Jim et al.

I have been reviewing the current scheduler we have out there in the EdgeXFoundry-Holding area. Additionally, I went ahead and reviewed our existing EdgeXFoundry area under Architecture for the overall design principal and desire for the scheduler.

The schedulers goal looked primarily to solve issues with device event storage maintenance.

Scheduler Current State:

Is a library currently and does not run as a standalone service?

This package currently resides in the Core "Supporting Services" area. Currently it runs as a service and has a client which is consumed by a dependent service

Performs Schedules and Scheduled Events.

These are two different aspects the actual schedule "When it should run an event and what eventId" and the associative scheduled event "the thing it does when *Scheduled* with a given eventId"

Scheduler package which is executed via core

It looks like the majority of what I have seen is sending off events to notifications which in turn do some MQTT queue-based work. Which in turn may send of requests to remove device event data.

Has a Scheduler Client package which services will need to consume

Services which desire to have scheduled events to occur will need to incorporate the EdgeXFoundry Scheduler Client. This client will need to be created. Once created you will need to create a *Schedule*. Once a *Schedule* has been created the client will be required to create a *Scheduled Event*.

Schedules via an HTTP webservice API

Pretty self-explanatory we use mux handlers to create, delete, and update schedules and scheduled events.

Maintains schedules in memory “no persistence is currently available”

If we require persistence we would need to implement something like that.

Supports Intervals through Frequency

Schedule Events facilitate reoccurring events to take place based on the Schedule

Username/Password support?

This could be a good thing or a bad thing. The Scheduler currently supports the use of Username and Passwords. There appears to be no security associated with this.

Requires clients “Services” to request their own *Schedules* and *Scheduled events*

With no persistence and no configuration of jobs, client need to contact the scheduler and submit their Schedules and Scheduled events. Additionally, you will need to incorporate an *Addressable* an endpoint which the scheduler can send the targeted *Scheduled Event* too.

Scheduler Proposed Future State

A standalone Service aka systemd etc.

With an eye to the future we may want to introduce a scheduler as a full service. We could end up scaling this depending on client requirements. Additionally, failover (leader, follower) support would be nice as well in the future.

Persistent Schedules and Scheduled Events

We should have the ability to have schedules and events exist through restarts and errant conditions. This would be either a configuration file or a semi/permanent data store.

Storage Orthogonality

It would be nice to support ephemeral/quasi ephemeral/permanent data stores like in memory cache, Redis, and persistent storage like BoltDB, Mongo or even Consul for Jobs.

Atomic scheduling

Currently clients are required to establish a “Schedule” and then create new “Scheduled Event” which are associated with a “Schedule”. Desirable would be sending or creating atomic actions which do both.

Ability to run Local as well as Remote Jobs

Currently the setup is for strictly http endpoints. It would be nice to have the ability to execute bash shell script etc.. As well as remote Jobs.

Implement Retries

Scheduler currently does not have the ability to fail and retry “Scheduled Events”

Implement Timeouts

Scheduler currently does not support Timeouts or any associative actions or results.

Implement Job Statistics

Scheduler currently does not implement any “real time” statistics on Jobs/scheduled events. Things like success, failed, scheduled, retries.

Implement ISO Standard timing

It would be nice to be ISO 8601 Notation compliant for date time.

I have been looking at potential alternatives to our code we have in our EdgeXFoundry-Holding area. There are several packages out there some from Java Quartz ports some from Python Cronos ports etc.

One that I am looking at is Kala. https://github.com/ajvb/kala

Kala addresses several of the “Desirable Feature” requests. We could use parts with disclaimers and proper annotations or all the features this Go package supplies. It should be noted that Kala does implement several packages. These would need to be scrutinized and approved/disapproved appropriately.

Additionally, I am comprising a document which compares some of the features I mentioned above.

Eric Cotter

Technical Staff

Dell Technologies | IoT DellTech

Eric_Cotter@Dell.com

Round Rock, TX USA