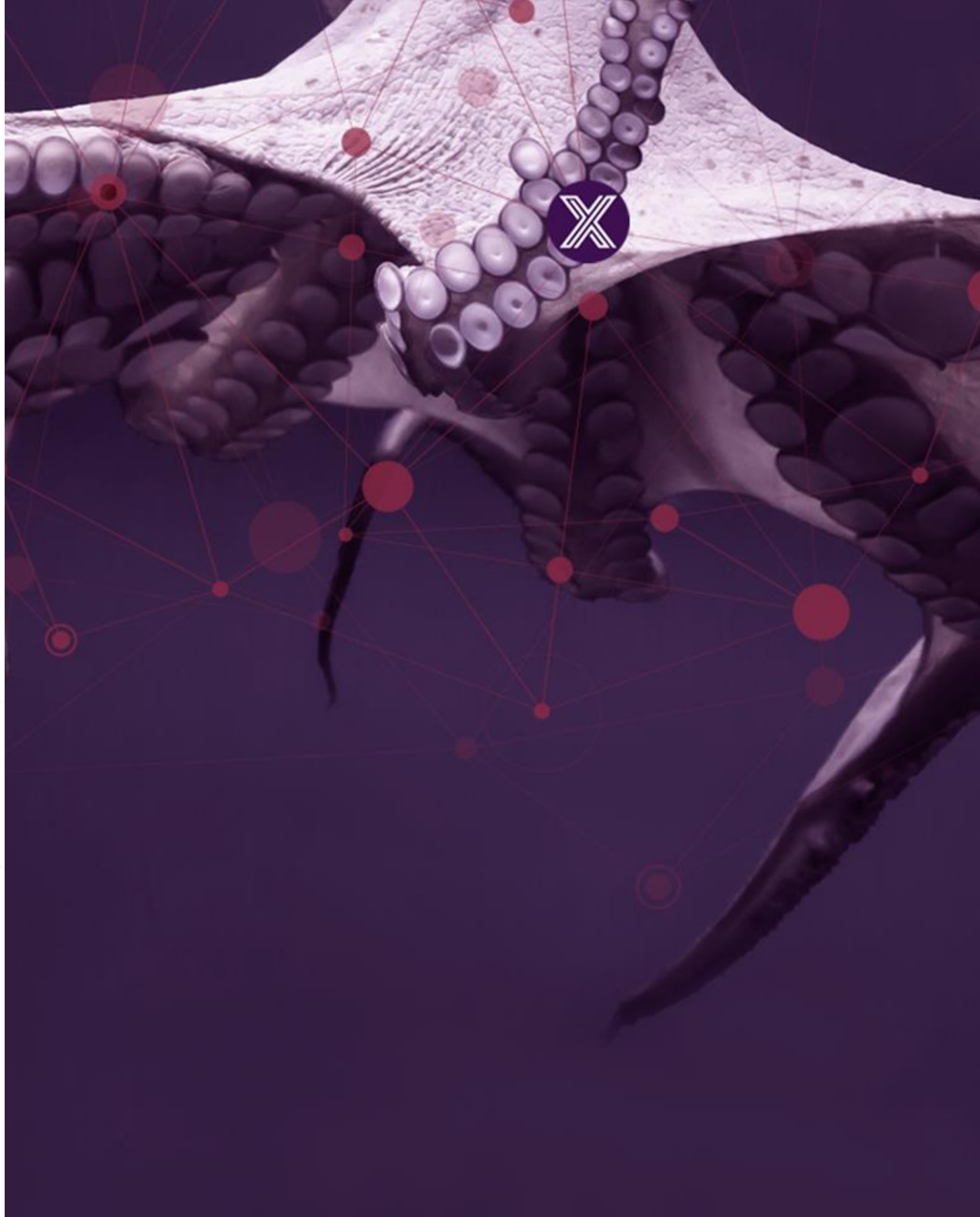
EDGEXFOUNDRY™

Application Services Design

Application Working Group
2018-12-11



LF Antitrust Policy Notice

Meeting Logistics

- Time: December 11, 2018 11am PDT – 12am PDT

Join from PC, Mac, Linux, iOS or Android: https://zoom.us/j/611544838

Or iPhone one-tap :

US: +16465588656,,611544838# or +16699006833,611544838#

Or Telephone:

Dial(for higher quality, dial a number based on your current location):

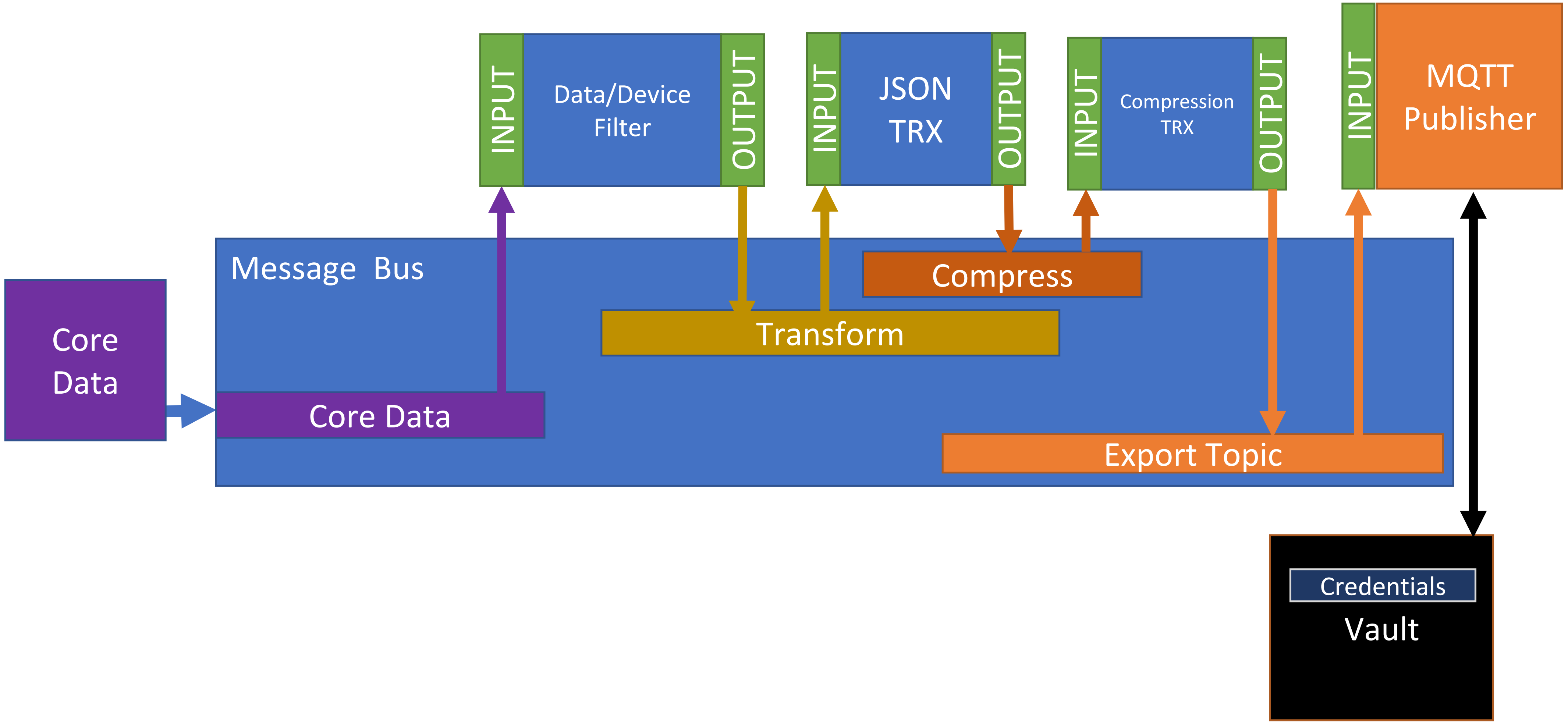
US: +1 646 558 8656 or +1 669 900 6833

or +1 855 880 1246 (Toll Free) or +1 877 369 0926 (Toll Free)

Meeting ID: 611 544 838

International numbers available: https://zoom.us/u/aoLL4E9yo

EdgeX – Application Services



Message Bus Config

- Set of preconfigured Topics/Channels
  - Core data
  - Validated Core data
  - Transform
  - Compress
  - Export
  - Custom 1…n?
  - Create topics on boot and provide config mechanism (bootstrapping process)
  - At the moment core data uses one single topic/channel in 0MQ
  - Abstraction layer between Mbus and Microservices
- Security – Who can connect to Message bus?
  - Pick a message bus which supports security and implement in future

Discussion on Message Bus

- How to discover the topics/channels on the Message bus
  - Register to Sys-config or use consul
  - Provide example set of topics which are not mandatory
- ZeroMQ - Issue reported running it on Windows (40% developers target windows - developer survey).
  - ZeroMQ can run in container in Windows
  - Docker has performance issues on Windows (order of magnitude slower - last year data)
  - There is ZeroMQ installer for Windows - Need to test and talk to Jim on the issue he reported.
- Pluggable Implementation

Discussion on Message Bus

- Message Bus - Single vs Multiple
- Pluggable Implementation
  - Need an SDK, share library, shared objects
  - Abstraction interface and APIs
- Properties that Implementation Offers:
  - Architecture Support (x86/ARM, 64/32bit)
  - Footprint
  - Scalability
  - Guaranteed Delivery
  - OS Support (Linux/Win)
  - Ordering
  - Language Support (Go, C)
  - Footprint
  - QoS
  - Performance

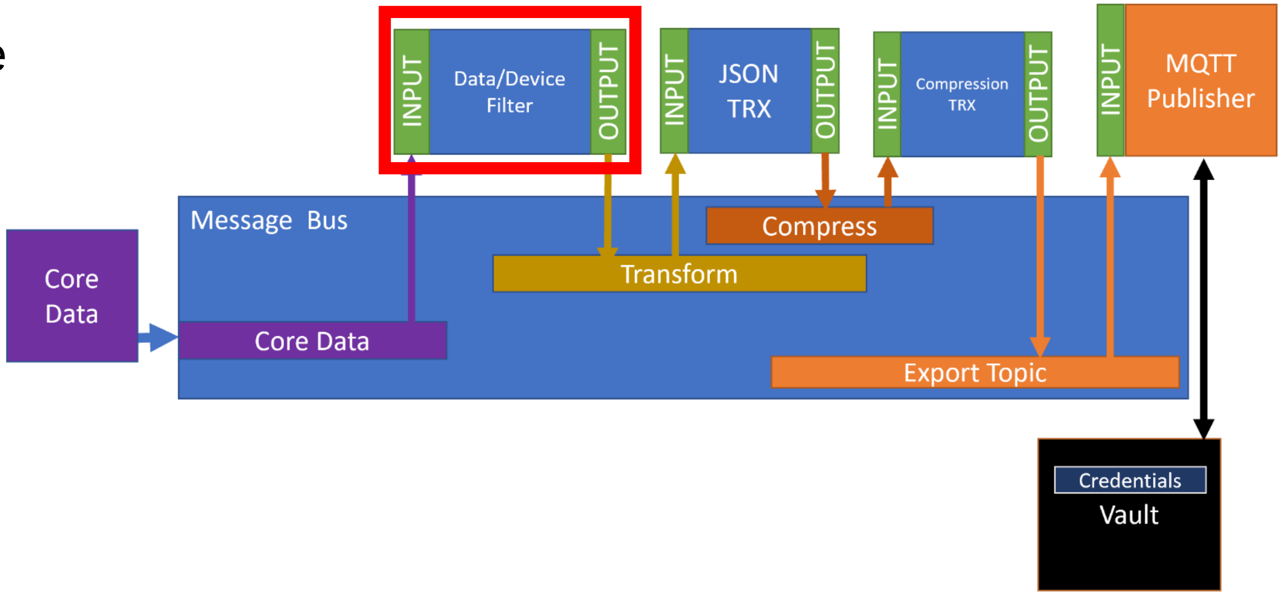
- Filtering
  - Tenancy,
  - Security,
  - Connection
- First implementation: split everything in export-distro except MQTT into independent services that use the MQTT exporter for the message bus

Message Bus Options

Name	Licence	Size	Prd. Ready	Lang.	Client	OS Support
0MQ						
Nanomsg						
GRPC						
Thrift						
DPS (Intel)						https://github.com/intel/dps-for-iot
Mosquito	EPL/EDL	very lightweight	Yes	C		Linux, Win, Mac
Paho	EPL-1.0	lightweight	Yes	C	C, Go, Java, ...	Linux, Win, Mac (client)
Apache ActiveMQ	Apache License 2.0		Yes	Java	Java, C, C++, C#, ...	Linux, Win, Mac
Apache Apollo						
RabbitMQ	MPL 1.1.	lightweight	Yes	Erlang	Erlang,Java, .NET, PHP, Python, JavaScript, Ruby, Go	Linux, Win, Mac
Qpid	Apache License 2.0			J, C++		Linux, Win, Mac
NATS	Apache License 2.0	lightweight	Yes	Go		Linux, Win, Mac

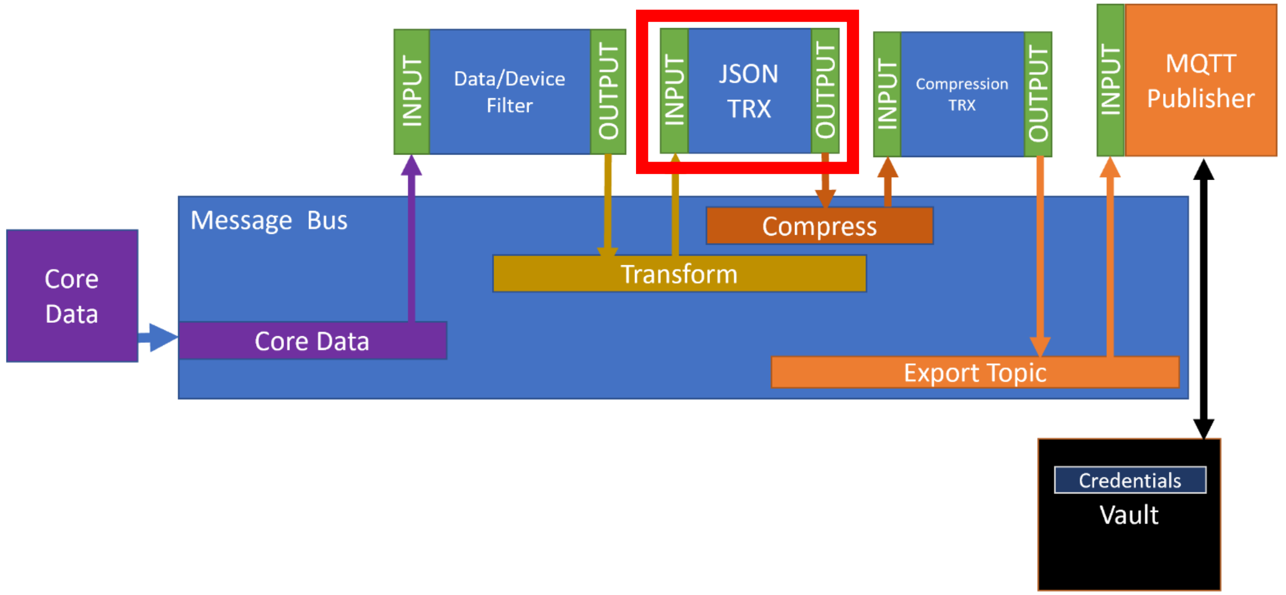
Data Device Filter

- Data device Filter service is used for data filtering:
  - Only give me readings from device A
  - Filter by Value
  - Only give me readings regarding temperature
  - Only give me readings from devices which belong to user A
  - …
- Should we have multiple Data Device filter Services? Yes
- Should the different services output to different message bus topics/channels?
- Should this microservice be implemented as example of Serverless function?



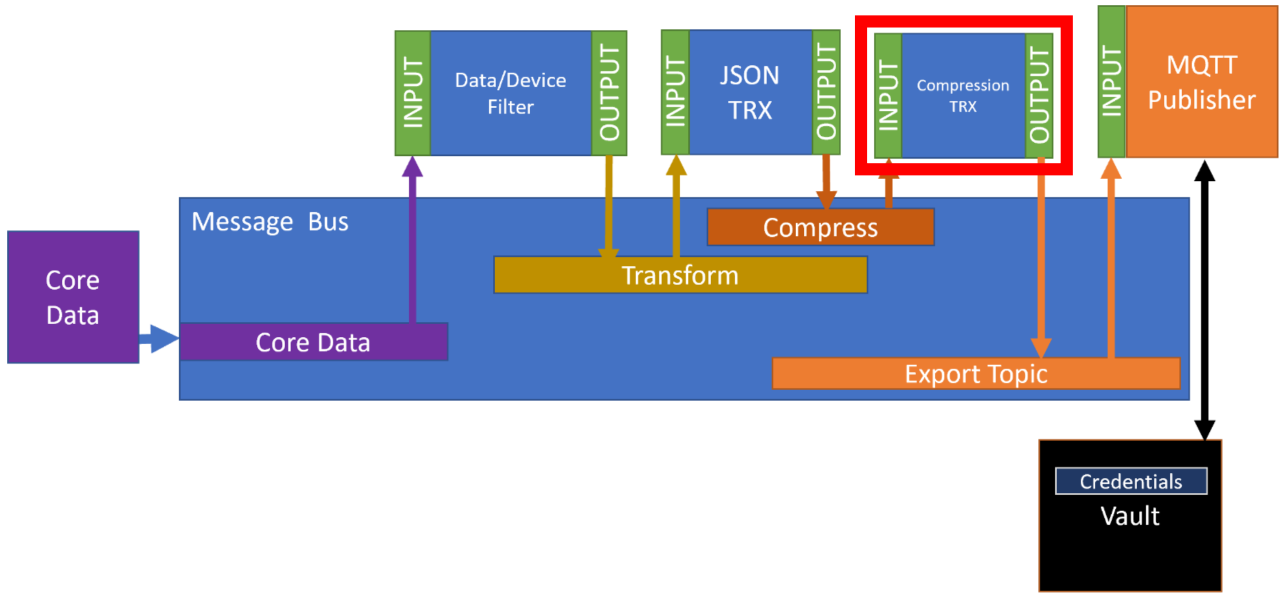
Transformation Service (JSON)

- Transformation Service transforms the validated data:
  - Convert C to F values
  - Convert meter to feet values
  - Convert CBOR to Protobuf
  - Converts data to
    - JSON
    - XML
    - CSV
    - ……
- Enrich (add device metadata to reading)
- Use Go Kit Framework



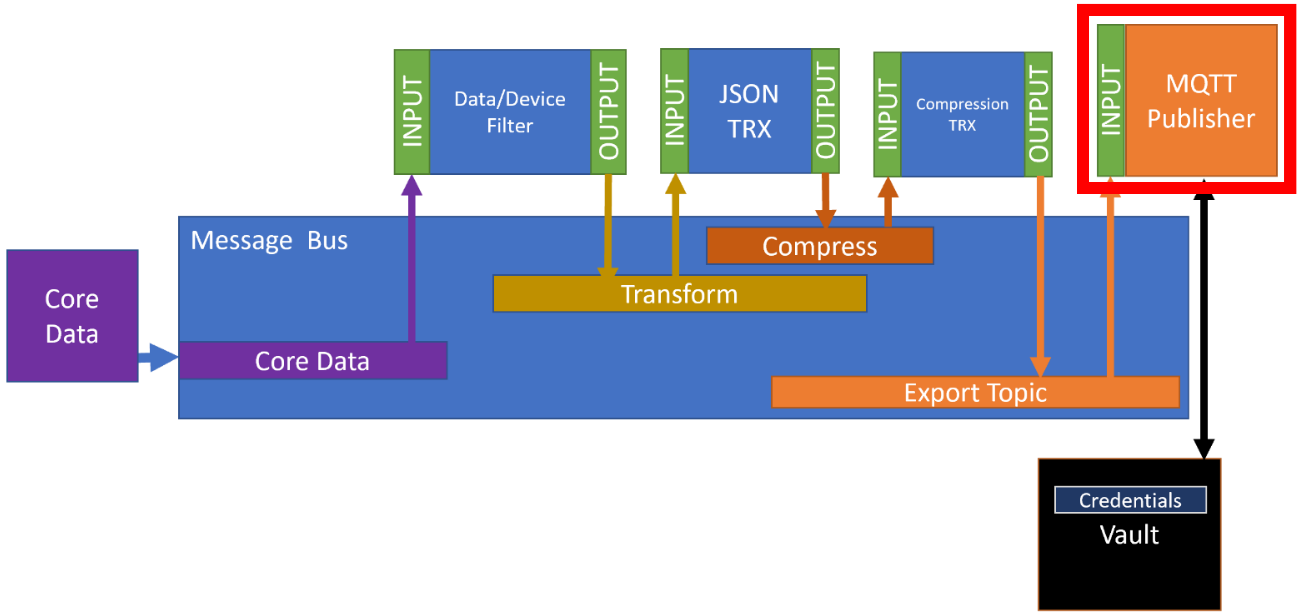
Compress Service

- This services does compression as a final part of data processing in the application services before exporting data to the target
- really different kind of transformation
- Use Go Kit Framework



Export Service – MQTT Exporter

- MQTT protocol
- Credentials Stored in Vault
- Are we able to store connection string in Vault
- Use Go Kit Framework
- Sync with System Management Agent on additional metrics and logging



Scope of Initial Work

- These tasks are to create 2 new micro services using GoKit framework and a message bus abstraction to do the following:
  - Microservice for pulling data from Core data via ZeroMQ and publishing that data to a message bus (name TBD, for now referring to it CoreDataProvider)
  - Microservice for pulling data (subscribing) from the message bus and publishing it a to a Cloud base MQTT (MQTT Exporter)
  - Message bus abstraction/plugin architecture so any message bus that has an implementation for our abstraction can be used.
- Includes a reference message bus implementation for TBD message bus.

EDGE X FOUNDRY™

Tasks for Message Bus abstraction/plugin

- Define minimal abstraction API that is required
- Create plug-in wrapper class that implements minimal abstraction API
  - Must load implementation library (SO file) from a known plug-ins directory
  - Maps/forwards func calls to loaded implementation
- Create implementation library (SO file) of minimal abstraction API for **TBD** Message Bus.

Tasks for CoreDataProvider

- Spin up on GoKit (if not already)
  - https://gokit.io/examples/
  - Specifically want to use GoKit for REST API, Logging, metrics and tracing
- Define REST API if any needed
- Create initial micro service using GoKit
- Create Connection to Core Data via ZeroMQ to receive raw data
- Create Connection to Message Bus via abstract message bus above to publish raw data
- Implement handler that receives the raw data from ZeroMQ and publishes it on the Message Bus.
- Implement REST API if any needed
- Implement standard configuration and registration with Consul
  - Includes watcher for configuration changes
- All implementations above include logging, metrics and tracing via GoKit.

EDGE X FOUNDRY™

Tasks for MQTT Data exporter

- Spin up on GoKit (if not already)
  - https://gokit.io/examples/
  - Specifically want to use Gokit for REST API, Logging, metrics and tracing
- Create initial micro service using GoKit
- Create Connection to Message Bus via abstract message bus above to received raw data
- Implement storing/loading MQTT authorization to/from vault.
- Implement standard configuration and registration with Consul
  - Include watcher for configuration changes
- Implement MQTT exporter by copying existing MQTT code from current export distro
  - edgexfoundry/edgex-go/internal/export/distro/mqtt.go
- Implement handler that receives the raw data from message bus and sends it to cloud via MQTT exporter
- All implementations above include logging, metrics and tracing via GoKit.

EDGE X FOUNDRY™

Standard Template for GoKit micro service

- This begs for an EdgeX standard micro service quick start template based on GoKit that:
- Sets up REST API support
- Implements standard configuration and registration with Consul
  - Includes watcher for configuration changes
- Other common stuff for logging, metrics, tracing, etc.