Fuji Release

Code Freeze: Oct 23, 2019

Release: Nov 15, 2019

The Fuji release is a minor release (EdgeX v1.1) that is intended to "hardened" the services, improve testing, and provide performance measures that give the user community even more trust and faith in the EdgeX platform from which IoT solutions are created.

Release Themes and Objectives

- Improve unit, black box and performance testing - increasing the test coverage and providing the means to identify breaks, performance problems or other issues in the code base sooner.
- Improved security features including use of Vault to store service secrets (ex: database username/password), improved PKI management, isolating micro service secrets in separate secret stores and ensuring the services running are those expected/trusted.
- Introduce EdgeX service self-assessment - the first step towards a program of EdgeX replacement micro service certification.
- Provide for the initial store and forward functionality; providing EdgeX the means to handle periods of being disconnected from north side systems by holding reading data until the system can re-establish connectivity and then forwarding the captured data.
- Improve upon the initial application services (and application functions SDK) - the new and more scale-able means of sending data to north side systems - with expectation of archiving/sun-setting EdgeX export services in the next release.
- Provide example application services to get data to Azure IoT Hub and Amazon IoT Core.
- Add additional device services to include BACNet, BLE and IP Camera connectors.
- Restart the EdgeX marketing work group.

Application Working Group Tasks and Notes

- With the Fuji release, the application services should be 100% functional replacements for export services (client and distro). This includes adding the following functionality to the application functions SDK for Fuji:
    - MQTTS/HTTPS support (and use of Vault for necessary keys, tokens, passwords, etc.)
    - Compression function
    - Encryption function
- An earlier decision not to support cloud IoT platforms directly was revisited and in the Fuji release, the new application services will now provide example services that get data to some of the large cloud providers such as Amazon IoT, Azure IoT Hub. Adding support for Google IoT Core or Chinese cloud providers like Tencent or Alibab is a stretch goal.
- Although sticking with the Drools/Rules Engine service for now (see research goals below), explore OpenJDK support and update the Rules Engine service JVM as necessary.
- Research rules engine replacement options.

Core Working Group Tasks and Notes

- Provide watchers and callbacks for all config and metadata changes (stretch goal)
- Update Consul to 1.4
- Address some critical refactoring to include
    - Making sure infrastructure touching code is decoupled and easier to test
    - Address current typing issues that are not helpful to adding more unit testing (more encapsulated types for tests)
    - Setting up better transactional boundaries
    - Addressing issues around Value descriptor & Device Profiles out of sync when changes occur

Device Service SDK and Device Services Group Tasks and Notes

- Priority of work for this release in the DS/DS SDK area is in adding blackbox tests (defined by a test plan) to the SDK (and thus device services).
- Create a generic IP camera device service (using ONVIF protocol where possible)
- Time permitting, additional goals (as stretch) for this release include:
    - Allow for the deregistration of devices/device services
    - Provide dynamic/automatic discovery scaffolding in the SDKs to allow device services to automatically discover and provision new devices at the DS creators discretion. This may include the need to establish black/white lists of devices to explicitly include or exclude from that discovery.
    - The SDKs will implement a means to provide a cache of readings. This allows the collection and response for a request of a reading to be decoupled (and more asynchronous). https://github.com/edgexfoundry/edgex-go/issues/829

Security Group Tasks and Notes

- In the Fuji release, PKI infrastructure will be added to generate the tokens and keys necessary to use Vault, Kong and other security services. Today, EdgeX relies on the keys to be generated elsewhere and then used by security apparatus.
- EdgeX micro service secrets will be stored and distributed per service in Vault (using namespaces). Today, all services access the secret store with the same key and therefore have access to all secrets.
- Include technology to ensure services running in EdgeX are thos expected (and authorized).
- By the Fuji release, the EdgeX community will define/design a hardware secure storage abstraction layer that will include a software implementation and allow platform providers to build hardware root of trust implementations that can be used by EdgeX to protect the Vault Master Key and other fundamental system secrets used at bootstrap time.
- Add documentation defining, a higher level, what security features EdgeX offers and what is the security feature roadmap of EdgeX. Additionally, the community will renew/refresh a threat assessment.

System Management Group Tasks and Notes

- Refactor the SMA executor to accomplish Start/stop/restart tasks by the executor (to include stop/restart of SMA). This requires the executor track completion of the operations and returns results.
- Refactor metrics collection - moving metrics collection to the executor so that it can remain platform and even service implementation agnostic.
- Add the ability of the SMA to set configuration (when the configuration is writable). Today the SMA can only get the configuration information from each service.
- As a stretch goal, add an SMA Translation layer. The SMA will provide a translation layer (implemented via necessary abstraction) to offer the SMA API (and associated data) via other protocols starting with one protocol (like LWM2M or SNMP). In effect, SMA will provide access to SMA API and control plane data in a fashion similar to how Application Services makes data plane available to 3rd parties in a fashion dictated by those 3rd party clients. https://github.com/edgexfoundry/edgex-go/issues/835

Certification Group Tasks and Notes

- Provide self-assessment of device services by EOD 2019 (with a stretch goal of self-assessment of all services).
- This effort requires device service black box tests in the Fuji release.
- Create a web page highlighting 3rd party services that have been self certified
- A certification program - once in place - will allow third parties creating EdgeX services to verify their services as alternative or enhancing capability to those provided by the EdgeX open source effort. This will allow 3rd parties to add value (proprietary or open source) to EdgeX that customers can rely on to meet the EdgeX APIs and work without additional code change (enable a plug-and-play ecosystem). Various levels of certification are being considered, from micro service replacement certification (validating alternate or commercial implementations of EdgeX micro services satisfy API requirements along with performance metrics and quality checks) to full EdgeX deployments (for commercial versions of EdgeX). Additional certification processes may be developed around particular cross cutting features such as security.

Test/QA/Documentation Group Tasks and Notes

- Improve and increase performance metric capture from all services. The goal is to, by the Geneva release, be able to answer 3 primary performance questions:
    - 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.
- Improve blackbox test structure including reorganization of the tests and better test case documentation.
- Create a new test framework (e.g. Robot or Cucumber) to support additional types of functional/blackbox and system integration tests (e.g. Device Service or system level latency tests).
- Remove documents from the other code repositories to its own repository
- Add performance testing automation. Specifically:
    - Automate API Load testing (measure response time) and metrics (CPU, memory) collection for all EdgeX micro services
    - Edgex micro service startup times
- Improve the documentation to address some critical and re-occuring needs:
    - How to get started with Windows and EdgeX (VSCode install and use, 0MQ install, etc.)
    - Add a common troubleshooting guide – how to pull the logs, how to decipher what’s in the log, how to check issues in docker, etc.
- Add a documentation versioning tool (like Sphynix)

DevOps Group Tasks and Notes

- Add static artifact analysis into the EdgeX Jenkins Pipeline (analysis of Docker /runtime artifacts, not the source code)
- Add code and artifact signing with semantic versioning
- Conduct build performance optimizations by:
    - Adding Pipelines for EdgeX Foundry base build images
    - Allow basebuild images to be managed locally within Nexus
    - Leverage PyPi Proxy for local pip dependencies
- Explore static code analysis like Coverity.

General Tasks and Notes

- Move to Go 1.12 (evaluate and possibly move to Go 1.13 on its arrival)
- EdgeX will move from RAML to Swagger for API documentation
- Use nanoseconds for all Event/Reading timestamps (a change from milliseconds).
- Update the EdgeX "Offerings" page on the EdgeX Website to highlight 3rd parties offering EdgeX products and services.
- Research options for better building/packaging/using alternate infrastructure elements that would have been accomplished by Go Plugins if not for the fact that this Go Lang feature is not supported and apparently dying.
- Reconstitute the EdgeX marketing working group - lost with the LF Edge umbrella project creation. This working group will serve to meet EdgeX marketing needs (event planning, promotional material, etc.) as well as provide the LF Edge Marketing Group with EdgeX feedback and insure EdgeX marketing needs are satisfied.
- Capture unit, integration, and other testing coverage metrics so that the test coverage can be prepared at each face to face meeting. This will help address the need to implement a test coverage metric in the future.
- Elect a new release manager for Fuji and each subsequent release. In order to address pace of change issues, the release manager will attempt to implement release milestones like "no new functionality" dates.