Barcelona Release

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

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

Delivery: late October 2017

This first EdgeX release is focused on delivering a minimum viable product (MVP) for further accelerating community contributions. It is the hope of the EdgeX community that with the Barcelona release, the shape of the architecture, APIs, etc. are such that the larger IoT development community will feel comfortable exploring and contributing to EdgeX and begin planning integration of the EdgeX foundation with their own IoT projects and offerings.

Release Themes & Objectives

- Stabilize the original seed code for the EdgeX platform
- Establish dev ops practices, build environment and versioning
- Build community understanding of the platform and code base
- General majority agreement on the architecture
- General majority agreement on micro services and APIs
- General agreement on longer-term performance targets and technology choices
- Progress towards the definition of unified APIs for security and system management

Key Features

- Stabilization of key APIs
- Better code quality, fit and finish
- More than double the test coverage across EdgeX microservices
- Addition of reference Device Services supporting BACNet, Modbus, Bluetooth Low Energy (BLE), MQTT, SNMP and Fischertechnik
- Extra “north side” Export Service interfaces that provide connectors to Azure IoT Suite and Google IoT Core as well as support for connections via MQTTS and HTTPS

General Release Tasks and Notes

- Supported OS
    - Windows 10 (latest 2016 version)
    - Linux, Ubuntu 16.04 classic (but does not enforce secure boot)
    - EdgeX supports and welcomes other OS providers to test and validate EdgeX works on their OS
    - Example, Canonical plans to test/validate EdgeX on Ubuntu Core 16
- Supported Hardware
    - Intel x86
    - Stretch goal: Arm Note: work to port to Arm is nearing completion although the automated build environment needs further attention.
- Performance
    - No significant performance-specific work will be accomplished in this release (unless it is easy to achieve with no other impacts)
    - Provide early performance estimations for alternative Go Lang implementation of microservices
    - EdgeX will use current scalability metrics on devices, collection, etc. as scale guidance
    - More formal scalability concerns to be addressed in future releases
- All microservices hardened; meaning
    - Works properly for the intended use case
    - It may not be 100% complete implementations for all use cases or parts of a protocol for example, but it provides enough implementation to sustain the demo use cases for Barcelona and could support extension to the full needs or protocol in the future
    - Handles errors and exceptions gracefully
    - Contains proper unit and integration tests
    - Follows good coding standards, and is well documented
    - Following some prescribed standard (like Oracle, Google or Twitter guides for Java)
    - Performs within the target measures established for Barcelona
    - Code base is not exploitable
    - API set is solid

Application Layer (export services and the rules engine microservice) Tasks and Notes

- Harden as defined above
- Provide Azure IoT Hub integration (improve/clean up functionality in the original seed code) and support for Google IoT Core
- Provide routing to endpoints by device ID
- Support MQTTS and HTTPS for endpoint distribution
- No additional formats/transformations/filters/etc. will be provided beyond what already exist in the export services today
- Provide guidance on number of simultaneous clients that can be supported (to address any potential scale problems with the export services)
- Stretch Goal: binary message format distribution (picking one to start)

Core & Supporting Services Tasks and Notes

- Harden as defined above
- Fix known bugs (logging OOM, …)
- Remove/clean up unfinished features (Device Manager)
- Complete at least one full replacement Core Service in Go Lang to feed estimates on performance trajectory

Device Service SDK and Device Services Tasks and Notes

- Harden as defined above
- Provide set of reference Device Services (Modbus, BACnet, BLE, MQTT, SNMP, Fishertechnik, and virtual device)
- Clean up SDK (and Device Services)
    - Improve documentation
    - Be more developer friendly (see improve tooling - stretch)
    - Merge device-sdk into SDK tools
    - Cleanup scheduler
- Apply cleanup back to DS (with exception of BACnet and BLE)
- Stretch Goals
    - Improve tooling (Eclipse Plugin)

Security Tasks and Notes

- No implementation provided with this release
- Build the longer term roadmap – the EdgeX security story
- Agree on overall security requirements for EdgeX
- Agree on what security features are going to be in EdgeX and what's going to be provided by the platform that EdgeX runs on (example: the underlying platform must have a keystore)
- Further definition on what EdgeX security service(s) need to be eventually implemented (e.g. modules for data protection, identity and access and operational security)
- Further definition on what security hooks need to be added to the existing microservices (e.g. APIs)
- Further definition on standards, protocols, etc. that are going to be adhered to and followed by EdgeX (ex: IIC specs, OAuth tokens, etc.)
- Guidance on how security features can/should be tested

System Management Tasks and Notes

- No implementation provided with this release
- Agree on overall requirements
- Agree on what features are going to be in EdgeX and what is reserved for OS, 3rd party systems, other open source systems, etc.
- Define what system management services need to be implemented as part of EdgeX (if any)
- Define what system management hooks need to be implemented
- Define any system management standards that will be followed/used in system management implementations (ex: LWM2M)
- Stretch goal: add some simple system manage hooks/capability into BaseService of EdgeX micro services – based on Dell Fuse work (for service start, stop …)

Test/QA Tasks and Notes

- Unit and integration tests for all microservices (all public methods)
- Implement Blackbox testing
    - Select and/or implement black box test framework
    - Testing must occur on all supported MVP platforms
    - Setup performance tests for capturing performance metrics of a micro service or combination of services
- Have check styles in build process
- Setup Bug tracking system

DevOps/Build/CI/Process Tasks and Notes

- WG allowed to setup their processes (Agile, Scrum, etc.) as they see fit (no community wide process for now)
- Automate build of code and docker microservices
- Implement standards on code commenting, branching, versions, …
    - Adopt and apply the Google code standards (aka Style Guides) for all code
- Stretch Goal: Automate build of API documentation (Javadoc, raml-doc ?)