Welcome!
Today we will be covering the App Functions SDK and the App Service Configurable.

EDGEXFOUNDRY

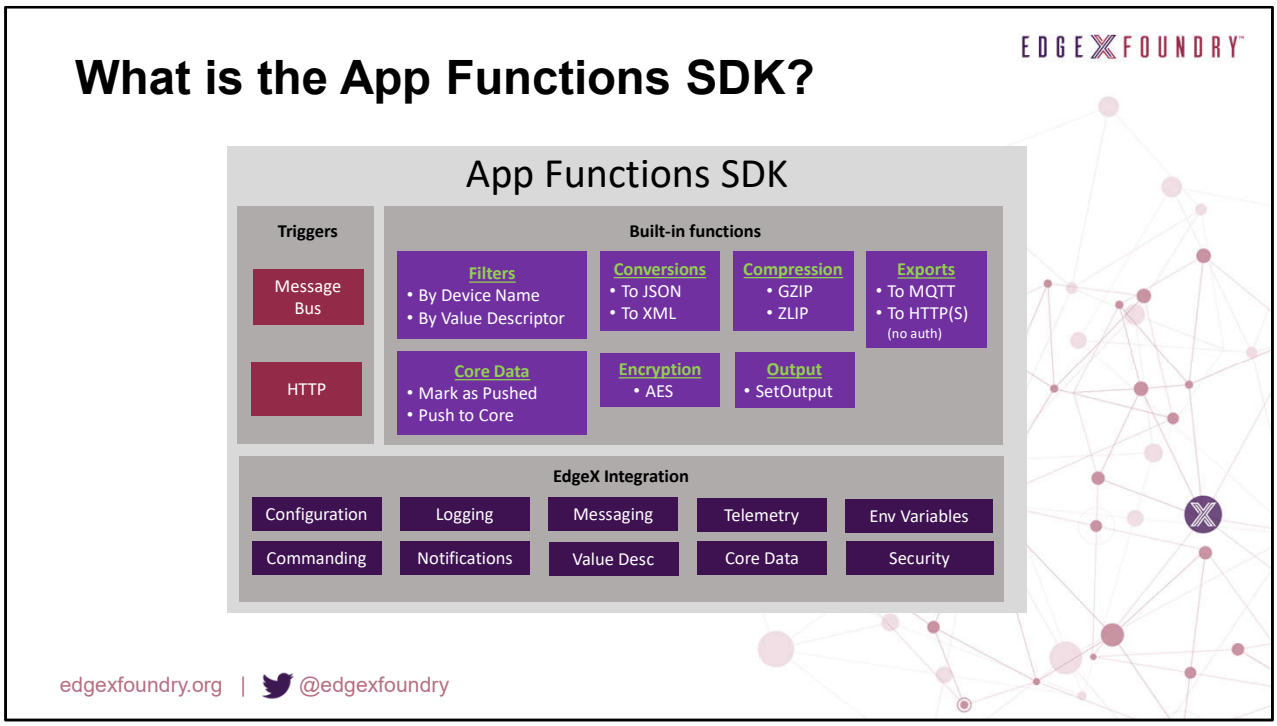
Agenda

- App Functions SDK (overview)
- App Service Configurable
- Labs (App Service Configurable)
- App Functions SDK (detailed)
- Labs (custom application services)

edgexfoundry.org | @edgexfoundry

We'll start with a high level overview of the App Functions SDK,
then cover the App Service Configurable,
Followed by a couple labs using the App Service Configurable

The we'll return a detailed look at the App Functions SDK
And wrap up with some labs using the SDK to create custom Application Services



The App Functions SDK provides the capability to process data via a functions pipeline using built-in functions for filtering, transforming, exporting, etc. and/or you own custom functions.

It also includes all the EdgeX integration for you, so you don't have to worry about loading configuration, setting up logging, etc.

EDGEXFOUNDRY

Functions Pipeline

- The SDK is built around the idea of a “**Functions Pipeline**".
- The pipeline is a collection of various functions that process the data in the order that you've specified.
- The pipeline is executed when the configured trigger receives data.
- The first function in the pipeline is called with the data that triggered the pipeline. This is either an Edgex Event or the specified **TargetType**.
- Each successive function in the pipeline is called with the return result of the previous function.

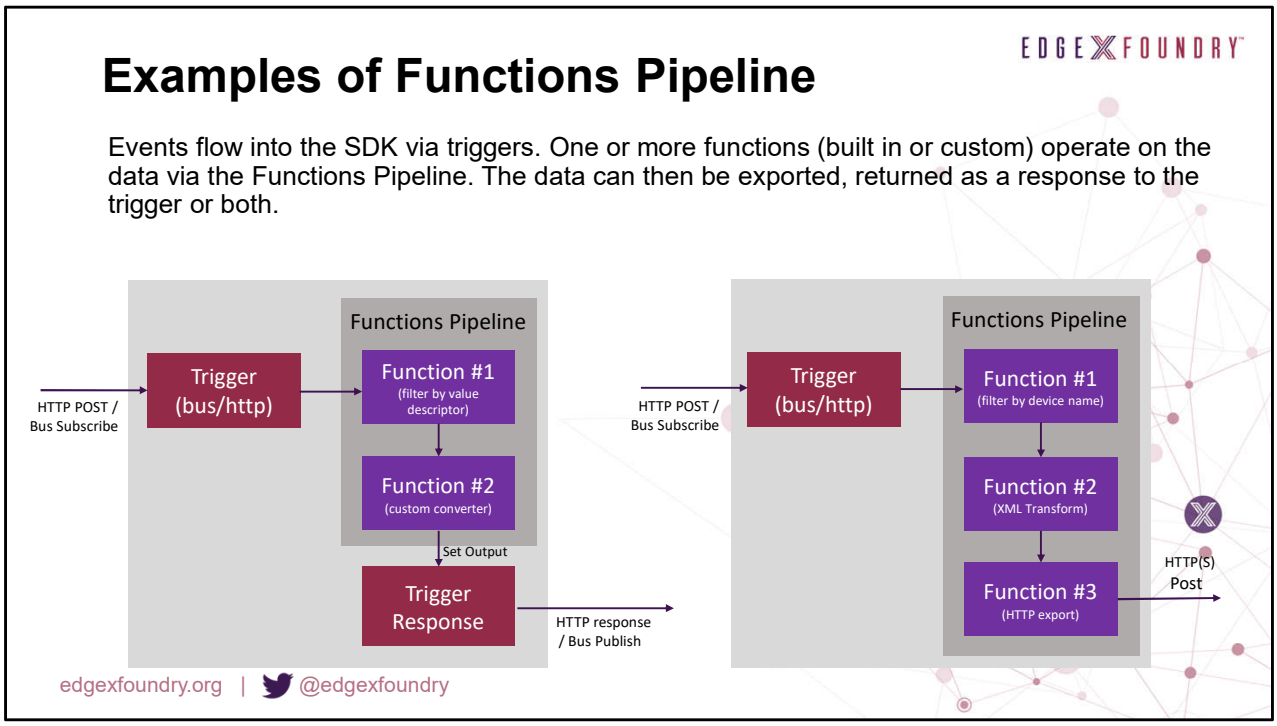
edgexfoundry.org | @edgexfoundry

So what is a functions pipeline?

We currently have Message Bus or HTTP triggers.

TargetType is specified when initializing the SDK. By default it is an EdgeX Event, but It can be a custom type defined by your application or simply a byte slice for raw data.

Target Type is useful when the data received by the trigger isn’t coming from Core Data.



Here is a couple examples of a functions pipeline.

One has the final result retuned as the “trigger response” and the other’s final result is exported to HTTP endpoint.

What is App Service Configurable?

EDGEXFOUNDRY

App Service Configurable

App Functions SDK

- Allows deploying Application Services without coding
- Multiple instances can be deployed, each with a different profile.
- Function Pipeline Configuration is “Writeable” allowing live updates from Registry
- Function Pipeline Configuration is limited to the built in functions

edgexfoundry.org | @edgexfoundry

App Service Configurable is an Application Service built on the App Functions SDK which allows defining the functions pipeline via configuration.

EDGEXFOUNDRY

Profiles

Profiles define the uniqueness of each instance

- http-export
- mqtt-export
- push-to-core
- rules-engine
- default (no-profile)

Profile name is used in Registry service key.

- i.e. AppService-mqtt-export

edgexfoundry.org | @edgexfoundry

The following profiles are provided in the Docker container

http-export

Configures pipeline to filter, json transform and export data received from Message Bus via HTTP.
FilterByDeviceName, **HTTPPostJSON** functions configuration parameters require modification. Includes calling **MarkAsPushed** function after successful export

mqtt-export

Configures pipeline to json transform and export data received from Message Bus via MQTT.
MQTTSend function configuration parameters require modification. Includes calling **MarkAsPushed** function after successful export

push-to-core

Configures pipeline to push data from HTTP trigger to Core Data.
PushToCore function configuration parameters require modification

rules-engine

Configures pipeline and message bus to forward Event messages to Rules engine via ZMQ

default (no-profile)

Sample pipeline that filters by device, transforms to XML and sets output. Configuration contains all built-in functions so they can be configured and added to the pipeline

Writeable Pipeline Configuration

```
[Writable.Pipeline]
  UseTargetTypeOfByteArray = false
  ExecutionOrder = "FilterByDeviceName, TransformToJSON, HTTPPostJSON, MarkAsPushed"

  [Writable.Pipeline.Functions.TransformToJSON]
  [Writable.Pipeline.Functions.MarkAsPushed]
  [Writable.Pipeline.Functions.FilterByDeviceName]
    [Writable.Pipeline.Functions.FilterByDeviceName.Parameters]
      DeviceNames = ""
  [Writable.Pipeline.Functions.HTTPPostJSON]
    [Writable.Pipeline.Functions.HTTPPostJSON.Parameters]
      url = "http://somewhere.com/data"
      persistOnError = "false"
```

edgexfoundry.org | @edgexfoundry

UseTargetTypeOfByteArray – Boolean

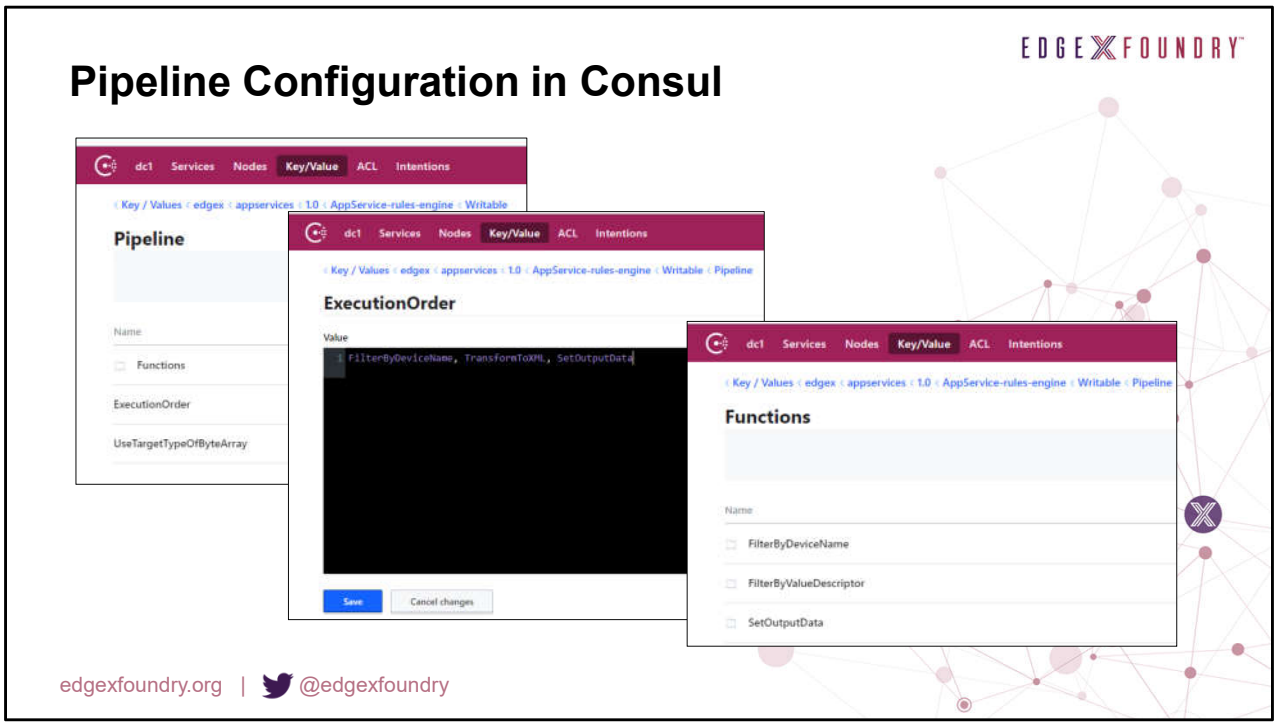
Allows for the input data received to be raw bytes rather than an EdgeX Event

ExecutionOrder – Comma separated list of built in function names

These are the functions that will be executed and the order they will be executed in Names here must be present in the following ”functions” list

Functions – List of pipeline functions

List of eligible functions and their configuration. The function names here must match that of a built in function. Each function can only appear once in the list



Here is what the Functions Pipeline configuration looks like from Consul

EDGEXFOUNDRY

Built in Functions

Filtering
- FilterByDeviceName
- FilterByValueDescriptor

Encryption
- EncryptWithAES

Conversion
- TransformToXML
- TransformToJSON

Compression
- CompressWithGZIP
- CompressWithZLIB

CoreData
- MarkAsPushed
- PushToCore

Export
- HTTPPost
- MQTTSend

Output
- SetOutput

edgexfoundry.org | @edgexfoundry

Here is a list of the built-in functions that are available to use via configuration.

EDGEXFOUNDRY

Environment Variable Overrides For Docker

```
edgex_registry: consul://edgex-core-consul:8500
edgex_profile : [target profile]
edgex_service : http://[service name]:[port]
Service_Host : [service name]
Clients_CoreData_Host: edgex-core-data
Clients_Logging_Host : edgex-support-logging
Logging_EnableRemote: "true"
Database_Host : edgex-mongo
Database_Username : appservice
Database_Password : password
```

edgexfoundry.org | @edgexfoundry

App Service Configurable no longer has docker specific profiles.

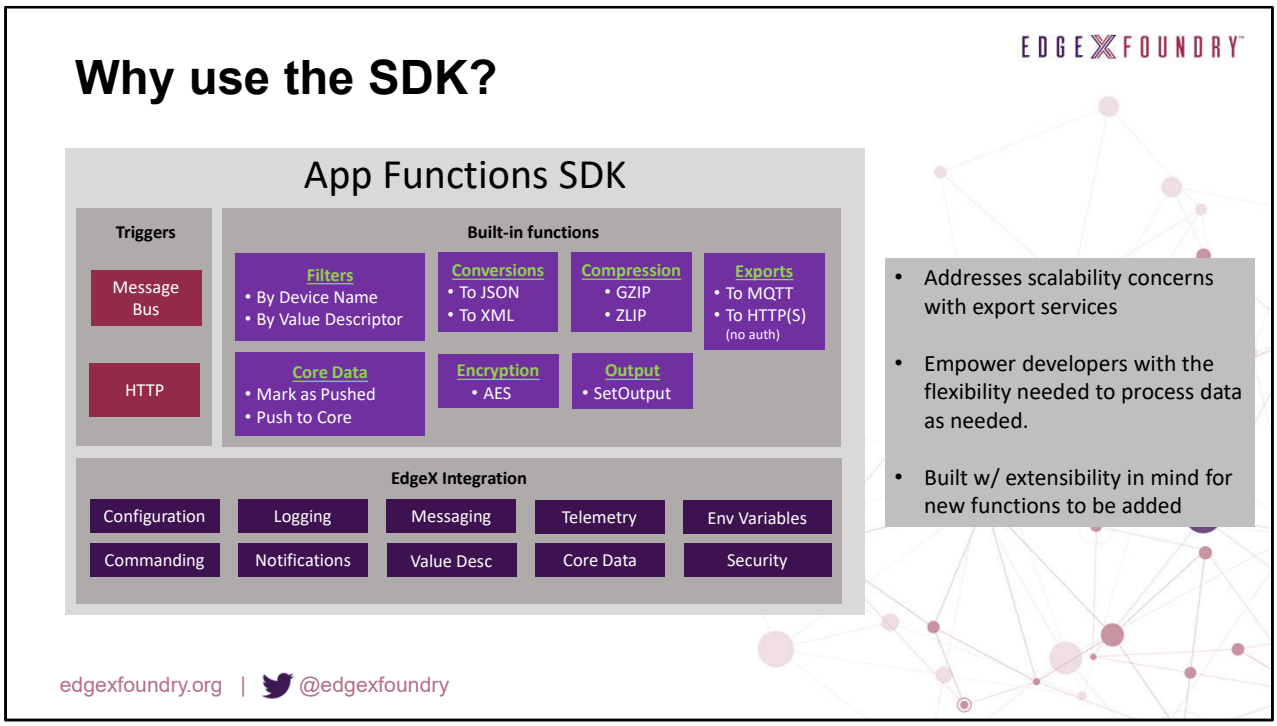
It now relies on environment variable overrides in the docker compose file for the docker specific differences

EDGEXFOUNDRY™

Labs 1 & 2
20 mins

https://github.com/rsdmike/labs

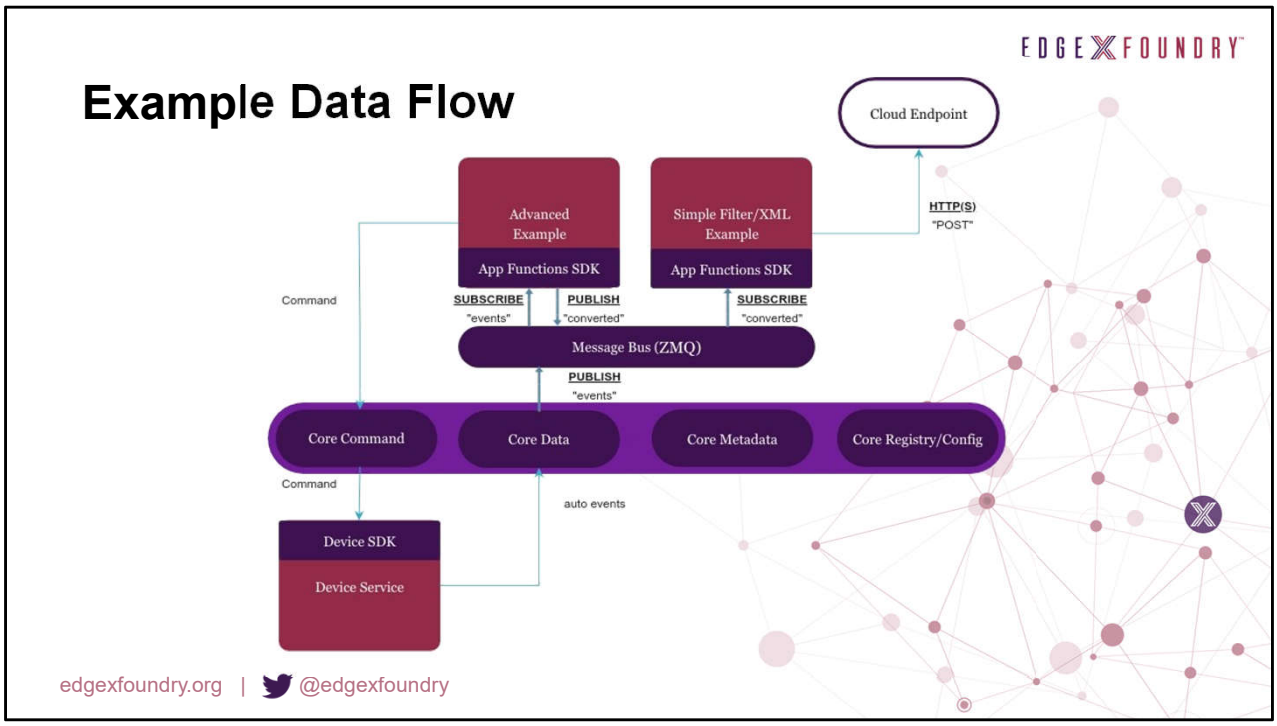
edgexfoundry.org | @edgexfoundry



The current export-distro doesn't scale since all data flows through this one service for all export registrations, becoming a bottle neck.

Application Services built upon the App Functions SDK are meant to replace export-client/export-distro after the Geneva release.

These services will scale much better as they will be single purpose and run in parallel, unlike export-distro where endpoints are called sequentially.



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

Data flows from a device service, through core data, up to App Functions SDK, to be processed, interpreted, or sent off box.

The SDK provides all the integration with EdgeX, allowing developers to focus on the custom logic of their application service.

EDGEXFOUNDRY

Creating custom Application Service

- First, create an instance of the EdgeX SDK and initialize it.

```
sdk := &appsdk.AppFunctionsSDK{ServiceKey: serviceKey}
sdk.Initialize()
```

```
sdk := &appsdk.AppFunctionsSDK{ServiceKey: serviceKey, TargetType: &MyType{}}
sdk.Initialize()
```

This is where **TargetType** can be specified.
Must be a pointer to an instance of the type

- Next, set the function pipeline.

```
deviceNames := []string{"Random-Float-Device"}
sdk.SetFunctionsPipeline(
   transforms.NewFilter(deviceNames).FilterByDeviceName,
   transforms.NewConversion().TransformToXML,
   printXMLToConsole, // Custom function
)
```

- Lastly, run the pipeline

```
sdk.MakeItRun()
```

This can all be done from the service's main function.

edgexfoundry.org | @edgexfoundry

Creating a custom Application Service is fairly simple.

1) Initialize the SDK, note here is where you specify your TargetType, if not using EdgeX Event.
2) Set you functions pipeline with built-in and custom functions
3) Run the pipeline.

EDGEXFOUNDRY

Writing a custom Pipeline Function

```
func(edgexcontext *Context, params ...interface{}) (bool, interface{})
```

edgexcontext

- Provides API for functions to access EdgeX Clients, configuration and convenience methods

***params*[0]**

- EdgeX Event
- Instance of TargetType
- Data from previous function

***params*[1]**

- Content type If TargetType is set to *[]byte, otherwise it is empty

edgexfoundry.org | @edgexfoundry

Custom Pipeline functions must to adhere to the AppFunction function signature seen here.

Each function is passed, a context and a params slice.

EDGEXFOUNDRY

Writing a custom Pipeline Function (con't)

All Pipeline functions should do the following

1. Validate the input data
2. Operate on the input data
3. Return appropriate value/data
   - **Bool** return value can be
     - **True** - pipeline execution should **continue**
       - Further processing required
     - **False** - pipeline execution should **stop**
       - Error has occurred
       - No further processing required.
   - **Interface{}** return value can be
     - Error object
     - Data to be passed to the next function in the pipeline
     - Empty (nil) - no further processing required

```go
func (f Conversion) TransformToXML(
   edgexcontext *appcontext.Context,
   params ...interface{})
   (bool, interface{}) {

   if len(params) < 1 {
      return false, errors.New("no Event received")
   }

   event, ok := params[0].(models.Event)
   if !ok {
      return false, errors.New("incorrect type received")
   }

   result, err := xml.Marshal(event)
   if err != nil {
      return false, err
   }

   return true, string(result)
}
```

EDGEXFOUNDRY

App Context

Provides the following APIs for the pipeline functions

Edgex Clients
- LoggingClient
- EventClient
- ValueDescriptorClient
- CommandClient
- NotificationsClient

Helpers
- MarkAsPushed()
- PushToCore()
- Complete()
- SetRetryData()

Data
- Configuration
- EventID
- EventChecksum
- CorrelationID

edgexfoundry.org | @edgexfoundry

Here is more detail on the App Context APIs

EDGEXFOUNDRY

SDK Specific Configuration

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

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

```
[MessageBus]
Type = 'zero'
    [MessageBus.PublishHost]
        Host = '*'
        Port = 5564
        Protocol = 'tcp'
    [MessageBus.SubscribeHost]
        Host = 'edgex-core-data'
        Port = 5563
        Protocol = 'tcp'
```

```
[Writable.StoreAndForward]
Enabled = false
RetryInterval = '5m'
MaxRetryCount = 10
```

```
[ApplicationSettings]
DeviceNames = "Random-Float-Device, Random-Integer-Device"
ValueDescriptors = "RandomValue_Float32, RandomValue_Int32"
```

edgexfoundry.org | @edgexfoundry

The following are the Application Service specific configuration items.

- **[Binding]** is where you configure the input Trigger
- **[MessageBus]** is where you configure the MessageBus Trigger type and host information
- **[StoreAndForward]** is where you enable and configure the new Store and Forward capability
- **[ApplicationSetting]** is where you can add custom configuration. Note this is a simple Map of strings

The rest of the Application Service's configuration is either boilerplate EdgeX configuration or for the Configurable Pipeline.

EDGEXFOUNDRY™

Command Line options

The SDK provides the following specific command line options.

-o
-overwrite
Overwrite configuration in the Registry with local values.

-s
-skipVersionCheck
Indicates the service should skip the Core Service's version compatibility check.

EDGEXFOUNDRY

Store & Forward

The Store and Forward capability allows for export functions to persist data on failure and for the export of the data to be retried at a later time.

- The retry restarts function pipeline with function persisted the data.
- On successful retry the execution of the pipeline continues
- Retry interval and max number of retries is configurable
- Stored data is removed when:
  - Successful retry
  - Max reties exceeded

Note: The order the data exported via this retry mechanism is not guaranteed to be the same order in which the data was initial received from Core Data

EDGEXFOUNDRY

Misc Features

- Appsdk
  - *AddRoute()*
  - *ApplicationSettings()* `map[string]string`
- Helpers
  - *util.CoerceType()*
  - *util.DeleteEmptyAndTrim() & util.SplitComma()*

```
util.DeleteEmptyAndTrim(strings.FieldsFunc(deviceNames, util.SplitComma))
```

edgexfoundry.org | @edgexfoundry

AddRoute()
Allows adding custom routes to the existing webserver

ApplicationSettings()
Returns the values (map[string]string) specified in the custom **[ApplicationSettings]** configuration section.

util.CoerceType()
converts a string, []byte, or json.Marshaler type to a []byte for use and consistency in pipeline functions.

util.DeleteEmptyAndTrim() & util.SplitComma()
use custom split func instead of .Split to eliminate empty values (i.e Test,,,)

EDGEXFOUNDRY™

READMEs

- https://github.com/edgexfoundry/app-functions-sdk-go
- https://github.com/edgexfoundry/app-service-configurable

EDGEXFOUNDRY

Labs 3, 4 & 5
35+ mins

https://github.com/rsdmike/labs