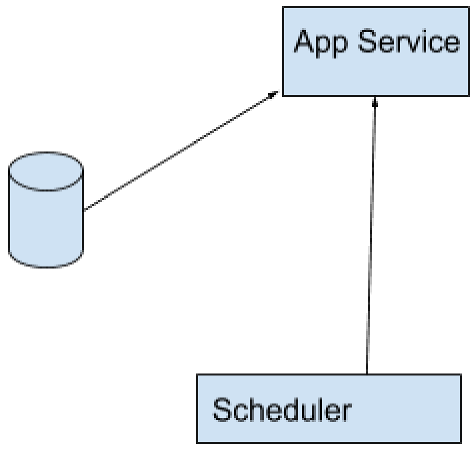
EdgeX Application Working Group 8/20/2019

Other Updates:

- TargetType is implemented to support custom types to be used between App Services (no longer requiring an EdgeX Event)
- Environment Variables -

Summary of Store and Forward:

Data will be stored upon error on export functions (HTTPPost, MQTTSend) enabled by a true/false flag "persistOnError". The following breaks out areas that would be affected that are exposed to the developer at a surface level. Internal workings are not detailed here (except the db).



Questions/Opens:

Is batch needed? - Not for now, not related to store/forward - lets leave this on the table as stretch goal.

Do we need a flush? - Push it off?

- QoS for Mqtt how does it related to MaxRetryCount - Need to determine

  How do we handle orphaned data? -- Do we care? - Do we have a TTL?

  Crawl - ignored

  Walk - Storage Service to manage data
- Opens: How is scheduler updated with URLs? Synced with consul?
- MarkAsPushed - handling for multiple app services than ingest the same event (how to know *all* successfully pushed) - same problem exists with export services
- Storage Service - discuss with TSC for Geneva?????

Assumptions:

- Data will be discarded if pipeline changes
- Data is removed after success
- Remove/ColumnName Changes to persistent store requires wipe?

New Initialization Parameters:
- RetryInterval (in minutes).
    - 0 = Do Not Retry and will remove any schedules from scheduler
    - > 0 = Register this app service with scheduler
- MaxRetryCount
    - 0 = Keep Trying Forever (only deletes upon success)
    - Threshold for when to remove the data from the db after so many retries
    - Provide traceability for when data is removed (i.e. Logging)

New Endpoint Added:
- /api/v1/RetryPipeline
    - Called by scheduler based on interval.

New Context Function:
- PersistPayload(payload []byte) - the function that will call the Create/Update dbPkg to persist the data

SDK Functions to be Affected:
- HTTPPost(persistOnError=true/false)
- MQTTSend(persistOnError=true/false)

Database Implementation (Help Wanted):
- Leverage official mongo driver: https://github.com/mongodb/mongo-go-driver (License: Apache 2.0)

DB: AppServices
CollectionName: RetryDataV1?
Columns:
- ID (uniqueId,guid) - unique identifier for this record
- AppServiceKey (string) - identifies the app service to which this data belongs
- Payload (byte[]) - the data to be exported (
- RetryCount (int) - how many times this has tried to be exported
- PipelinePosition (int) - where to pickup in the pipeline
- Version (string) - hash of the functions to know if the pipeline has changed
- CorrelationId - from EdgeX to track this individual record as it moves
- EventId/Checksum - in order to identify edgeX event from core and mark as pushed

CollectionName: SchemaVersion
Columns:
    SDKVersion: schema

DB Pkg - ideally abstracted for implementation for Redis and Mongo

----
Create() - Store()
Retrieve() - RetrieveFromStore()
Update() - UpdateRetryCount()
Delete() - RemoveFromStore()

Example:

Filter
Compress - return value of this would be persisted
HttpPost

Topics from last time:

- Store & Forward Goals:
    - When connectivity is lost
    - Support Batch Mode and sending Data on a schedule

  Proposal
    - Leverage existing reference implementations MongoDB and Redis
        - Probably best way to go to create its own connection and its own db collection
        - Can use same mongo instance or other - ensure isolated
    - Add new parameter/option to Export functions (HTTPExport, MQTTSend) to persist on error
        - Persist on error would store event data to db on failed request
        - Should we consider a timeout for data persisted for it to be aged out
    - Add new function - Batch(count int) - to hold messages until count is reached before outputting to next function
    - Provide /endpoint for scheduler to call in order to retry previously failed requests
    - Need to be clear with examples of how and when voluminous data versus occasional data can be persisted or dropped
    - When processing is picked up again, its done at the export point, not the beginning of the pipeline
    - Need identity of pipeline of that originated the data as well as where in the pipeline it was.
        - App Service Configurable - pipeline changes, what do you do with the data if the stage in the pipeline no longer exists
    - Future consideration - Fork Pipeline based on conditions
    - Example pipeline 1 (Valuable occasional data):
        - FilterByDeviceName()
        - TransformToJSON

    - Batch(50)
    - CompressWithGZIP
    - HTTPPost(persist=**true**)
    - MarkAsPushed - not called until connectivity is restored
  - Example pipeline 2 (voluminous telemetry data drop it if we fail to send it out):
    - FilterByDeviceName()
    - TransformToJSON
    - HTTPPost(persist=**false**)

- Feature Requests - Brad Corrion