EdgeX App Functions SDK

Seoul F2F Training

Seoul Korea, April 29 – May 2, 2019

EDGE X FOUNDRY™

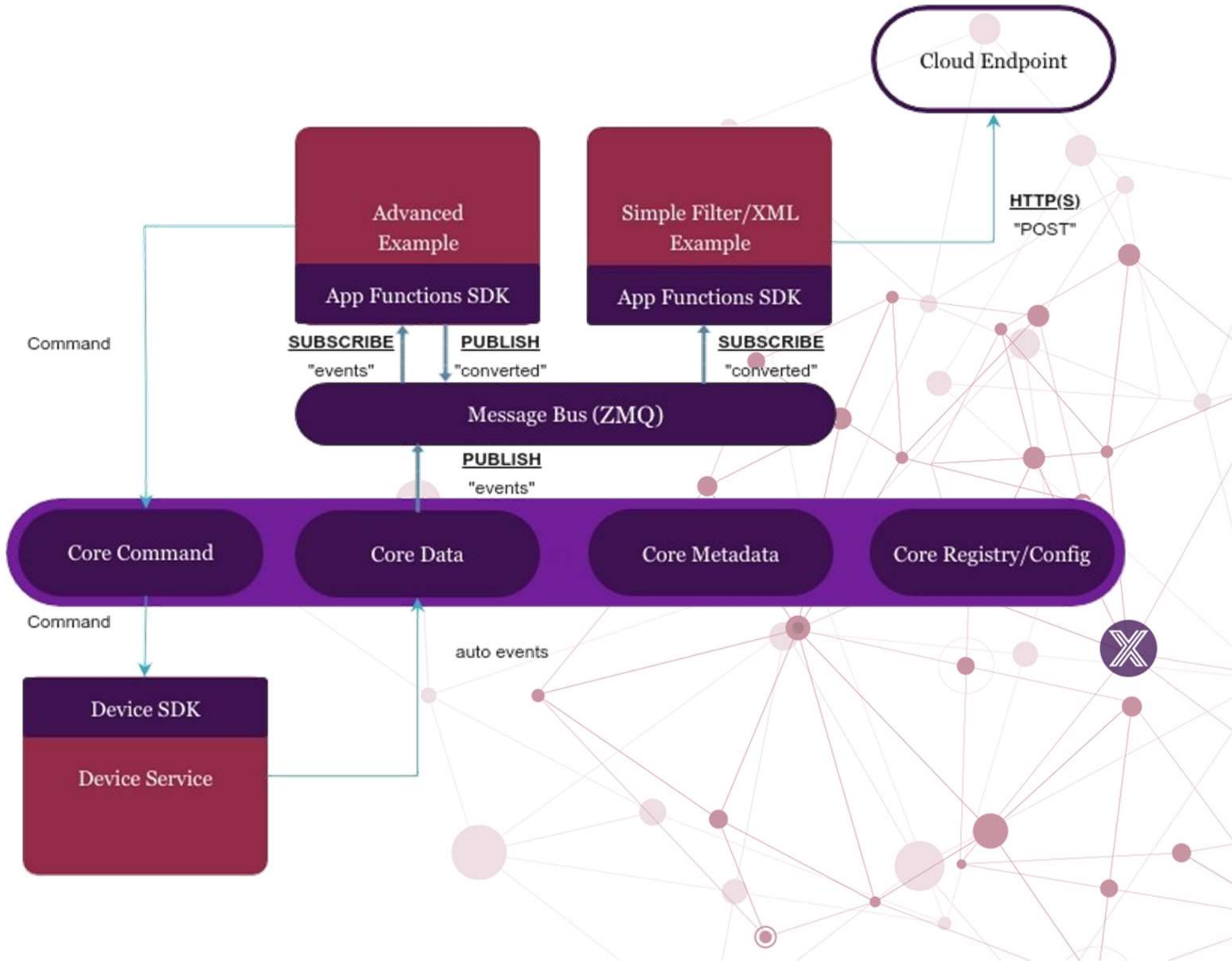
Agenda

- Overview
- Why use the SDK?
- How to use the SDK
  - Tutorial
  - Demos
  - Exercises
- What's missing and what's coming in Fuji
- Q&A

EDGE X FOUNDRY™

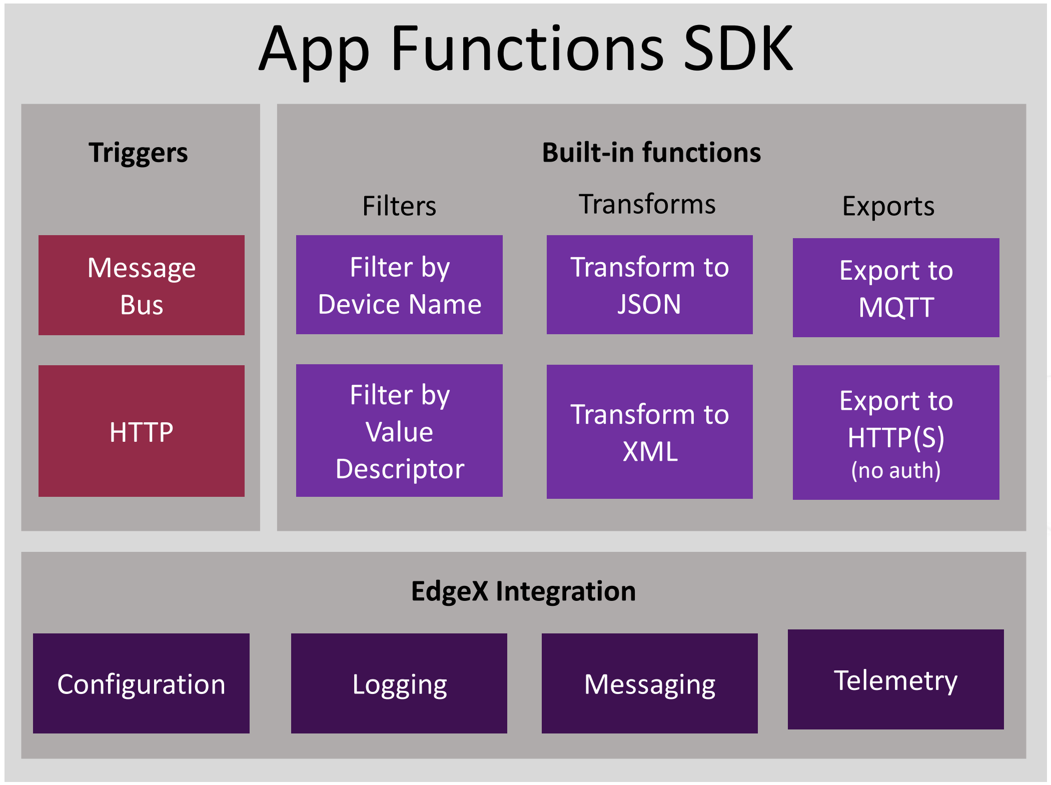
Overview

The App Functions SDK enables rapid development of Golang Application Services for EdgeX.



Why use the SDK?

The App Functions SDK provides all the EdgeX integration out of the box and some built-in functions for filtering, transforming and exporting.



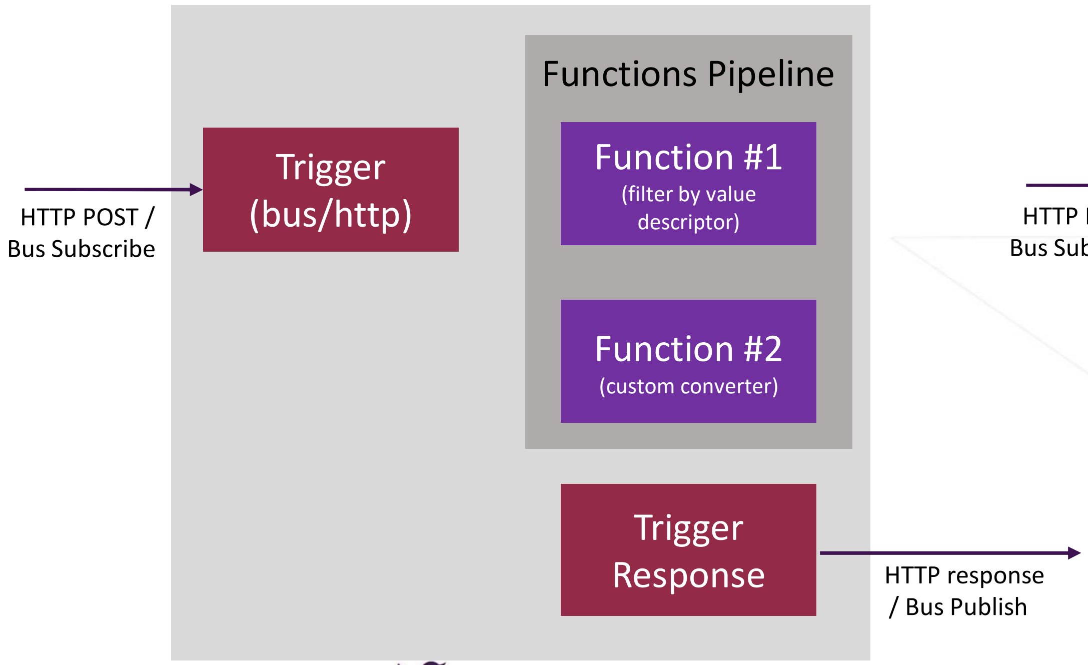
Integrations details:

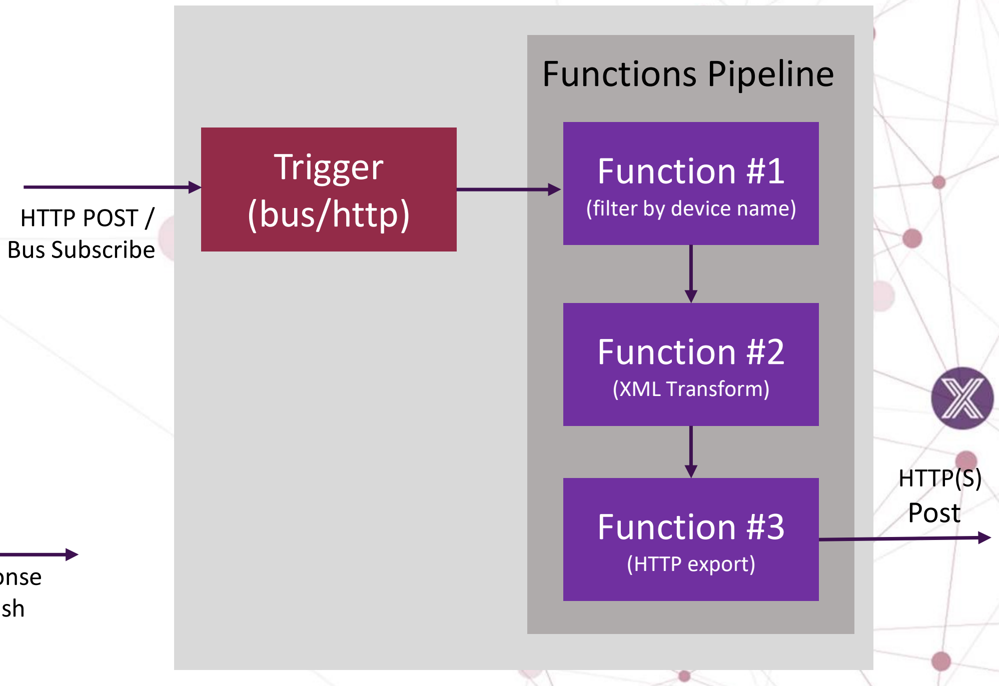
- Messaging
    - Message Bus
    - HTTP
    - JSON/CBOR messages
- Configuration
    - Local
    - Remote Registry
- Logging
    - Correlation ID tracing
    - Access to logging client
- System Management Telemetry
    - CPU Usage
    - Memory Usage
    - etc.
- Events marked as exported

How to use the SDK (Overview)

Events flow into the SDK via triggers. One or more functions (built in or custom) operate on the data via the Functions Pipeline. The data can then be exported, returned as a response to the trigger or both.

Developers' main task is to create any required custom functions and setup their Functions Pipeline. The rest is boilerplate boot strapping code.


HTTP POST / Bus Subscribe
Trigger (bus/http)
Functions Pipeline
Function #1 (filter by value descriptor)
Function #2 (custom converter)
Trigger Response
HTTP response / Bus Publish


HTTP POST / Bus Subscribe
Trigger (bus/http)
Functions Pipeline
Function #1 (filter by device name)
Function #2 (XML Transform)
Function #3 (HTTP export)
HTTP(S) Post

EDGE X FOUNDRY™

How to use the SDK (Tutorial - Example)

Let's take a look at a simple example that creates a pipeline to filter particular device ids and subsequently transform the data to XML

How to use the SDK (Tutorial - Triggers)

Triggers determine how the app functions pipeline begins execution. In the simple example provided above, an HTTP trigger is used.

Message Bus Trigger

- A message bus trigger will execute the pipeline every time data is received off of the configured topic.
- Message bus connection configuration must be specified

```
[Binding]
Type="messagebus"
SubscribeTopic="events"
PublishTopic=" “

[MessageBus]
Type = 'zero'
 [MessageBus.SubscribeHost]
    Host = 'localhost'
    Port = 5564
    Protocol = 'tcp'
 [MessageBus.PublishHost]
    Host = '*'
    Port = 5565
    Protocol = 'tcp'
```

HTTP Trigger

- Designating an HTTP trigger will allow the pipeline to be triggered by a RESTful POST call to http://[host]:[port]/trigger/.
- edgexcontext.complete([]byte outputData) - Will send the specified data as the response to the request that originally triggered the HTTP Request.

```
[Binding]
Type="http"
```

Important Note: Publish Host for ZMQ MUST be different for every topic you wish to publish to since the SDK will bind to the specific port. 5563 for example cannot be used to publish since EdgeX Core Data has bound to that port. Similarly, you cannot have two separate instances of the app functions SDK running publishing to the same port.

How to use the SDK (Tutorial - Misc)

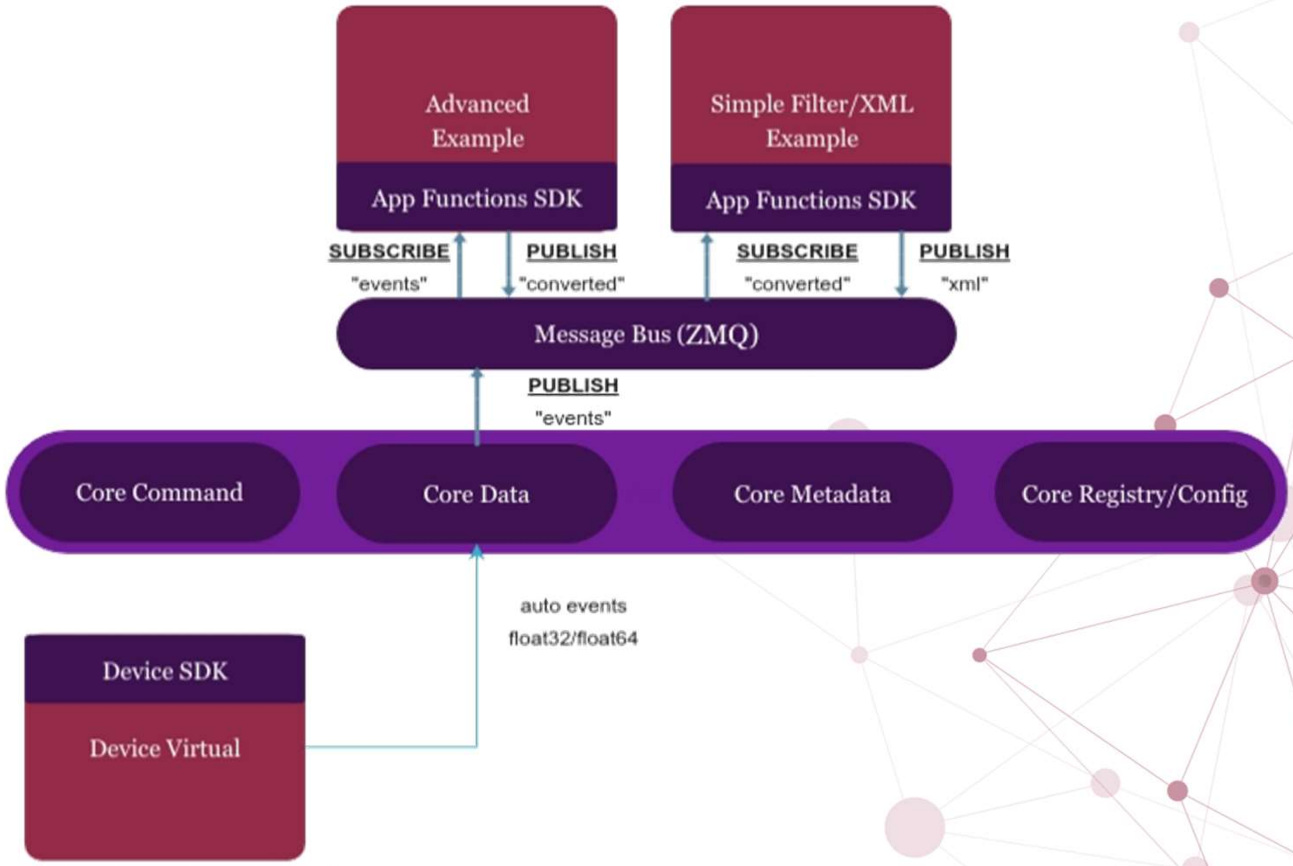
- Configuration
  - The ApplicationSettings API returns a map[string] string containing the contents on the ApplicationSetting section of the configuration.toml file.

```
[ApplicationSettings]
ApplicationName = "advanced-filter-convert-publish"
ValueDescriptors = "RandomValue_Float32, RandomValue_Float64"
```

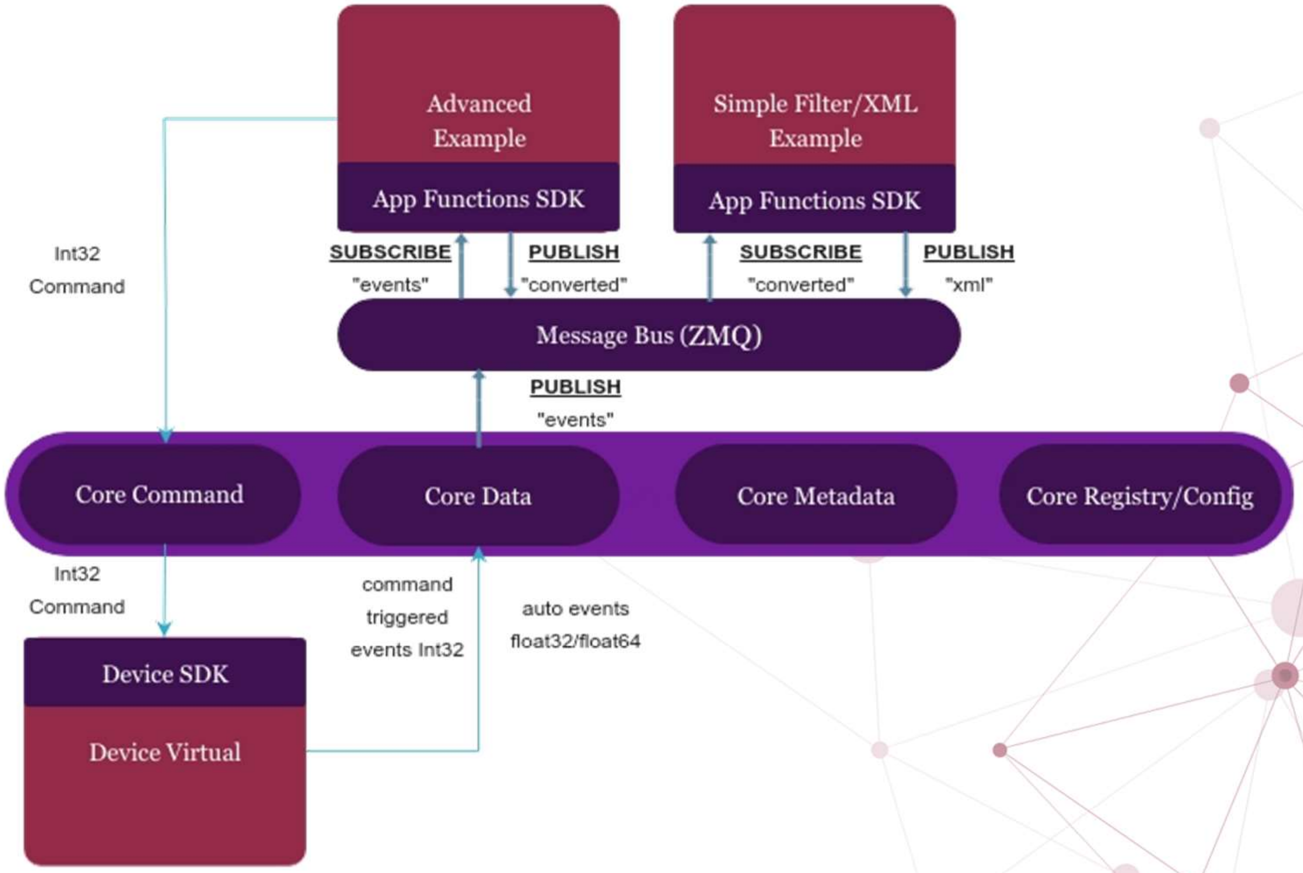
- Error handling
  - Each transform returns a ***true*** or ***false*** as part of the return signature. This is called the ***continuePipeline*** flag and indicates whether the SDK should continue calling successive transforms in the pipeline.
  - ***return false, nil*** will stop the pipeline and stop processing the event. This is useful for example when filtering on values and nothing matches the criteria you've filtered on.
  - ***return false, error***, will stop the pipeline as well and the SDK will log the errorString you have returned.
  - Returning **true** tells the SDK to continue, and will call the next function in the pipeline with your result.

How to use the SDK (Demo)

https://github.com/edgexfoundry/app-functions-sdk-go/blob/master/examples/advanced-filter-convert-publish/README.md



How to use the SDK (Hands on Exercise)



What missing and what's coming in Fuji?

The following are being considered for the Fuji release

- Archive the current Export Services and Export Client micro service
  - ➢ App functions SDK at minimal parity
- Send data to the rules engine
- Provide tutorials on how to build a cloud-supporting endpoint (Azure, AWS, etc.
- Provide Vault integration (request secrets)
- Expand triggers: Timer based
- Expand current set of built in transforms
  - ➢ Filters (Value ranges, location based/geo-fenced, De-duplication values, Eliminate noise from readings, Time based)
  - ➢ Format transforms (CSV, YAML, TOML, RAML)
  - ➢ Data Transform (Unit conversion, Cloud ready i.e. Azure, AWS, etc.)
  - ➢ Encryption (payloads) – depends on vault integration
  - ➢ Signing – depends on vault integration – near last step in pipeline
  - ➢ Compression (ZIP, TAR.GZ, Lossy vs lossless compression)
  - ➢ Media transformation (JPG to GIF, MPEG to WAV, etc)
  - ➢ Enrich (metadata additions, commands, ...)
  - ➢ Additional endpoints – maybe additional protocols
  - ➢ HTTPS and MQTTS (with token exchange via OAuth, JWT, ...)
- EdgeX device commands to the north side from the Cloud

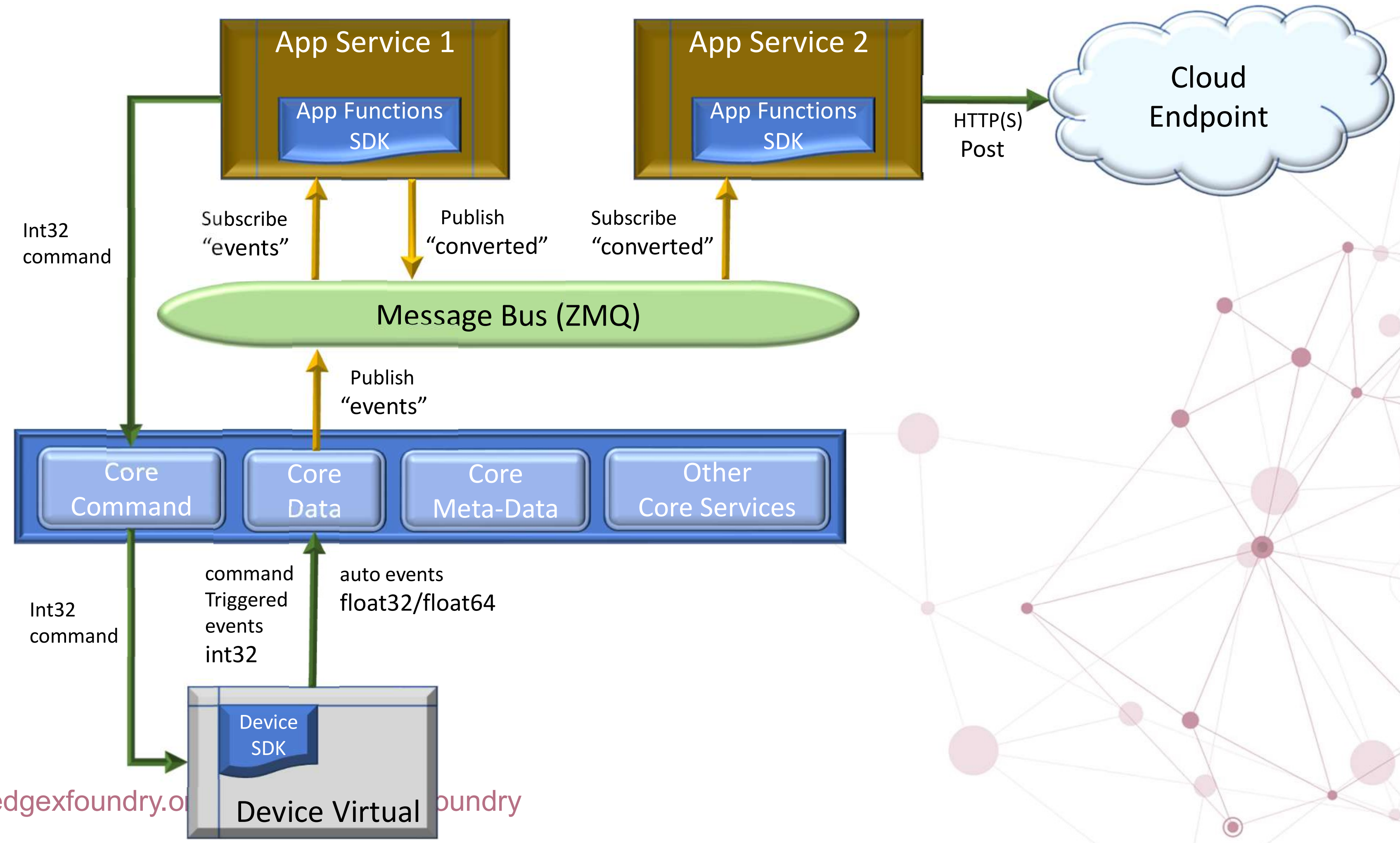
Q&A

Thank you!

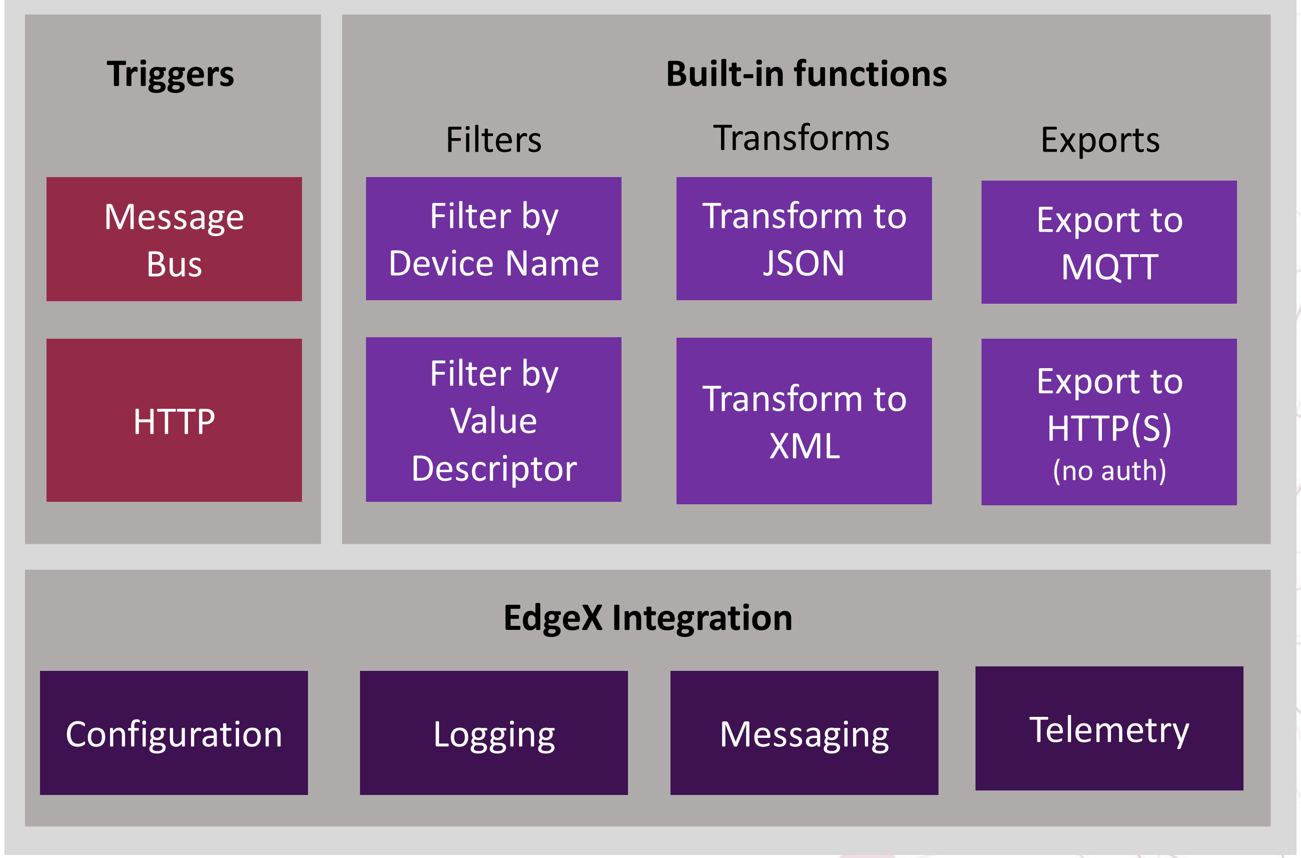
edgexfoundry.org | @edgexfoundry

EDGE X FOUNDRY™

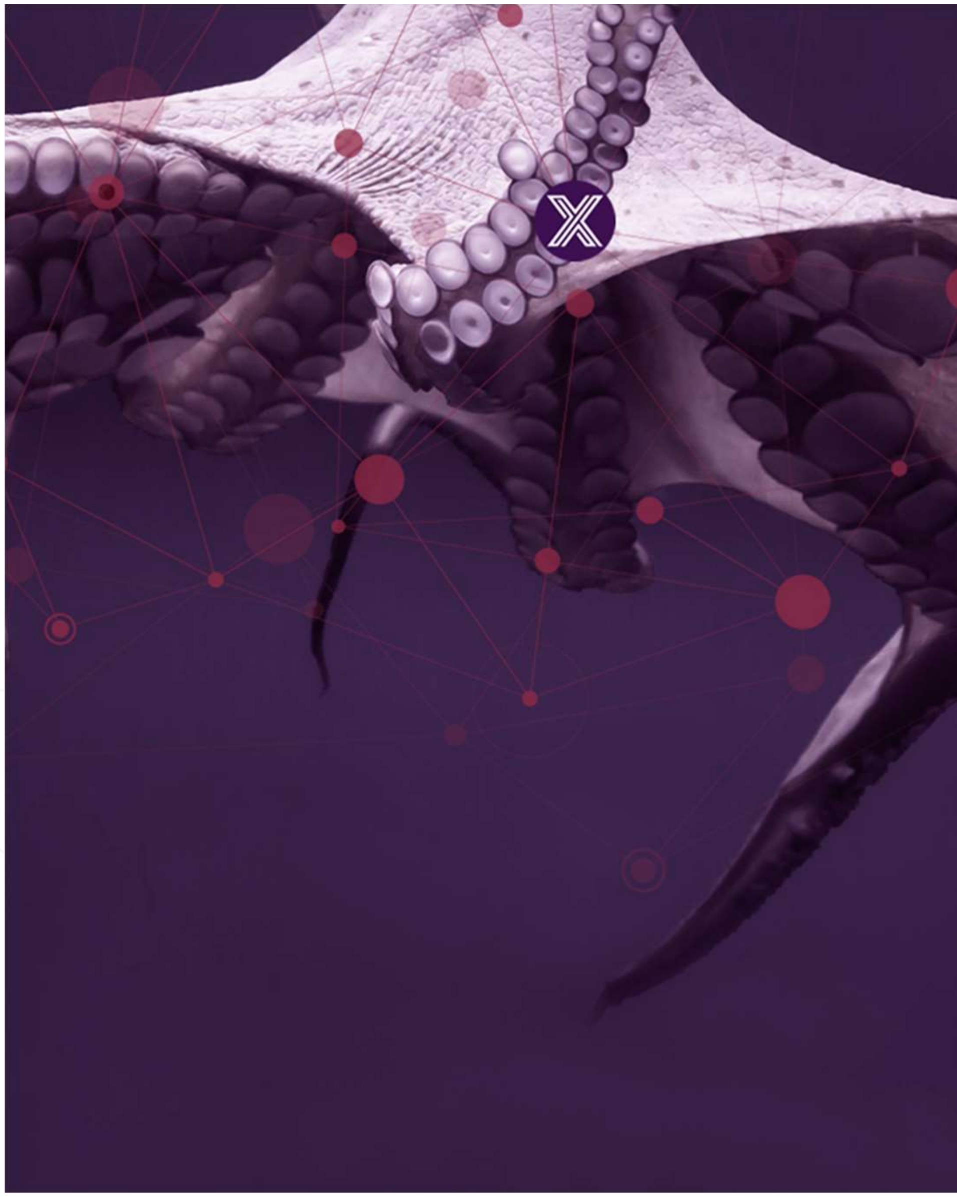
Backup


App Service 1
App Functions SDK
App Service 2
App Functions SDK
Cloud Endpoint
HTTP(S) Post
Int32 command
Subscribe "events"
Publish "converted"
Subscribe "converted"
Message Bus (ZMQ)
Publish "events"
Core Command
Core Data
Core Meta-Data
Other Core Services
Int32 command
command Triggered events int32
auto events float32/float64
Device SDK
Device Virtual

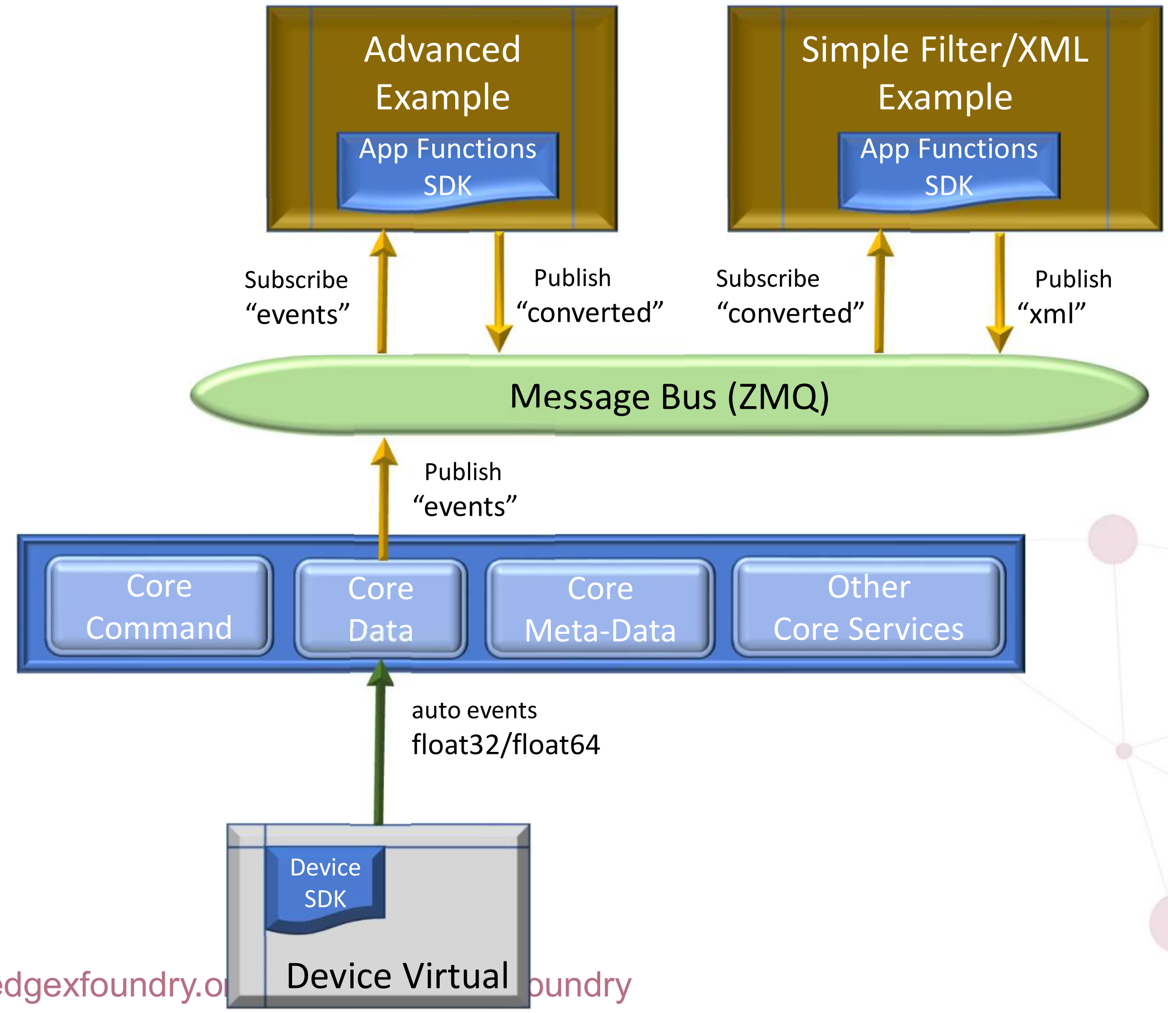
App Functions SDK

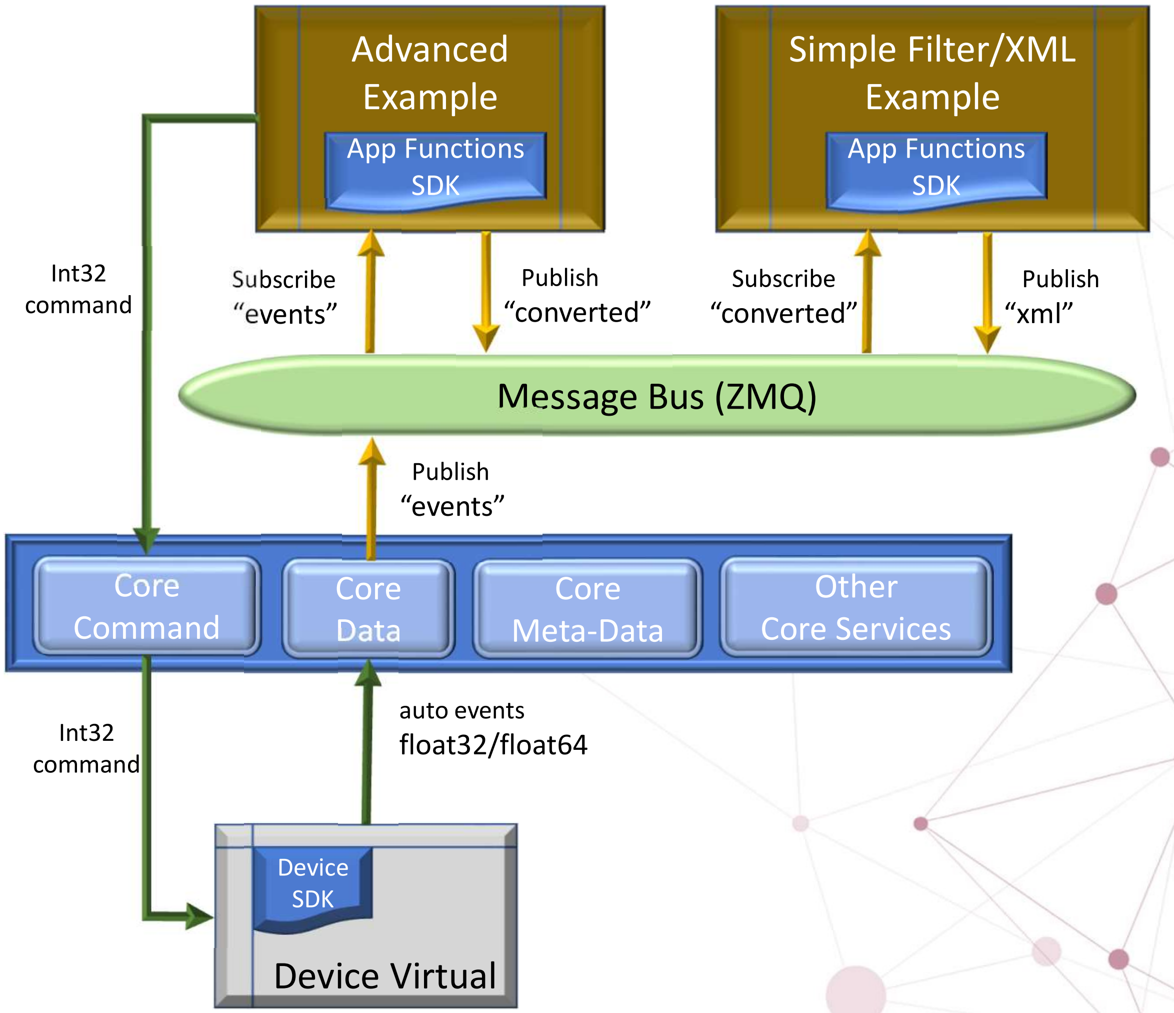


EDGEXFOUNDRY™



edgexfoundry.org | @edgexfoundry


Advanced Example
App Functions SDK
Simple Filter/XML Example
App Functions SDK
Subscribe "events"
Publish "converted"
Subscribe "converted"
Publish "xml"
Message Bus (ZMQ)
Publish "events"
Core Command
Core Data
Core Meta-Data
Other Core Services
auto events float32/float64
Device SDK
Device Virtual


Advanced Example
App Functions SDK
Simple Filter/XML Example
App Functions SDK
Int32 command
Subscribe "events"
Publish "converted"
Subscribe "converted"
Publish "xml"
Message Bus (ZMQ)
Publish "events"
Core Command
Core Data
Core Meta-Data
Other Core Services
Int32 command
auto events float32/float64
Device SDK
Device Virtual

EDGE X FOUNDRY™

How to use the SDK (Overview)

- The SDK is built around the idea of a “**Functions Pipeline**".
- A pipeline is a collection of various functions that process the data in the order that you've specified.
- The pipeline is executed by the specified trigger in the configuration.toml.
- The first function in the pipeline is called with the event that triggered the pipeline (ex. events.Model).
- Each successive call in the functions pipeline is called with the return result of the previous function.