System Management WG Meeting Minutes: 08/20/19

Attendees: Joe (IBM), Michael, Akram (Dell), Rodney (Beechwoods), Lenny, Mike (Intel), Ike (ALICON SE). (Attendees who may have joined after the start of the meeting may not have been captured and listed)
Note: Discussion and action items as a result of meeting in RED

Old Business

- Open Horizons sub-project
  - State of current POC
  - Next meeting again next week, on Monday, August 26th.
- Fuji Work
- Update on Metrics collection by Executor – Akram to provide this, as well an update each on the items that follow thereafter.
  - Refactoring code for testability.
  - Challenges faced in the process, and the strategy to overcome them.
  - Walkthrough of increased testability, and implications for code coverage.
  - Current results, in brief.
    - Walked attendees through the testability
- Tasks ahead: A recap.
  - Set Configuration – planned
  - Start/stop/restart all done by the executor to include stop/restart of SMA – planned
    - Incorporate industry best practices and design patterns as applicable
  - Without restart (Fuji, or Geneva?)
  - Trevor: Security
    - JWT token not in?!
    - Geneva possibly since security aspects will be in place then.
  - Rodney: Certification process (Variability opted for, rather than rigid structures, by way of flexible JSON responses,)

New Business

- Update on refactored code: (a) refactoring for increased testability, (b) along with new unit tests to increase code coverage) ➔ Metrics collection by Executor
  - Akram to provide an update:
    - Current results, in brief.
      - Gave demo of tests in action, including test coverage numbers.
    - Walk attendees through the testability.
      - Showed how code was restructured.
    - Elements of the latest design-related discussion can be found in EdgeX Issue #1486 (Implement the design: SMA to get Metrics via Executor) @ https://github.com/edgexfoundry/edgex-go/issues/1486
      - Taking into account the variability that we've opted for, rather than a rigidly-imposed set of structures, by way of flexible responses encoded in maps of JSON structures:
        - Go SDK-based services would return a set of metrics.
        - C SDK-based services would return a set of metrics.
        - When leaning on the executor, another set of metrics would be returned.

            - What unifies these variable responses is the flexibility afforded by key-value pairs (conveniently encoded in maps of JSON structures)
                - Rodney: We can look to how *device-services* addressed this, by tapping into the attached meta-data. For example, given a key, **query**, say, core-metadata, for relevant data. Would this lead to breakage perhaps? How about expansion?
                - Michael: Consider basic **meta-data-like** packets. Get raw results from underlying implementation—Provide, in turn, high-level, normalized version. Client gets basics, and remain unaware of details. But, should they wish, clients can get to passthrough via introspection. We provide, thereby, a simple **contract**, all the while providing (an optional) passthrough to get deeper details, should they wish.
                - Rodney: Sounds good. This would be certifiable, pending conformance.
    - So yes, the metrics responses being dependent on the chosen metrics executor is very much by design.
        - The metrics response looks likes in terms of responses is dependent on the chosen metrics executor. The concern raised is that, for example if I'm writing a client of the SMA and want to get known CPU usage I need to look at the CpuBusyAvg key with a direct-service executor, and the cpu_perc key for an executor. How would a client introspect from the SMA which executor is being used to generate metrics?
                - Akram: Illustrated the elements we're working with (raw data, along with the various JSON responses that it's unmarshalled and packaged into for reporting to clients (Also, how we can switch between different underlying metrics-fetching implementations).
                - Michael: Speaking of implementation, the Docker executor would provide a unique *type* (a field, of acceptable value), to go along with a unique implementation—Note: For *device-services*, we ran into unique-port-number concerns and considerations… i.e. How do we track these, for disambiguation, so that there's no overlap? Perhaps assign *type* to *implementation*. Complement by a testing regimen to validate the correspondence with BLOBs (to stave off breakage concerns).
                - Rodney: Registry-like solutions perhaps? What we see is great progress.
- Being worked:

        - Set Configuration.
            - Akram to report progress at our next WG call.
- Tasks ahead:
    - Start/stop/restart all done by the executor to include stop/restart of SMA – planned
        - Incorporate industry best practices and design patterns as applicable
            - Executor will, effectively, be made the watcher of (what *used* to be) the watcher (i.e. the SMA) as a result of the refactor coming up for Fuji.