Chicago Geneva Architect's Pre-wire meeting

10/9/2019

Hackathon review / synopsis

- Great event – especially given it was the first. Well done Intel, HP, Canonical and all that brought it together.
- Logistically, we need to find out why 3 teams dropped out before the event.
- Participant reactions to EdgeX
    - Device profile was complicated
    - Use of the rules engine needs to be made simpler – it could have been used in a couple instances
    - MQTT and REST were used a lot (well known and easier). Other connectors not so much.
    - See Brad's deck

Fuji Release

- Decision made to not produce another release for security features.
- Incorporate security features after release date into Geneva
- Feeling is that it is too much pressure to put on testing, CI/CD with little extra value.
- Point customers who feel they need them now to master

Geneva Scoping

Key – scope owners / focus
General change or addition
Application Services (or SDK) change or addition
Core (and Supporting) Services change or addition
Device Service (or SDK) change or addition
DevOps change or addition
Test/QA change or addition
Security Service change or addition
System Management Services change or addition

In Scope

The following list of tasks and features are currently considered in scope for Geneva (spring 2020) release pending approval at the Geneva F2F TSC meeting.

- Providing a 0MQ alternate implementation between core and application services. There is a messaging abstraction in place, but no alternate to 0MQ provided. With Geneva, find and offer an alternative. Requirement to date has been brokerless and works in all environments. This does not mean providing a message bus to the entire system yet. It also does not mean we are yet looking at providing more messaging as alternative to REST communications with services near term. These are longer term aspirations.
- Move to Go 1.13. No major feature obtained with this version, but keeps up with latest.
- Configuration simplifications where possible. Goal is to get more consistency across services and more straightforward overrides. Organization of configuration.toml should be cleaned up and simplified where possible as well.

- Application services should provide for batch and send modes. In other words, take a collection of event/readings over time, and send on certain intervals or triggers. Store and forward based on a schedule.
- Separate the configuration and registry APIs (currently in the go-mod-registry interface) allowing for alternate implementation of either in the future without being conjoined.
- Create and separate Request and Response types for API endpoints; these would be defined in go-mod-core-contracts request and response types and clean up metadata (removing any unused elements and simplifying where possible). This will allow the request and responses bodies of some APIs – especially around things like creating a device profile – to be greatly simplified and smaller. This will require Geneva be a 2.0 release because of non-backward compatible changes associated.
- Type the data in the event/reading. Eliminate use of value descriptors directly in the event/reading. Min, max, formatting and other attributes of the VD goes someplace else (to be determined where). This will reduce the number of lookups that need to occur across services. This will require Geneva be a 2.0 release because of non-backward compatible changes associated.
- Establish a document template for API information (description, request/response details and organization, response and error codes, etc.) This will be used to better define the Swagger documents that will be created as well as better support the certification process needs.
- Create a hardware secret storage design (following on a lot of the work already completed by Bryon N for the security F2F meeting this summer). [Note – see maybe in scope item with regard to possible reference implementation done this release]
- Create and use a per service Vault token in the security services.
- Perform dependency checks in the security service bootstrapping and initialization. When a dependent service is not up, keep checking until the dependent is up or a timeout is achieved.
- Service token revocation and rotation.
- KEY FEATURE – Device services (and the SDK) will include automatic/dynamic device provisioning capability – that is dynamic onboarding of new devices that conform to an existing device profile. As all devices would not support this feature, it must be optionally configured and used per device service. This feature must be supported via the SDK.
- KEY FEATURE – provide interoperability tests (? – to what degree and over which services initially?). This is testing to make sure data that enters from DS is available all the way up into application services & rules engine.
- Implement enough performance testing in order to be able to answer key performance measures such as
    - 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.
- All system management (SMA) APIs to be made asynchronous.
- Allow the SMA to stop/start/restart itself.

- Apply Synk scan to other services and images (it can't do ARM images)
- KEY FEATURE – replace CI/CD Jenkins jobs with Pipelines. This requires the use of Github.org Plugin for Jenkins.
- Use of Dependency Injection in Go services found in edgex-go. Goal of simplifying bootstrap of client and configurations and improve ability to unit test (with mocks).
- Move Redis to default database. Implement with username/password protection.

Maybe in scope – to be researched and additional discussion needed; may be set as stretched goals.

The following list of tasks and features may be considered in scope for Geneva (spring 2020) release. More research in advance of and discussion at the the Geneva F2F TSC meeting is required.

- Generic error handling (per Anthony Bonafide's presentation) – using Go's error/exception types. This would be for Go services/code only. This would include the DS SDK's driver to feedback more meaningful error information.
- Allowing data from Core Data to be pushed to multiple channels / topics and how to deal with marking an event/reading as pushed in that circumstances.
- Create a design and implement a means for application services to feed data back into core data (through a device service, via the core data API, etc.). For example, 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
- Support Cloud Event import (device service) and export (if not supporting Cloud Events model throughout)
- Provide a hardware secret storage implementation (following on a lot of the work already completed by Bryon N for the security F2F meeting this summer). In summary - read a file or script to get a password that encrypts the vault master key -- essentially a hook. Later a reference implementation possible.
- Create a plan/design for how to implement HTTPS in EdgeX (that is, how to protect all service endpoints with HTTPS).
- Create a plan/design for how to implement role-based security across our all EdgeX services.
- Provide a call in the Device Service to call out and execute a filter operation (similar to how the application services do it today). The hooks should be built into the SDK. Provide a design about how to implement this before implementing. If possible, can the filter functions be shared across App Services and D.S.?
- Providing a strategy and solution around handling a moving (wearable) device (-- metadata). I.e. how to handle a device that gets moved across device services in multiple EdgeX instances.
- Add unit tests/testing for global libraries.
- Sharpen our use of SonarQube (which can check for design flaws, code smells, etc.) and provide developer education around it. Possibly only apply to failure messages.

Out of Scope

The following list of tasks and features are currently considered out of scope for Geneva (spring 2020) release pending approval at the Geneva F2F TSC meeting. Some items have been ruled out of scope for future releases and will be removed from the backlog.

- Read only configuration watchers (like port) – configuration watchers for writable config is already in place. Any time a service needs to have its read configuration changed, this would require a service to be restarted anyway. Remove from future backlog consideration.
- Adding a reading cache to device service SDK (and device services). Keep in the backlog.
- Add additional language SDKs (application service, device service, etc.). Allow 3rd parties to make a contribution if necessary. Keep in the backlog but not a near term need.
- Adding new device service connectors (LORA, Zigbee, ZWave, etc.). Allow 3rd parties to make a contribution if necessary. Keep in the backlog but not a near term need.
- Add a cloud service app function to the App Functions SDK. Keep in the backlog.
- Rewrite of the device-service SDKs. Keep in the backlog.
- Set read-only configuration data and optionally restart the service via system management. Keep in the backlog.
- Set configuration via system management when the config is file-based. This creates some system-of-record issues. Can and should be handled more manually or by larger deployment/orchestration tool. Keep in the backlog.
- Provide command chaining. This is the ability for one command to cause the call of a set of commands (example: get causes a call to get temp, get humidity, etc.). This was a feature supported in Java command service and device services. Determination is to keep things simple and remove this from future backlog consideration.
- Provide query string support in the Event payload.
- Monolithic service build and deploy. That is, creating a single core service (from core data, metadata and command) to help reduce REST traffic in services and shrink the overall number and size of services for some deployments and use cases. While somewhat proven in research, this task will require many more steps of research to automate build and deployment. Keep in the backlog.
- HTTPS implementation – support to protect all service endpoints by requiring/using HTTPS over HTTP communications. Requires better / more secret store use (for certs, etc.). Keep in the backlog.
- Role-based security – allowing access to service APIs not just by a single authentication step, but also allowing services and service APIs to be secured by role based authorization/access controls (example, give SMA access to a more authorized user than get function calls on Core Data). Keep in the backlog.
- Provide down-sampling (and up-sampling) in device services. This is the capability whereby a device service decides to sample a device less frequently (or more frequently for up-sampling) based on the sensor readings it is getting or based on the context of what is happening in the system. For example, if a thermostat DS sees that the temperature is taking a reading every 5 seconds and the temperature has not varied in over an hour, it may decide to sample once every 30 seconds. This helps to reduce the amount of useless data collected and stored. Keep in the backlog.
- Code/artifact signing. Too many questions about what and how to sign and for what purpose at this time. Also, what would validate the signatures? 3rd parties can offer this at their discretion and some environments are already doing this (Docker for example). Wait for use case and demand. Keep in the backlog but not a near term need.
- Implement minimum/maximum restrictions on actuation parameters Keep in the backlog.

- More tooling around profile creation. Allow 3rd parties to make a contribution if necessary. Keep in the backlog but not a near term need.
- Make more configuration dynamic (ideally, all of it) including the driver-specific configuration. More information is needed from Iain on this work item.

Research Issues or Questions that need to be answered prior to the Face-to-Face

- What work is needed to implement TAF for all testing? Will moving to TAF clean up a lot of issues and bugs currently in the black box tests (James Gregg? to test/QA group).
- Are the UI teams ok with combining UIs? The desire is to have one not two. Who would build the combined UI? What technologies would be used? (Jim to coordinate with UI teams).
- Are there additional profile simplifications that can be done? Things like size as an example. Is everything in the profile being used? (Iain, Tony and DS WG)
- MUCH DISCUSSION NEEDED AT F2F: Certification – what do we really want to certify and what do we need in place to do this? When do we want to offer it? Certification WG is going to need a lot more help to accomplish this goal as it is perceived today. Who is going to really use certification?
- How can we provide better cloud support (AWS, Azure, etc.) without creating a long term maintenance issue? Keep the cloud application services in holding or move to regular EdgeX repo?
- Do the MQTT and REST device services need clean up? Some find them difficult to use. In particular, can they be made to easily ingest data without having to set up all the topics for all the get and set commands?
- Validate that the work of PKI Init has been merged and in the current secret store work.
- Slack integration with Jenkins Pipelines. We'd like Jenkins Pipelines to deposit issues/results to Slack but this would cause our 10K message limit to be exceeded more quickly. Can a new Slack group be set up for DevOps and this need?
- Make more configuration dynamic (ideally, all of it) including the driver-specific configuration. Iain – need more info on this scope request before making a determination.
- What additional needs do we have of the CLI? Can / should CLI work with security services in place? Could CLI be used to setup an alternative database.

At WG Lead discretion

These items are considered localized to the particular work group concerns and are designed/implemented at the discretion of the WG lead. They are more like issues than particular release scope tasks.

- Implement the non-backward compatible scheduler API changes (per Eric Cotter PR) – Core
- Allow driver to manage device operational state – Device Service
- Provide a clean API for query parameters – Device Service
- Allow driver to manage device operational state – Device Service
- Allow driver to feedback meaningful error information (not just ok/fail) from get/set handlers – Device Service
- Nexus cleanup and a lifecycle policy enabled - DevOps

Architectural Topics

Issues to be discussed at the face-to-face

Resolution to these items will likely add to the Geneva scope.

- How do we fix the issue of missing release deadlines or missing on functions in a release?
    - The overwhelming consensus is to take the "train leaves on time" approach (over "scope/functionality is king" and a release date is arbitrary and can be moved). This approach says that the release deadline is extremely fixed. Functions not ready at the release date are descoped for the release. **Mike J** to provide a proposal at the F2F for consideration on how we consecrate this policy. How do we improve our ability to better predict and deal with a missing feature? Do we have more deadline dates in the schedule? What role does the release czar play in this?
- Developer best practices going forward - 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.
    - **James G** to provide a proposal at the F2F for consideration by the community on how we improve our developer practices. James intends to set up a small sub-group to draft this proposal.
- How can we simplify configuration and bootstrapping?
    - IOTech has some recommendations. **Jim** with work with IOTech to present a proposal at the F2F for consideration.
- 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?
    - **Steve O** will present proposal at the F2F on how maps/arrays should be added to the event/reading structure (and how to handle it in the services).
- Rules engine replacement.
    - **Malini B** will provide research on replacement options and recommendations; exploring buy v build, Trevor providing input from the shell of the service he started to create. Others in the TSC offering help and suggestions. Requirements include: an executable that is relatively small (non-VM based). Easy to use and populate with "rules". Does not have to be fancy or complete but allow demos and other POCs to get off the ground quickly.
- What did we learn from OH integration? What recommendations do we want to provide to OH team and the LF Edge?
    - **Jim and Joe** to present proposal for OH sub-group at F2F. Broader LF Edge implications and impacts.

Issues tabled for a later date of discussion

- How do we harmonize the experience between Go and C DS SDK? Between all SDKs? Belief is that DS SDKs are not divergent, but the WG will assess. Larger issue of how to sync the App Functions SDK and DS SDK is larger and tabled for now.
- Experimentation in artifact creation/deployment packages. Producing multi-service deployment artifacts (e.g. a single core service) - Services Monolithic deployment. Not in scope for the next few releases.

- 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)? **Trevor** will look at simple 0MQ replacement for alt-go-mode for this release, but wholesale message capability is not in scope for the next few releases.
- Should we explore more flexibility of the database? Do we invite other databases? Could we do without the database altogether?
- Is it time to consider more fine-grained access control. Yes, but not in scope for this release without other security elements in place (like Vault and use of certs/tokens by services).
- 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?
- 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? Still a 3rd party issue at this time.

Issues resolved or to be resolved through normal working group channels

- If Geneva becomes V2, how do we support V1?
    - **Trevor C** proposed a way to handle this that was accepted by all. Trevor will write this endpoint convention up (that includes version in the API) and **work it through Core WG** and eventually up to TSC for final vote.
- 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.
    - **Lenny G** has an acceptable solution and will **work this through Core WG** and eventually up through TSC for a final vote as necessary.
- Move to Markdown (vs RST) for documentation.
    - Sent to **Robin** and **Test/QA WG for evaluation** in the amount of effort to accomplish this task and implications / impact. TSC is generally in support of this move, but before adopting Markdown and determining which release to do this, sent back for more input from the Test/QA group.
- Dynamic service management - How to manage services when the list of services is ever changing (device and application services notably)?
    - A solution is already believed to exist using the registry. **Michael E** will **work this through the System Management WG**. Suggest this be an issue/task for Geneva tracking.
- 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?
    - **Brad C** to provide proposal at F2F on non-device data and **work this in the new Architect's monthly meeting**. This is a longer term issue that will not be resolved for Geneva scoping.
- 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.

    - Huge topic worthy of some attention in **new Architect's monthly call**. **Jim** to orchestrate the discussion.
- Load Management in Device Services.
    - **Back to Device Service WG** for consideration and drafting of proposal. Probably not in scope for Geneva.
- May also be time for executor SDK.
    - **Michael E** to take up with **System Management WG**.
- C SDK artifact creation (library and/or tarball?).
    - Tony to take back and work a recommendation with **Device Service WG**.
- How do we tag/version the C SDK
    - **Jim** to orchestrate the discussion at **monthly Architects call**.
- Use of Dependency Injection in Go services.
    - To be implemented by **Trevor C** and team in edgex-go services. Added to Geneva scope. Revisit for wider adoption in future meetings.
- Redis as default DB going forward
    - Approved by TSC for Geneva with stipulation that default username/password be provided. **For Core WG implementation.**

Miscellaneous topics

- How do we improve usability of the platform? How do we improve the user experience? What tools, approaches, etc. can we take up to get useful information from users so we know what/how to improve it?
- Geneva will be a 2.0 release by nature of the anticipated non-backward compatible changes.
- If we can reduce the time it takes to go through scope, issues and backlog, how would we like to use the additional time at the F2F? Demo's, coding exercise, more training?
- MUCH DISCUSSION NEEDED AT F2F: LTS – Fuji nor Geneva are anticipated to be LTS releases at this time. Official criteria for LTS should be established by this F2F meeting. The current list includes
    - Pipelines must be in place and working well so that cutting a release with bug fixes is easier
    - We must feel comfortable enough with the platform to be able to announce at least one release cycle in advance that the next release will be LTS. I.e. – we already feel the current release is pretty stable.
    - We need at least some end-to-end testing (interop testing)
    - We need a "major" user or customer that requires LTS
- We need a tool to help collect and record architectural discussion and decisions.

Other Note Taker Raw Docs

- Original agenda, scope and architecture issues list: https://wiki.edgexfoundry.org/download/attachments/37912582/EdgeX-Prewire-Chicago-Oct-9.pptx?version=1&modificationDate=1570769427755&api=v2
- Malini - https://docs.google.com/document/d/1lEOvXydN75Iuaz39gmiT-0_8OtiH86Hua8oxpEV340A/edit

- Lisa - https://docs.google.com/spreadsheets/d/1DYICrYplbruUvLusXFN0Z8Tq4QI-Ddmu7IuPKWhW8C8/edit#gid=0