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.

- 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

- JSON Logic and/or EMQX Kupfer Demos (App WG)
- Archive Support Rules Engine (App WG)
- API Minor Versioning (Certification WG)
- Data Feed Back into Core Data (App WG)
- App Services: Batch and Sand (App WG)
- Archive Export Services (App WG)
- V2 API Specification (Core WG)
- V2 API Implementation (Core WG)
- V2 API Test Plan (Core + Test/QA WG)
- System Integration Tests (Test/QA WG)
- Blackbox TAF Framework (Test/QA WG)
- Alternative Message Bus Provider (Core + App WG)
- Data Filter Design DS and Core Data (Device WG)
- Dynamic Device Provisioning (Device WG)
- Blacklist/Whitelist of devices (Core + Device WG)
- Archive Export Services (App WG)
- Update to Golang 1.13
- Separate Config and Registry APIs
- Redis Default DB
- Jenkins Pipelines Transformation (DevOps WG)
- Open Horizon "Walk" Phase (System WG)
- Open Horizon Build Automation (DevOps WG)
- Hardware Secret Storage (Security WG)
- Per Service Vault Token (Security WG)
- Vault Token Rotation (Security WG)

Release Themes and Objectives

- Improved Security
• Interoperability testing
• Dynamic device provisioning/on-boarding
• Alternate messaging support (to OMQ)
• Archive of Export Services - in favor of Application Services which was implemented with the Edinburgh and Fuji releases
• DevOps Jenkins Pipelines
• Provide an alternate local analytics service (archiving the Java-based rules engine service)
• Deprecating the following:
  • Logging service
  • MongoDB support

General

• Move to Go 1.13
• Redis as default DB
  • Implement with username/password protection.
• OMQ Alternate between core and app services
  • Should help with Windows dev (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

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

• 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
• 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 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

Certification

- Planning and design work toward certification and self-assessment for when LTS hits.

Vertical Solutions WG

- China Project Team in place

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.
- 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.