EdgeX Foundry

Device Service Scheduling for Dehli

tony espy <espy@canonical.com>

Scheduling Applied to Device Services

- A DS may create schedules & schedule events in Core Metadata
- These can also be created out-of-band (ie. by another client or service)
- Schedules objects are used to trigger readings & actuation commands via the command endpoint
- The endpoint returns CBOR/JSON Events & Readings on success
- ...and also sends the same CBOR/JSON Events & Readings to Core Data

Schedule

- A Schedule:
  - extends BaseObject (ie. they have Id, Origin, Modified, and Created fields)
  - is generic, it's not tied to specific services
  - must have a unique Name
  - has start & end times specified using ISO8601 date/time format:

    YYYYMMDD'T'HHmmss

- A Schedule defines events to be triggered:
  - just once
  - repeatedly at a given frequency (an ISO 8601 duration; eg. PT20.5S, P1D, PT10H)
  - repeatedly using a cron-style expression

ScheduleEvent

- A ScheduleEvent:
  - extends BaseObject (ie. they have Id, Origin, Modified, and Created fields)
  - is tied to a specific service via service name
  - is tied to a specific event via event name
  - must have a unique name
  - has an addressable, an object that represents a remote method (MQTT topic, REST endpoint, …)
  - has an optional parameters string that can be sent to the addressable

Support Scheduler

- At startup may create Schedule & ScheduleEvents in Core Metadata
  - ScheduleEvents *can* be created for other services: https://github.com/edgexfoundry/support-scheduler/blob/master/src/main/resources/application.properties#L54
- Queries Core Metadata for:
  - all Schedules
  - all ScheduleEvents with Service == "support-scheduler"
- Schedules events and when they're triggered:
  - iterates through all of the Schedule's ScheduleEvents and invokes the addressable for each
- Question: if SS can create ScheduleEvents for other services, but only queries for ScheduleEvents it owns, how does this work?

Java Device Service SDK

- Given a name and Device Service template, the SDK generates a skeleton Device Service by applying the given name to a base set of Java classes
- The Device Service Template can specify "SDK Scheduler Block=true" to enable an internal Device Service scheduler (vs. using Support Scheduler)
- The internal scheduler works by generating REST calls **to itself** to trigger ScheduleEvents used for readings & command actuation
- The results from the REST call are not used
- Successful REST calls trigger the same JSON/CBOR Events & Readings to be sent to Core Data via separate REST calls

Schedule a single Reading for a single device

- Create an hourly Schedule (PT1H)
- Create a ScheduleEvent
  - method=GET, path=/api/v1/device/12345678890/AccelX

    OR

  - method=GET, path=/api/v1/device/BLEDevice1/AccelX

Schedule a single Reading for all devices

- Create an hourly Schedule (PT1H)
- Create a ScheduleEvent
  - method=GET, path=/api/v1/device/all/AccelX

Schedule readings for all device resources

- Create an hourly Schedule (PT1H)
- Define a resource command in the device profile called "Status" which aggregates all device resources (eg. AccelX, AccelY, AccelZ)
- Create a ScheduleEvent
  - method=GET, path=/api/v1/device/all/Status

Schedule one Actuation Cmd for a single device

- Create an hourly Schedule (PT1H)
- Create a ScheduleEvent
  - method=PUT, path=/api/v1/device/12345678890/AccelX, parameters=...

    OR

  - method=PUT, path=/api/v1/device/BLEDevice1/AccelX, parameters=

Problems

- Support Scheduler doesn't query Core Metadata for device service ScheduleEvents
- See earlier question about Support Scheduler & core-data ScheduleEvents
- Triggering automatic readings to Core Data requires a REST call, where the results are thrown away (lots of extra work)
- If the SDKs are to provide internal scheduling, they should not make REST calls to themselves...

Do we need Support Scheduler for Device Services?

- DS implementations could implement their own scheduler based on Schedules & ScheduleEvents
- SDKs could implement a scheduler based on Schedules & ScheduleEvents (better, but non-trivial)

Device Virtual Approach

- each DeviceResource has a collectionFrequency field
- DeviceResource implemented its own Scheduler which triggered automatic calls via ScheduleEvents to its /collector endpoint
- The collector endpoint doesn't return any JSON, it just triggers the reading and pushes it to Core Data
- collectionFrequency can be changed, this results in an update to the Schedule associated with the device resource

Possible Solutions for Dehli

- Can we invent a REST-less mechanism for triggering automatic readings?
  - Instead of triggering a REST call, SDK would trigger a call to the ProtocolDriver, and push results to Core Data
- Alternatives include:
  - Extend DeviceObject to have a collectionFrequency & delay
    - wouldn't extend to actuation commands
  - Extend ScheduleEvent to include:
    - a command name & method (used to indicate actuation vs. reading)
      - mutually exclusive with addressable
      - *we decided that it might may be possible for a DS to extract command name (a parameter in the URL path), and method from the Addressable*
    - a list of device names (mutually exclusive with device-id list)
    - a list of device ids (mutually exclusive with device-name list)
    - a delay (used to add an additional delay for each device in the name or id list)