Geneva Release

Delivery: ~ April 2020

The Geneva release will be a minor release (1.2) with a focus on testing, a few but important key features (such as automated device provisioning and a replacement for the final Java service - the rules engine) and the first part of the V2 API (which will be provided in "beta" form for core services with the intention of completing the V2 API in all services by the Hanoi release).

- Geneva Release Overview
- Release Themes and Objectives
- General
- Core/Supporting Services
- Application Services and App Functions SDK
- Device Services and Device Service SDKs
- System Management
- Security
- Test/QA
- DevOps
- Certification
- Vertical Solutions WG
- Additional Release Notes

Geneva Release Overview

- Improved Security
- Interoperability testing
- User guidance on platform needs
  - More performance statistics
• # of devices/per recommendations
• 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:
  • 1) Will EdgeX fit on my system? - size of EdgeX services, infrastructure, etc. and hardware/platform requirements
  • 2) 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?
  • 3) How many “things” can be processed at a time? – with caveats on the type of thing, type of data, etc.
• These questions need to be answered on real hardware (both Intel and ARM)
• Dynamic device provisioning/on-boarding
• Alternate messaging support (to 0MQ)
• Archive of Export Services - in favor of Application Services which was implemented with the Edinburgh and Fuji releases
• DevOps Jenkins Pipelines
• The first steps toward a revamped V2 API that includes cleaner (and fewer) endpoints and a simplified request/response model
• Provide an alternate local analytics service (archiving the Java-based rules engine service)

General

• Move to Go 1.13
• Redis as default DB
  • Implement with username/password protection.
• 0MQ Alternate between core and app services
  • Should help with Windows divy (long standing backlog)
• Separate the configuration and registry APIs
• Establish a document template for API information
  • Used to better define the Swagger documents
• Use of Dependency Injection in Go services found in edgex-go

Core/Supporting Services

• Combine/reduce UIs
• Blacklist/whitelist of devices (w/ DS WG)
  • As part of auto provisioning
• Alternate message bus provider (w/ App WG)
  • 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
  • Implementation of the V2 APIs across core services (beta/use at your own risk)

Application Services and App Functions SDK

• Export Service archive/deprecation
• Application services should provide for batch and send modes
• Rules Engine Replacement (w/ Core WG)
• JSON Logic and/or EMQX Kuiper implementation
• Create a design and implement a means for application services to feed data back into core data
• Support Cloud Event import (device service) and export (if not supporting Cloud Events model throughout) – stretch goal

Device Services and Device Service SDKs

• Automatic/dynamic device provisioning capability
• Array of data types (w/ Core WG)
• Data filter design between DS and Core Data
  • Provide a design about how to implement this before implementing.
  • If possible, can the filter functions be shared across App Services and D.S. (w/ App WG)

System Management

• Start/stop/restart all done by the executor to include stop/restart of SMA
• CLI improvements (TBD)
  • Move CLI out of holding
• Open Horizon “Walk phase” (TBD)

Security

• Create a hardware secret storage design
  • HW secure storage abstraction layer
  • How to protect the Vault Master Key
• Create and use a per service Vault token in the security services
• Service token revocation and rotation
• Ensuring the services running are those expected and authorized (w/ DevOps assistance)
• Blackbox tests of APIs through the API gateway
• Design work
- How to implement HTTPS in EdgeX (that is, how to protect all service endpoints with HTTPS)
- How to implement role-based security across our all EdgeX services.

**Test/QA**

- Device Service testing – complete testing for current set of EdgeX Device Service (w/ DS WG)
- New blackbox tests to support V2 API changes on new Robot-based Test Automation Framework
- Documentation – move all API definitions to Swagger (w/ all WG assistance)
- Documentation – move from RST to Markdown
  - Explore documentation versioning – stretch goal
- System integration / interoperability tests - Device Service read data -> Core Data -> Rules Engine or Application/Export Service -> Command
- Implement enough performance testing in order to be able to answer key performance measures – extend existing Robot perf test summary suite developed during Fuji
- Add unit tests/testing for global libraries. (w/ DevOps help) – stretch goal

**DevOps**

- Move to Jenkins Pipeline
  - Requires the use of Github.org Plugin for Jenkins
- Apply Synk scan to other services and images (w/ all WG input)
  - Synk can’t do ARM images
- Sharpen our use of SonarCloud and provide developer education around it – stretch goal

**Certification**

- How to certify and track minor API versioning works in practice.
- Planning and design work toward certification and self-assessment for when LTS hits.

**Vertical Solutions WG**

- China Project Team in place
- Industrial Project Team – stretch goal

**Additional Release Notes**

- Geneva will not be LTS. While no specific future release is pinpointed for LTS, the general hope is that the Ireland release will be a bug fix only, minor release that might be our best opportunity for an LTS.
- V2 API implementation will be accomplished, tentatively, over 2 releases (Geneva and Hanoi). For Geneva, core services will implement the V2 API as “beta” and use at your own risk. The work is going to be generally completed by teams from Dell, Intel and IOTech but will likely involve impact all developers to some degree.
- Important new features of Geneva (ex: automatic provisioning) will still use V1 APIs
- SDKs and other services would still generally call on V1 APIs (still rely on V1 clients)
- The V1 APIs will be retired when the V2 APIs are released (again, likely with the Hanoi release)
- Export services are removed from the Geneva release. This required the approval of a backward compatibility exception for Geneva since it is no longer a major release.
- TAF will be used as overarching API/blackbox testing platform. Tests for V2 APIs (only) will be created by TEST/QA team
- Postman tests remain for V1 APIs
- Dev teams are welcome to (encouraged?) to provide language dependent service level tests in their code base as they see fit (example Go Tests) but this is optional at this point