app, DS, sdks, clients impacted too)
  - May be extra large and need to be developed across multiple releases - TBD
- Cloud templates need updating (currently Hanoi based)
  - Deprecate the cloud templates
  - Small
  - Update the EdgeX Ready Wiki (provide references to other options/documentation)
- Additional Metrics (per arch discussion)
  - Each Work Group to review metric list in ADR– determine which go in this release
    - Can add other metrics
  - List determination to be accomplished before the end of June
  - Implementation – in scope for this release
  - Move this feature out of Beta with Levski
- NATs Implementation
  - Make experimental feature – build and enable (like 0MQ)
  - Redis PubSub and MQTT still there by default
  - We should collect some performance metrics comparing options as release comes out
  - Cross cutting concern (alt build files, etc.)
  - Large (but with PRs already submitted to use as starting point)
- Explore goweight to see impact of libs
  - May create additional "Lite" builds based on outcomes

Core

- CPE
  - ADR process; we'd like impl too, but not sure of its size; high desire for this release
  - ADR – small (follow the new requirements/design process)
- Core Data Cache
  - ADR process; implementation dependent on what comes from ADR
  - Small (follow the new requirements/design process) – Eaton to take the lead on ADR
  - NATs exploration may have some impact
- Parent/child device relationships
  - ADR process – Eaton commits to ADR process this cycle
  - Use case/requirements important as this feature has been considered several times
  - Could labels be used to cover this requirement?
  - We have attempted this several times in EdgeX (even removed it from original impl)
- Deep Copy of message bus info (#4021)
  - Tech debt; just going to get done in Levski (in scope but not worthy of discussion)
- Deprecate lastConnected and lastReported fields in both device service and device (per Arch discussion)
  - Small
  - Implemented device lastConnected/lastReported via metrics telemtry

Core Continued

- K8s support
  - No specific/targeted additions/updates in scope; individual contributor updates/fixes/additions possible
  - May include some small tweaks to help facilitate – but not anticipating big impact to code/platform
  - Interest on Intel's part as part of internal work/needs
- CLI
  - No resource availability at this time – so nothing this cycle
- GUI
  - Add unit tests (Selenium) to CI/CD
  - System page deprecated (per Arch discussion)
  - Stretch goal: add something around new metrics telemetry
- Test QA
  - Always CBOR option for performance – In scope but lower priority
    - Test JSON encoding vs CBOR encoding for all event traffic
    - Try to get the numbers so we can make a decision (need to check on REST and message bus side)
    - Future tuning options – how to lower JSON overhead – but not part of this research – Jim to add this to backlog
      - Other tests – REST vs MQTT vs Redis vs NATS…
      - Test impact of number of readings per event
      - Perhaps we need a "harness" at our disposal to be able to check this type of stuff easier/quicker
    - Small to medium in scope
  - Automated tests for delayed start
    - A number of ways this could be accomplished – need to design what the test would look like (ex: use delayed start app or ds)
    - Design options – Security WG to lay out what is desired ; then have test team implement

Non scope related notes:
- MQTT DS needs to be redesigned.
  - Needs to support multiple readings
  - Add to backlog and EdgeX 3.0 tasks – Jim to add
- C++ SDK
  - Eaton to explore
  - Seek community guidance as desired

Device Services

- Device Services work to be done this cycle
  - Deprecate / abandon UART
  - Refresh/update: BACNet, CoAP, Grove
  - New: ONVIF/USB
  - Deprecate old camera
- Metrics / Telemetry collection in C services (per ADR) – implement metrics ADR in C SDK
- Close command chaining issue
- Fix configuration size limits
  - MaxRequestSize – limits request size
  - MaxEventSize – on device service top level config setting; make this consistent across SDKs; clean up other config

EDGE X FOUNDRY™

Application Services

- Some low priority backlog items
- Record and replay service
    - ADR process – outline requirements this cycle

Security

- Micro service authentication
  - Prototype Ziti-based solution and modify micro service authentication
    - Large in scope
    - Ziti Potential replacement for Kong
    - Change/update the ADR based on results of prototype
    - Possible update/change in our proxy/API gateway implementation as a whole
- Finish hardening Consul Security
  - Small/Medium
  - Changes to secret store setup and other security services (no changes to services themselves); also need to tighten Consul bootstrap
  - Give each service its own policy so it can read its own configuration (we have given each service Consul access but not scoped its access)
  - Service X can get at Service Y config today
- SPIRE workload attestation agent for snaps
  - Needed for delayed start services in snap
  - Medium (based on complexity and possibly custom implementation work)
- Secret generation for MQTT
  - Injecting configuration into it for authentication into the default MQTT broker (for all MQTT flows)
  - Creating new “known” secret for configuration (that must be distributed to all services)
  - Check MQTT libraries we use support this
  - Check/address eKuiper configuration that we inject can do this – Lenny to take this up in App WG (CCC)
- Secure internal MQTT comms
- Use existing mechanisms (as we do for Redis) to push (inject) known secrets (MQTT user/pass) to each service and the broker.
  - Start the broker started with security enabled.
  - Use Mosquitto broker

DevOps

- htmlproofer checks on each edgex-docs PR
  - Plugin in for yaml disabled – task to enable it in the pipeline always
  - Small
- Makedoc – support for variables
  - Small
  - Research first
    - Understand impacts on Htmlproofer
    - This would help with docs as well as API docs
- Automate version bump in swagger api files on release, i.e. https://github.com/edgexfoundry/edgex-go/blob/main/openapi/v2/*.yaml (Ask from Lenny)
  - Small
  - Modification to global libraries; adding a new step in the release process
  - Done manually today
- Assist GUI team automate testing – Jim to coordinate with GUI
  - Size TBD
- Create a consistent make version target in the makefile
  - snaps and CI/CD share the same version
  - Requires touching most repositories
  - Medium
- Having our own SPIFFE/SPIRE Nexus image for use in builds
  - Medium

Snaps

- Upgrade snaps base to core22 (built from Ubuntu 22.04 LTS)
  - Small
  - Just going to each repo and touching some files
- Guide on headless deployment of EdgeX on Ubuntu Core
  - Medium – add as an example in documentation
  - Guide to create an OS image (Ubuntu Core)
  - Bootstraps everything the user needs
  - Similar to the dev kit for snap
- Standalone device-virtual snap
  - Small
  - Nice to have - the delayed start for snaps
  - Have it as a stand alone snap
  - Side issue: support services and delayed start

Miscellaneous

- Tool/script to create new device or application services
  - First step – create templates
    - Done App Service template already; develop the Go SDK template this release
- Examples
  - RP4 (PR already pending from Alex – review and merge)
  - Camera service – app service to manage the cameras (Intel doing)
- Docs
  - Why CNCF approach/Kubernetes is partially supported
  - The “tony” DS documentation (#598) needs
  - Improve documentation where focus is on use of Docker-Compose (security, etc.)
    - Identify areas of improvement first
  - Embed Swagger docs in EdgeX Docs
    - Restructure API docs in the docs
    - Small/Medium
    - Farshid to tackle first page trial
    - Lenny/Jim to restructure and replication across all pages

Note:
Support for RISC-V/64 bit – Jim to add to backlog

Lessons Learned in Kamakura

- Positives
  - Docs: Incremental adds to automated release
    - More done in automated vs manual fashion for docs
  - DevOps – more automated; easier; automated release of compose
  - Scheduling – much better without overlap with planning
  - Security – hit our targets for the most part
  - Testing identified issues early/ ahead of the release
    - Identified the size issue quickly (smoke tests)
  - Kanban Board maintenance
    - More consistent use
  - Lenny in crisis switch mode without losing anything

- Areas of improvement
  - ADR process
    - “Farshid” use case approach promises to fix
    - Still an issue (carry over from Jakarta)
    - Decisions still taking too long
  - Lack of critical testing usually bites us
    - No tests to check delayed start, etc.
  - May not have gone deep enough in our implementation considerations
    - Ex: delayed start impl
  - All the steps in release and post release processes need to be better documented
    - Ex: when/how to do change logs, manual testing, etc.
    - Make sure we don’t forget something

Long-term Roadmap

- Backlog
  - https://wiki.edgexfoundry.org/display/FA/Backlog
- EdgeX 3.0 Project
  - https://github.com/orgs/edgexfoundry/projects/48
- Open discussion about next EdgeX 3.0 and LTS
  - Strawman schedule
    - 3.0 Spring 2023
    - 3.1 Fall 2023
    - Need to start to triage/winnowing of the Edge 3.0 list at the next architect's meeting
- What would we see as likely big features in Minnesota?
  - EdgeX 3.0 features
  - May be that the 3.0 release may have to slip past spring
- We need more information from adopters about LTS and our release schedule
  - Add something to the Web site and documentation that takes people to a survey
  - Work this through Outreach WG

Planning Meeting Lessons Learned

- Release Planning Meeting
  - Any lessons learned?
  - Any thing that could be done better?
  - Start doing, continue doing, stop doing
  - What worked well and what did not?
- If we shift to F2F for the next meeting, what would we do different?
- Discussion
  - Desire to return to F2F; wait and see through summer and conference feedback
  - Currently well paced; good division across ½ days
  - Understanding the use cases before the discussion – maybe making requirements definition a necessity before topic is introduced (architecture topic in particular)
    - Do this part of the pre-wire
  - Get more of the voice of the customer in these meetings.
    - Seek a volunteer for dev advocate