Hanoi

Hanoi Release Notes-v4.pdf

Release Date: November 8th, 2020

Namer: Joan Duran

The Hanoi Release is the 7th successful community release of EdgeX Foundry. It is a minor (dot) release (version 1.3) and backward compatible with Edinburgh (1.0), Fuji (1.1) and Geneva (1.2) along with any patch releases of the 1.x releases.

Release Major Themes

- Restructured Docker Compose Files
- Distributed Device Services
- Initial Performance / Scalability Tests
- Docker Compose file “make”
- Edge Data Tagging
- Command Line Interface Tool
- Security Guidelines
- UI Refactor/Improvement
- Fledge Integration
- Beta/Experimental V2 APIs
- Device Service Contributions

V2 APIs

Although a dot release, Hanoi includes a significant number of new features. It also incorporating the first collection of new, platform-wide, micro service APIs called the V2 (for version 2) APIs. The existing EdgeX APIs have been in place since its first release. The V2 APIs will remove a lot of early EdgeX technical debt and provide a better informational exchange. It will take the community a couple of releases to complete the V2 API. We plan to complete the V2 APIs with the next release called Ireland. The new APIs will also allow for many new, future release features. For one, the request and response object models in the new APIs are richer and better organized. The models will better support communications via alternate protocols in the future.

The first set of the V2 APIs released with Hanoi are beta APIs. The V2 APIs can still change in the future. These APIs provide developers a feel for what will become the principal way to interact with EdgeX micro services in the future. The V2 APIs are not backward compatible with the V1 APIs. Thus, view the V2 APIs as means for early exploration. The EdgeX community hopes to elicit commentary and suggestions on additional improvements.

This release includes many core data and core metadata service V2 APIs. The device and application services have also been outfitted with new V2 APIs. These first APIs allow for device and device service provisioning and setup. Provisioning is a critical element of EdgeX "thing" management and serves as a good place to gauge the new V2 APIs effectiveness.

Adopter Warnings

Per the Geneva release in the spring 2020, the EdgeX community deprecated three EdgeX features. This includes MongoDB support, the Support Logging Service, and the Support Rules Engine (which wrapped Java-based Drools rules engine). These services remain but are still deprecated in Hanoi in order to maintain backward compatibility. EdgeX adopters should make plans to move away from these services. The community will remove these in an upcoming major release (currently forecast for the Ireland release in the spring of 2021).

Know Bugs

Please see the Issues List in EdgeX Foundry Github repositories for known bugs. In any repository Issue List, look for issues with the “Bug” label – such as https://github.com/edgexfoundry/edgex-go/labels/bug for bugs in the main repository of edgex-go.

- In the case where a large number devices are connected through a device service, and the device service is restarted, the maximum HTTP request size (for Go Lang applications this is 10MB) may be exceeded when the device service restarts and makes call to Core Metadata for all relevant devices and device profile information. This issue is being addressed in the V2 API to be released with Ireland release (spring 2021). Although it requires some custom coding, there are workarounds to this issue. Connect with the EdgeX community via Slack (https://edgexfoundry.slack.com/archives/CE44YG81E) if you need assistance if this issue is applicable to your use case.
- When a device is locked / unlocked via core metadata (via set of AdminState update), the callback to the device service is not notified of the change. See https://github.com/edgexfoundry/edgex-go/issues/2880.

General

- Migration of the platform to Go 1.15
- Upgraded to Redis 6
- A program to vet and approve 3rd party libraries and modules
- Restructure of the Docker Compose Files and make facility to create them (removing a lot of duplicate code and the need for users to uncomment for specific service needs). The Docker Compose files now take advantage of the multi-file Compose approach.
- Corrected service Dockerfiles to use CMD versus ENTRYPOINT
- Snap artifacts now being managed, built and released to the Snap store by Canonical. The EdgeX Snap names have also been transferred to Canonical.
- EdgeX web site refresh
- Introduction of the User Self-Endorsement program (see EdgeX Wiki).
- Design of Device Service to Application Service communication via message bus (bypassing Core Data).
- Adoptions of Conventional Commits specification across EdgeX repositories
- Use of Dependabot to watch for and create pull requests on dependency upgrades
- EdgeX Documentation refactor; improvements, cleanup and refactor
- Upgraded to new releases of Vault (1.5.3), Kong (2.0.5), Consul (1.8), Kuiper (1.0).
- Removed all RAML API specification documents and moved all API documentation to Swagger (also provided in SwaggerHub).
- Created a new edgex-examples repository and moved all project examples to this repository. Developed policies and procedures for managing and maintaining these examples.

Core and Supporting Services (along with common modules)

- Experimental V2 APIs in Core Data and Core Metadata; V2 APIs also provided in the App Functions SDK and Device Service SDKs.
- Added support for long/HTML email messages in support-notifications.

Application and Analytic Services (and Application Functions SDK)

- Event data tagging; ability to configure each EdgeX instance to tag the data as it is exported from the EdgeX instance. This allows the EdgeX data to be pinned in some meaningful way so that using systems and applications know where the data originated.
- Improvements to the Kuiper Rules engine; upgrade to version 1.0 of Kuiper
- Added ability to use PUT HTTP method for HTTP exports.
- Added new MQTT Trigger function to the App Functions SDK

Device Services (and SDK)

- Allow a device service to be distributed (running on a different host than the rest of EdgeX).

System Management

- Initial steps to better facilitate Kubernetes deployment by providing a limited, single pod deployment file for EdgeX.

CLI and UI

- Formal release of version 1.0 of the EdgeX Command Line Interface (CLI) tool
- Refactor and improvements of the EdgeX UI
- Roadmap development for both the EdgeX CLI and UI

Security

- SSH tunneling how-to-guide; how to provide SSH tunneling between EdgeX services.
- Overlay network how-to-guide; a demonstration on how to setup EdgeX in Docker Swarm and how to use an overlay network to provide encryption between service containers.
- Design of a secrets abstraction that provides all EdgeX services with a consistent means to retrieve secrets for a secret store.
- Design a means to generate and inject random passwords into Vault (EdgeX's secure storage).
- Added a new security service to bootstrap Redis.
- Designed the means to better secure the EdgeX Docker containers (and avoid escalation of privileges).
- Designed a new EdgeX security utility that performs post-installation EdgeX secrets configuration.
- Added hooks to the security secret store setup for hardware assisted protection of the secret store (Vault) master key.
- Added a persistent data volume for Postgres (Kong's DB).
- Modified the Docker Compose files to run service containers with read-only file systems.
- Provided documentation on how to add additional services to the API gateway

DevOps

- Snap build improvements to include build speed improvements for amd64 and arm64
- New faster build pipeline for edgex-go (build images in parallel)
- New edgex-cli build pipeline, create and publish edgex-cli binaries
- Snyk scanning pre-release docker images
- Auto generation of documentation for edgex-global-pipelines
- Linting of groovy code in pipeline to avoid syntax failures
- Cleanup and standardization of Mocking in unit tests
- Implement mocking of transitive functions in edgex-globalpipelines
- Introduce concept of ”build commits” to trigger rebuild of artifacts
- EdgeX release optimization (rebuild artifacts)
- EdgeX AWS ECS reference stack deployment
- Implemented GitHub release automation
- GitHub prune of stale/pre-release git tags
- Implemented a GitHub label/milestone sync (with throttling)
- Generated reporting for stale docker images

Test/QA (and documentation)

- Addition of TAF blackbox tests to include TAF tests of app services and the V2 APIs. TAF testing will eventually replace Postman blackbox testing in a future release.
- Improvements to the performance tests to include running weekly performance metrics checks and adding range checks on response times. A design has been created for conducting long running performance tests which will be implemented in the next release.
- Initial scalability testing (using Modbus) to understand how many events can be handled by an EdgeX instance over a specific period of time
- Platform smoke test to check for issues when any infrastructure is updated (Kong, Consul, etc.).

List of supported connectors

Those in bold are new with this release

South Side (Device Services)

- Modbus (TCP/RTU) in Go
- Virtual Device in Go
- SNMP in Go
- MQTT in Go
- BACnet (IP & MSTP) in C
- ONVIF Cameras in Go
- REST in Go (NEW!)
- Grove in C
- LLRP RFID in Go (2021) – contributed and under review
- Bluetooth Low Energy in C (2021) – contributed and under review
- OPC-UA in C (2021) - under dev
- CoAP (2021) – contributed and under review
- GPIO (2021) – contributed and under review
- UART (2021) – contributed and under review

Commercially Available Device Services through community members

- File Exporter in C
- BLE in C
- Zigbee in C
- GPS in C
- CAN in C
- CANOpen in C
- MEMS in C
- EtherCat in C
- EtherNet/IP in C
- Profinet in C
- OPC-UA Pub/Sub in C
- ONVIF in C
- **Siemens S7 in C**
- **ZeroMQ (Q1 2021)**
- **RFID (Q1 2021)**

North side (Application Services)

Available functions as part of the application functions SDK. Those in bold are new with Hanoi.

- AES Encryption
- GZIP Compression
- ZLIB Compression
- MarkAsPushed
- PushToCore NEW!

- Device Name Filter
- Value Descriptor Filter
- Batch
- **AddTags**
- JSONLogic Filtering
- XML Conversion
- JSON Conversion
- MQTT(S) Export
- HTTP(S) POST Export
- **HTTP(S) PUT Export**

Example Open Source Application Service Connections. Those in bold are new with Hanoi

- Azure IoT Hub
- Amazon IoT Core
- IBM Watson IoT
- Cloud Event Transformation
- **Secret Retrieval Example**
- **Fledge South HTTP**
- **LLRP RFID App Service (contributed and under review)**

Release Design Decisions

- During the course of the release, some architectural design decisions were made that affect this and future EdgeX releases. The decisions could also impact adopter implementations or extensions.
- Decision was made to remove client monitoring. This could result in a backward compatibility issue if someone was relying on the clients to detect a change of location by a service. While not likely, it should be noted.
- As it pertains to the V2 API, the decision was made that Gets (queries) for object by ID are to be removed and we will have Gets by name instead. An example would be for get events by device id vs device name.
- With regard to IPv6, the EdgeX community has tested that EdgeX services will work with IPv6. However, configuration for Kong or device services running on a separate box have not been addressed for IPv6. The EdgeX community has not focused on overlay network (service to service on the Docker network). This is on the roadmap for Ireland or beyond.
- When an event (model instance usually done by a Device Service) is created, a UUID gets created and attached to the event no matter where it is created.
- In support of tagging and identifying the origin of event/reading data, a new field "tags" will be added to event to allow the system to add key/value pairs to the event to indicate its origin. It could be set with GPS coordinates, factory location, etc. For this Hanoi release, the tags field will be populated via the app service function just before export. In future releases, the tags may be populated by other implementations and by other services (such as the originating DS).