Information Model for Large-Scale Measurement Platforms (LMAP)

draft-ietf-lmap-information-model-07

Jürgen Schönwälder, Jacobs University

Changes since IETF 93 (1/2)

- The default execution mode is pipelined (LI12)
- Added text to define which action consumes data in sequential, pipelines, and parallel execution mode (LI11)
- Added ma-config-measurement-point, ma-report-measurement-point, and ma-config-report-measurement-point to configure and report the measurement point (LI10)
- Turned ma-suppression-obj into a list that uses a start event and a stop event to define the start and end of suppression; this unifies the handling of suppression and loss of controller connectivity (LI09)
- Added ma-controller-lost-obj and ma-controller-ok-obj event objects (LI09)
- Added ma-status-schedule-obj to report the status of a schedule and refactored ma-task-status-obj into ma-status-action-obj to report the status of an action (LI07, LI08)

Changes since IETF 93 (2/2)

- Introduced a common ma-metric-registry-obj that identifies a metric and a set of associated roles and added this object to expose metric capabilities and to support the configuration of metrics and to report the metrics used (LI06)
- Introduced ma-capability-obj and ma-capability-task-obj to expose the capabilities of a measurement agent (LI05)
- Use 'ordered list' or 'unordered set' instead of list, collection, etc. (LI02)
- Clarification that Actions are part of a Schedule (LI03)
- Deleted terms that are not strictly needed (LI04)

LI14: Remove ma-preconfig-obj

- Since `ma-preconfig-obj` is really just a subset of `ma-config-obj`, do not define a data structure for it but simply identify the pre-configuration subset in a textual description. The YANG data model for example does not model this as separate objects (since NETCONF does not really distinguish between pre-configuration and configuration of a device).
- On the other hand, it does not hurt to keep it as long as data models are allowed to shortcut this.

LI16: Result reporting for tasks that implement multiple metrics

- The current information model structure makes it problematic to report a set of metrics measured by a single measurement task since the structure is not flexible enough (a `ma-report-task-obj` can only report one metric).

LI17: Result reporting efficiency

- Result reports are rather heavy weight; while it is useful to have a proper record of the parameters and options used to produce a result record, repeating all that information in _each_ result record makes them rather verbose. Can we instead rely on the state information to provide the necessary information? Or can we assume that the controller shares information with the data analysis component? In some deployed systems, all you have in the result record is a tag and all other information (column structure) is inferred from this tag and key parameters are echoed as part of the result vector. This seems way simpler and more efficient.

LI16 and LI17 Examples (1/2)

```
$ mtr -C -n www.google.com
MTR.0.86;1446370302;OK;www.google.com;1;2001:638:709:5::1;315
MTR.0.86;1446370302;OK;www.google.com;2;???;0
MTR.0.86;1446370302;OK;www.google.com;3;???;0
MTR.0.86;1446370302;OK;www.google.com;4;???;0
MTR.0.86;1446370302;OK;www.google.com;5;2001:7f8::3b41:0:1;8731
MTR.0.86;1446370302;OK;www.google.com;6;2001:4860::1:0:abf6;8429
MTR.0.86;1446370302;OK;www.google.com;7;2001:4860::8:0:abf2;19922
MTR.0.86;1446370302;OK;www.google.com;8;2001:4860::8:0:8f8f;19170
MTR.0.86;1446370302;OK;www.google.com;9;2001:4860::8:0:6400;19069
MTR.0.86;1446370302;OK;www.google.com;10;2001:4860::1:0:6e0f;19568
MTR.0.86;1446370302;OK;www.google.com;11;2001:4860:0:1::4b;19666
MTR.0.86;1446370302;OK;www.google.com;12;2a00:1450:4008:800::1014;18813
```

- CSV format with a convention for a common beginning
  - 1st column: tag + version (or the result format)
  - 2nd column: time (seconds since 1970-01-01T00:00:00)
  - 3rd column: status (OK, FAIL, …)
  - 4th column: target
  - 5th column: hop count
  - 6th column: IP address
  - Additional columns report measured rtt values

LI16 and LI17 Examples (2/2)

```
$ happy -c -a www.google.com -m
DNS.0.4;1446369960;OK;www.google.com;2a00:1450:4008:800::1013;www.google.com;ber01s08-in-x13.1e100.net
DNS.0.4;1446369960;OK;www.google.com;173.194.32.209;www.google.com;ber01s08-in-f17.1e100.net
DNS.0.4;1446369960;OK;www.google.com;173.194.32.210;www.google.com;ber01s08-in-f18.1e100.net
DNS.0.4;1446369960;OK;www.google.com;173.194.32.211;www.google.com;ber01s08-in-f19.1e100.net
DNS.0.4;1446369960;OK;www.google.com;173.194.32.212;www.google.com;ber01s08-in-f20.1e100.net
DNS.0.4;1446369960;OK;www.google.com;173.194.32.208;www.google.com;ber01s08-in-f16.1e100.net
HAPPY.0.4;1446369960;OK;www.google.com;80;2a00:1450:4008:800::1013;18829;18565;18935
HAPPY.0.4;1446369960;OK;www.google.com;80;173.194.32.209;11790;11878;11817
HAPPY.0.4;1446369960;OK;www.google.com;80;173.194.32.210;11714;11488;11484
HAPPY.0.4;1446369960;OK;www.google.com;80;173.194.32.211;11638;11488;11456
HAPPY.0.4;1446369960;OK;www.google.com;80;173.194.32.212;11526;11995;11848
HAPPY.0.4;1446369960;OK;www.google.com;80;173.194.32.208;11500;11400;11363
```

- CSV format with a convention for a common beginning
  - 1st column: tag + version (or the result format)
  - 2nd column: time (seconds since 1970-01-01T00:00)
  - 3rd column: status (OK, FAIL, …)
  - 4th column: target
  - Additional target and result columns follow

Observations (1/2)

- A task can produce results for multiple metrics
  - Current model assumes only one metric
- All data reported in one format
  - Current model splits out some (but not all (!)) common values like start/end and leaves the rest to the metrics
- Results are shipped without capturing configuration parameters (it is assumed that the data analysis tools have access to the information how measurements were configured, which task implementations were used, …)

Observations (2/2)

- What do platforms use today?
  - One platform uses a CSV format as shown on previous slides
  - CAIDA's tool scamper uses a binary format (warts format) to store results and they have a tool to turn that into some JSON format
  - I am not sure which format RIPE Atlas uses to send data to a collector; the data you can retrieve from their measurement data store is structured JSON

Proposal

- Split the current `ma-report-obj` which includes a set of `ma-report-task-obj` which in turn include a list of `ma-report-row-obj` into
  - a structure to report metadata (submitted infrequently and when relevant things change) and
  - a structure to report values (optimized for efficiency)

Next Steps

- Update the I-D once we have figured out a result format that is flexible and efficient
- Update the YANG data model and validate examples against it
- A proof of concept example using the IPPM registry would be welcome as well (where should it go?)