Delhi Release

NOTE: Delhi has been released. See the release page for details of this release.

- NOTE: Delhi has been released. See the release page for details of this release.

Delivery: ~ November 2018

The Delhi release is slated to provide the first system management capabilities while also implementing additional features in the security. Delhi will continue to show improvements in performance by offering Device Service SDKs in Go and C. Improved testing at all levels (unit, blackbox, performance, etc.) to ensure the quality of the system going forward while also providing a means to check backward compatibility. Finally, this release will include design and implementation plans (scheduled for Edinburgh release and beyond) for the replacement of MongoDB as the reference implementation database, and a replacement to the export services to allow for better scale and flexibility at the northbound layer. The scope of this release is smaller given the development cycle for the prior California release was made longer to accommodate all the Go Lang refactoring.

Release Themes and Objectives

- Deliver system management APIs in all the microservices and provide a system management agent for orchestrating multi-service system management commands and communications to 3rd party system management tools/platforms.
- Go and C device service SDKs with at least one device service implementation for example sake.
- The next wave of security features to include access control lists to grant access to appropriate services, and improved security service bootstrapping.
- Refactored and improved Go Lang microservices
- Options and implementation plan for database replacement
- Design and implementation plans for export service replacement with application services
- Provide an EdgeX UI suitable for use in exploring several instances of EdgeX

General Tasks and Notes

- Provide an initial EdgeX user interface. This Javascript-based user interface will allow users to explore the sensor data collected by EdgeX as well as provide some provisioning and management capability. Used for prototyping, demonstration and management of small clusters of EdgeX instances.
- Update all services to return proper HTTP status codes on HTTP requests

Core & Supporting Services Tasks and Notes

- Replace Java scheduling microserice.
- Improve the configuration of EdgeX
    - Refactor config-seed to only load the configuration appropriate to the application setting based on profiles (i.e. development, production, production docker or snap, etc.)
    - Replace the simple key/value pair organization of service configuration with more structured and hierarchical configuration as supported by Consul
    - Upgrade Consul and improve the configuration seed
    - Implement a process to move the latest configuration into the config-seed through the CI process
- Core, supporting and export services are refactored to offer better code organization, abstraction from certain resources (offering better flexibility) and improve unit testing.
- Implement security & system management API hooks. Depending on the implementation needs of security and system management, there may be a need to make changes to the micro service code in support of these facets. It is likely that each micro service will implement some sort of base service to provide data and system management functionality to the system management agent. Access control to the service APIs will also be managed by the security services and code added to the service to require authentication before accessing the services will be needed.
- A design and implementation plan will be provided with Delhi (for Edinburgh implementation) to add support for binary data (namely via CBOR) through device services, core data, and export services.
- While MongoDB has been the foundational EdgeX data persistence store since its beginning, concerns around its license, ARM 32 support, and size/performance have the community looking into possible alternatives. At the very least, the platform should more easily support the swap of the database by 3rd parties. The Delhi release will provide designs and plans for implementation in the Edinburgh.

Export Services Tasks and Notes

- Implement security & system management API hook. Depending on the implementation needs of security and system management, there may be a need to make changes to the micro service code in support of these facets. It is likely that each micro service will implement some sort of base service to provide data and system management functionality to the system management agent. Access control to the service APIs will also be managed by the security services and code added to the service to require authentication before accessing the services will be needed.
- Design the application services - a replacement of export services. The existing export services will not scale because additional code must be added to the client and distribution services with each additional transformation, filter, formatting, or endpoint need. Further, it does not allow for eliminating filter, transformation, or endpoint distribution code when it is not needed. Adding additional capability requires additional code be added to the existing service. The new application services (as initially specified in the document here) will be backed by requirement specifications and designed in time for the Delhi release with the goal of implementing this replacement service(s) for the Edinburgh release.

Device Services and SDK Tasks and Notes

- Implement device service SDKs in Go Lang and C. The device service SDKs and resulting device services will allow the EdgeX community to retire the last of the Java micro services and complete the dramatic performance improvement of EdgeX.
- Implement security & system management API hooks. Depending on the implementation needs of security and system management, there may be a need to make changes to the SDKs and existing micro service code in support of these facets. It is likely that each micro service will implement some sort of base service to provide data and system management functionality to the system management agent. As device services are typically created from an SDK, the same boilerplate code for the base service needs to be added to the SDK(s). Access control to the service APIs will also be managed by the security services and code added to the service to require authentication before accessing the services will be needed.

Security Tasks and Notes

- In California, some of the initial security services were put in place - namely the reverse proxy to protect the micro service APIs and a secure store for storing EdgeX secrets (like database passwords, certificates, etc.) Additional security capability will be added on top of these initial building blocks in Delhi. New security features to be added include:
    - Use an ACL plugin to provide access control to micro services
    - Adding various EdgeX secrets to the secure storage
    - Improve the secure bootstrapping

System Management Tasks and Notes

- System management API (action, alerts, metric) as discussed and outlined here
- System management Agent (see same document above outlining the agent functionality and duties)

Testing/QA

- Performance tests on startup time, request/response times on all APIs, latency to actuation from device service collection, through core data, to rules engine, command back to a device service.
    - Performance metric testing will include CPU and memory usage statistics
- Implement a performance test harness to establish a baseline of performance characteristics that can be used to understand EdgeX resource utilization, data throughput, etc.