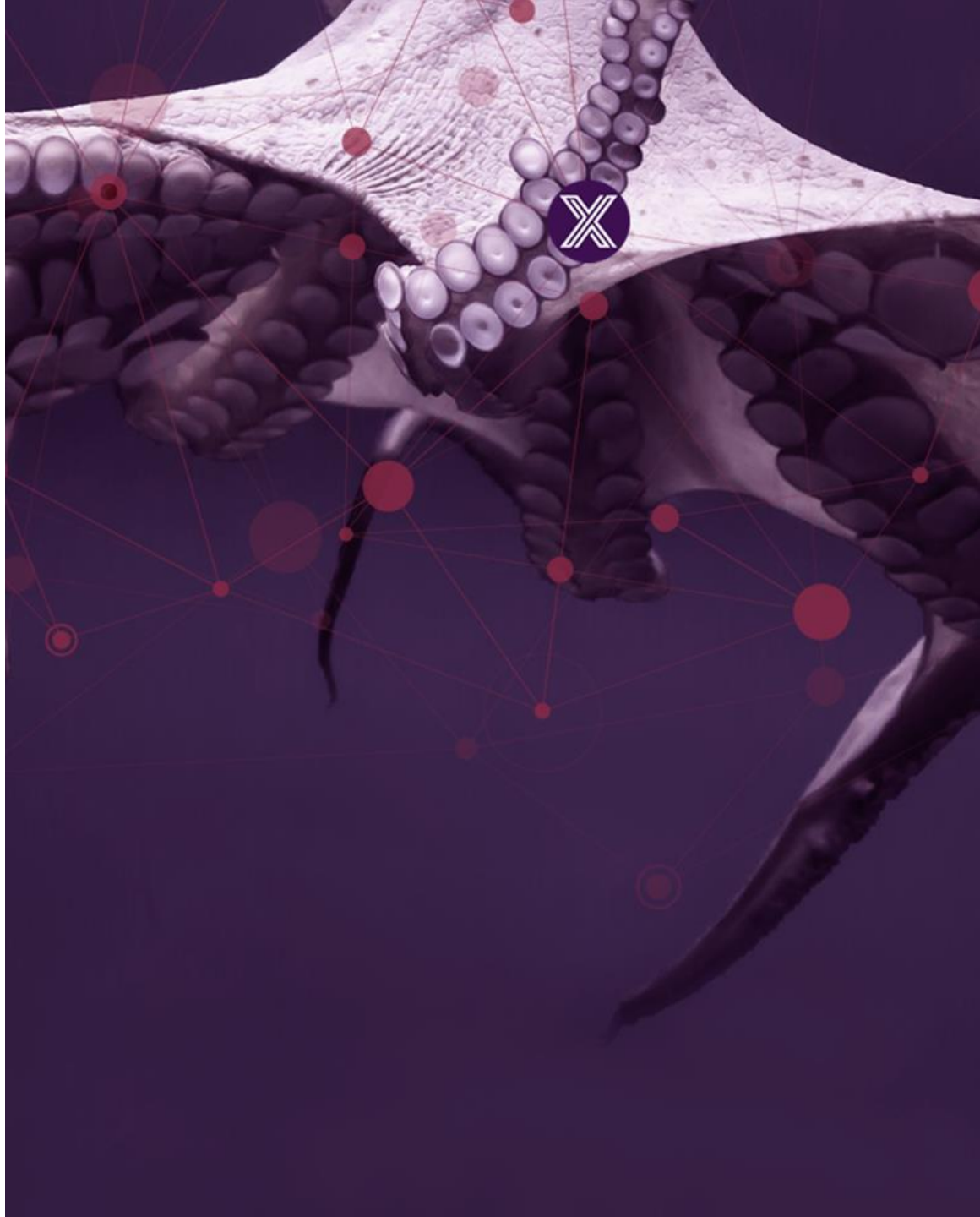
Geneva F2F Scope and Architectural Topics 10/3/19

Jim White

EDGE X FOUNDRY™

Geneva Scoping



Geneva Scoping

- The next slides contain early Geneva scope from:
  - WG leads
  - Project backlog
  - Prior F2F/roadmap sessions
- Too much for Geneva!
- Without deciding everything in this pre-wire meeting, is there consensus for some candidate tasks that can be immediately…
  - Moved to high priority for Geneva
  - Moved to the general backlog or other release

Geneva Release (Early Inclusion List)

- General/Cross Cutting Items
  - * - Alternate (to ZMQ) message bus (as default)
  - Config watcher on all services to react to config changes
  - Error Handling Refactoring as suggested by Anthony Bonafide presented in Core WG (22-Aug-2019)
  - * - Configuration – consistency and standards across EdgeX services
- Application Services
  - Support batch mode (store and forward based on a schedule)
  - * - Support for multi-channels and how pushed is then marked.
    - In other words, have multiple “pipes” hooked to the outbound channel from Core Data and allow the event to be sent to multiple endpoints – and then how does the Event’s marked pushed get set?
  - Additional language SDKs (or how to support alternate language export with single Go SDK)
  - Include configurable application service libs in application service SDK (to better support cloud exports)
  - * - Feed application services back into core data (example: aggregate readings, separate readings one per event, perform transformation from F to C on temp and return these to core)
  - * - Support Cloud Events
- Supporting Services
  - Implement the non-backward compatible scheduler API change that Eric Cotter put forth
- Core Services
  - Separating configuration/registration APIs and modules (go-mod-registry)
  - Explicit Request and Response types for API endpoints; these would be defined in go-mod-core-contracts (effects a lot of the system to include DS)
  - Elimination or separation of value descriptors (moving the type information into the reading); min, max, formatting of VD goes someplace else)
  - ~~Querystring support in the Event payload~~ (per Iain A – 9/30/19)
  - Configuration.toml structural changes as warranted
  - * - Core Services Monolithic deployment
  - Remove legacy fields in metadata (ProvisionWatcher: OperatingState, PropertyValue: Precision, Size, ResourceOperation: Resource, Object, ProfileProperty: Units becomes String)
  - Reduce boilerplate in coreCommands; all it really needs for each command is the name and whether it's get, set, or both?

Items on this list should not be considered accepted for the release – just proposed. The star (*) indicates an item up for architectural discussion.

Geneva Release Forecast Inclusion (p2)

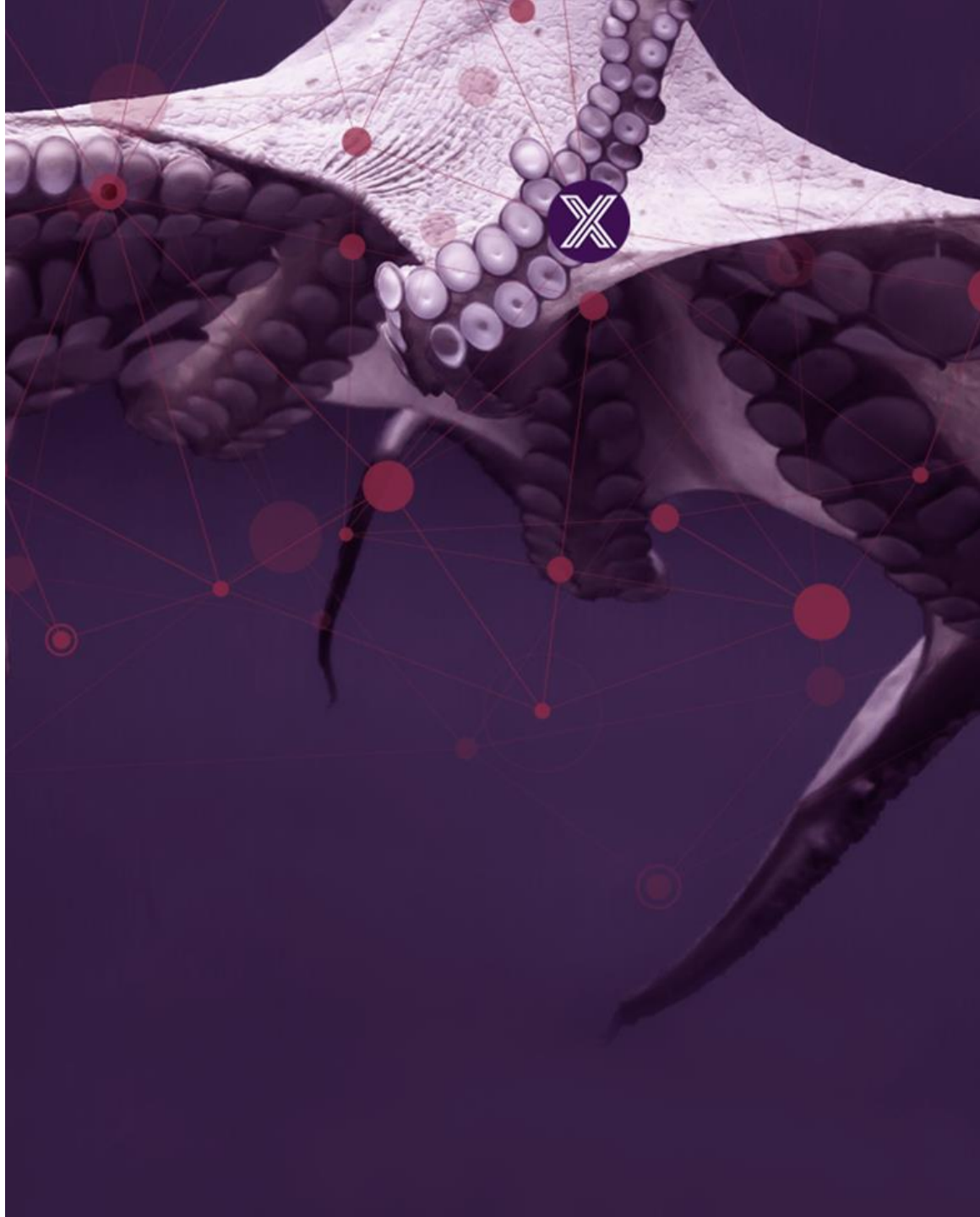
- Security
  - Hardware Secret Storage per Bryon N design (discussed at August F2F)
  - HTTPS endpoint security option for all service APIs
  - PKI Init merged into secret store (per August F2F meeting)
  - * - Role based security on SMA and other service APIs
- Device Services
  - Target new connectors (LORA? Zigbee? ZWave?)
  - Support the dynamic onboarding of new devices which conform to an existing device profile (Implement dynamic discovery using Provision Watchers)
    - Eliminating the need to itemize devices in the device service configuration.toml and could include the dynamic re-assignment of a device from one device service to another as in the case of a moving (wearable) device
  - Provide clean API for query parameters
  - Make more configuration dynamic (ideally, all of it)
    - Including the driver-specific configuration
  - Implement command chaining (deviceCommand contains other deviceCommands)
  - Allow driver to manage device operational state
  - Allow driver to feed back meaningful error information (not just ok/fail) from get/set handlers
  - Implement minimum/maximum restrictions on actuation parameters
  - More tooling around profile creation
  - Filter options for device service data to core data
  - Support downsampling (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)
- Certification
  - Define process for self-certification of device services. Currently estimated (For Fuji version) N.E.T. January 2020, but any Fuji slippage will push this back
- UI Projects
  - * - Consolidate to one UI

Geneva Release Forecast Inclusion (p3)

- System Management
  - Support async calls to SMA (everything today is blocking) - use of callbacks or pub/sub to support
  - Set read-only configuration data (and then optionally restart service)
  - Support set config of file-based config (non-Consul)?
  - Start/stop/restart of services inclusive of SMA (has asynch dependency)
  - CLI additions
    - work with security services (API gateway) in place
    - point to alternative database
- Test/QA
  - Interoperability testing (beyond unit and API tests) – testing to make sure data that enters from DS is available all the way up into application services & rules engine
  - 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 (These questions need to be answered on real hardware both Intel and ARM):
    - 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.
- DevOps
  - Snyk or other like code analysis tool implemented across services and integrated into pipeline
  - Full Pipeline transformation for EdgeX services (convert JJB Freestyle to Jenkins Pipelines)
  - Unit testing to global libraries
  - SonarQube or codecov.io
  - Code or artifact signing
  - Add Ocean Blue Jenkins Plugin to Jenkins for better visibility of the Jenkins Pipelines
  - Introduce GitHub Org Plugin
  - Re-Organization of Jenkins Views
  - Add Unit testing to global-libraries used within Jenkins
  - Nexus lifecycle policy enabled for cleanup of images
  - Slack integration with Jenkins Pipelines (failed messages only)

EDGE X FOUNDRY™

Architectural Issues



Architecture Topics for Discussion

- The next slides contain a list of architectural issues for discussion. They com from
  - WG and TSC leads
  - WG and TSC meetings
  - Prior community F2F meetings
- Which ones do we consider important to Geneva or future release?
- Which ones can be postponed indefinitely?
- Can we prioritize those that are to be discussed at the F2F
- Goal today is not to solve these, but to prioritize discussion
  - Appoint advocates for issue perspective for the F2F
  - Appoint research tasks for issues requiring more information/study prior to F2F

Architectural Topics (in advance of Geneva and 2020)

No priority has been given to these at this time

- The project is 2+ years old. Most release dates have been missed. How do we fix this issue? Do we fix it?
  - Train leaves the station on time approach
  - Each service is released independently approach
  - How do we deal with non-backward compatible change when needed?
- Developer best practices going forward (brought by Tony E.)
  - How to build better developer communications; reduce contention; improve planning
  - How to prevent large PRs
  - How many reviewers is enough for PRs, moves from holding, etc.
- If Geneva becomes V2, how do we 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.
  - 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.
- 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
- Move to Markdown (vs RST) for documentation
- How do we harmonize the experience between Go and C DS SDK? Between all SDKs?
- Dynamic service management - How to manage services when the list of services is ever changing (device and application services notably)?

Architectural Topics (in advance of Geneva and 2020)

- Experimentation in artifact creation/deployment packages
  - Producing multi-service deployment artifacts (e.g. a single core service) - Services Monolithic deployment
  - Producing service executables that combine services (e.g. a single core service executable vs micro services)
    - 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?
- How do we simplify EdgeX out of the box and make it easier to deploy/configure (aka – usability improvements)?
- Where should we consider adding message support between services (more Pub/Sub)? Do we ever get rid of 0MQ and if so, with what and when? Do we want to support other protocols between services (web sockets for example)?
- Do we want to expand our event/reading types to include maps and arrays? If so how?
  - Do we need to improve or add on to our binary data support?
- How (and should) data collected in non-device/sensor manor be easily entered into EdgeX data flow. Example – weather data read from the Internet; does it require a full-blown DS? Can a simple mechanism be used to enter data to Core Data and the rest of EdgeX sans profile, etc?
- How (and should) application services be able to re-enter events back into the system (core data). Example: an application service combines/reduces 3 readings from 3 events into 1 event/reading – how can this be re-circulated into core data as a new event?
- Should we explore more flexibility of the database? Do we invite other databases? Could we do without the database altogether?
- Rules engine – what to do?
- Is it time to consider more fine-grained access control
  - How to allow each service to have its own user/role based security (example: different user and access for SMA)
  - How to allow APIs to have individual access control (example: different permissions for get config versus set config)

Architectural Topics (in advance of Geneva and 2020)

- Are we ready to discuss HA needs yet? Improving EdgeX distributability and 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 cross EdgeX instance command actuation (ex: device X on EdgeX box A triggers action on device Y on EdgeX box B)
- Load Management in Device Services
  - Sources of demand
    - Device producing data autonomously
    - REST 'device' endpoint
    - AutoEvents
  - Shedding demand
    - For autonomous and REST-driven data, we can feed back errors to the caller, eg 503 unavailable.
    - For AutoEvents we could apply a scaling factor to the intervals?
  - Load monitoring
    - Monitor the number of simultaneous calls to get/put handlers?
    - Serialize Event submission and monitor the queue length?
    - Compare time taken to process an AutoEvent with its specified interval?

Architectural Topics (in advance of Geneva and 2020)

- The sys mgmt. executor presents all sorts of security risks. How should this be protected? How can we prevent someone with access to the EdgeX platform box from invoking the executor in an unauthorized fashion?
- Time of an SDK for the executors?
- CLI – expansion of capabilities? Bring into edgexfoundry organization?
- What did we learn from OH integration? What recommendations do we want to provide to OH team and the LF Edge?
- Is our strategy about facilitating many alternate deployment / orchestration mechanisms need to be revisited? Do we need to do more with K8s/K3s, Swarm, etc.? At what cost? What does the LF Edge community have to offer?
- Can we/should we provide a CloudEvents export? Do we use CloudEvents elsewhere? What about other event “standards”?
- Configuration is a bit complex with a lot of options. There are varying opinions about what should be done and what should be supported going forward
- C SDK artifact creation (library and/or tarball?).
- How to tag/version C SDK
- Dependency Injection – being explored in Core. Framework to consider? Use it throughout on only in core/support for now?
- Redis as default DB going forward
- LTS – when? Under what conditions?
- Security features done in December. Do we have another release in Feb or wait until Geneva?