Thing-to-Thing Research Group (T2TRG)
Internet-Draft
Intended status: Informational
Expires: 6 December 2026

L. Toutain
IMT Atlantique

4 June 2026

CORECONF for Machine-to-Machine Communication
draft-toutain-t2trg-coreconf-m2m-00

Abstract

The document addresses the specific challenges of M2M interactions where both endpoints may be constrained nodes, and explores the use of CORECONF primitives.

This document describes the use of CORECONF (CoAP Management Interface) for Machine-to-Machine (M2M) communication in constrained IoT environments. It defines a YANG data model enabling remote management and configuration of constrained devices using CoAP, CBOR, and YANG SID identifiers. The serialization in CBOR of this data model limits the payload size.

Table of Contents

1. Introduction

This document proposes a YANG data model designed for constrained devices and low-power networks. By combining YANG's strong typing with CBOR's compact binary serialization and CoAP's lightweight transport, the model enables efficient Machine-to-Machine (M2M) data exchange while remaining within the bandwidth and energy budgets of constrained environments. SCHC header compression [RFC8724] may further be used to reduce the overhead of IPv6, UDP, and CoAP headers

on the most constrained links.

Some data models and protocols already exist for M2M communication in IoT environments, but none is fully adapted to the constraints of low-power networks in terms of message size, energy consumption, and interaction patterns. This section reviews the main existing approaches and explains why a new model is needed.

SenML [RFC8428] has become a widely adopted format for Machine-to-Machine (M2M) data exchange in IoT environments, enabling constrained devices to report sensor measurements and time series in JSON or CBOR. However, SenML is primarily a data serialization format: it structures payloads but does not enforce strong type checking, schema validation, or support for configuration and remote operations.

SenML is also part of the LwM2M framework [OMA-LwM2M], which defines a broader device management protocol built on CoAP and SenML for operator-to-device interactions. However, LwM2M relies on periodic reporting and registration messages that impose a non-trivial overhead, particularly on Low-Power Wide-Area Networks (LPWANs) [RFC8376] where bandwidth and energy budgets are severely constrained.

In some ways, SenML may be described by a YANG Data Model [I-D.gudi-t2trg-senml-as-coreconf], but the integration in the YANG ecosystem remains limited.

The CORECONF protocol stack using YANG [RFC7950] for Data Modeling, CoAP [RFC7252] for data transport, and CBOR [RFC8949] and YANG SID identifiers [I-D.ietf-core-sid] for the compact data serialization provides the richer foundation that SenML lacks: a strongly typed data model, schema validation, and support for full CRUD operations and actions. However, CORECONF has so far been designed for operator-to-device management, leaving peer-to-peer M2M communication — where both endpoints may themselves be constrained nodes — largely unaddressed.

Some YANG Data Models have been defined for telemetry. [RFC9232] introduces Network Telemetry used to collect vast amounts of data to supervise a network. [RFC8639] allows subscribing to a datastore filtered through XPath and receiving notifications. [I-D.birkholz-yang-core-telemetry] proposes to extend telemetry to CORECONF, but using a traditional approach.

This document adopts a different approach. The goal is to define a YANG Data Model that will benefit from CBOR serialization to optimize the bandwidth to extend CORECONF for M2M use cases over low-power links. This document focuses on transducer management: resource

discovery, value polling, statistical computation, threshold alerts, and time-series history notifications. This is an early-stage work; future revisions will explore other categories of measurements and interaction patterns.

1.1. Use Cases

The targeted use cases are remote sensors installed in the field and connected with LPWAN or Satellite connectivity. SCHC [RFC8724] [RFC8824] is used to compress headers such as IPv6, UDP and CoAP for CORECONF traffic. The payload results from the serialization of the datastore in CBOR.

The model covers the following use cases:

* resource discovery: sensors or actuators, regrouped under the name transducers, are discovered with their characteristics (units, precision)

* simple query: each transducer can be individually queried or set.

* statistical computation: the sensor can compute some statistical values, such as mean, variance, min and max. They can be reset.

* alert notification: when a value reaches a threshold (minimum and maximum) a notification message is sent

* time series: values are collected by the device and sent when a limit is reached (number of samples, duration, message size)

1.2. Data Representation

CBOR is designed to be concise to represent numerical information since it is directly coded in binary and not represented in ASCII. CBOR also uses binary representation to encode structures such as Maps and Arrays. The length of a numerical value depends on its value; for instance, numbers between -24 and 23 are coded on a single byte, values between -255 and 255 on two bytes,…

Nevertheless, some representations may be less efficient numerically or less precise. CBOR defines 3 IEEE 754 encodings on 3, 5, or 9 bytes. The smallest representation introduces a close to 1% error. CBOR also provides a decimal fraction type (tag 4) encoding a value as a [exponent, mantissa] pair, which avoids floating-point rounding. However, this representation requires the exponent (the divider) to be repeated alongside every individual quantity, adding overhead for each encoded value.

The assumption leading to this YANG module is to avoid floating-point numbers for their size or precision and rely on integers with a precision parameter indicating, if positive, the number of digits after the decimal point, or the power of 10 if negative.

The module also introduces the notion of time series to record several measurements during a period of time and send them in a single message using a notification. Time series values may further be compressed depending on the nature of the data. This version proposes a compression based on delta encoding: instead of transmitting absolute values, each sample is encoded as the difference from the previous one, which significantly reduces the CBOR payload size for slowly-varying measurements.

The device can also send an alert when a measured value reaches a threshold, allowing the receiver to react promptly without waiting for a scheduled report.

1.3. Requirements Language

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

The following terminology is used in this document:

CORECONF: CoAP Management Interface, as defined in [I-D.ietf-core-comi].

M2M: Machine-to-Machine communication, referring to direct data exchanges between devices without human intervention.

SID: YANG Schema Item iDentifier, a compact numeric identifier for YANG data nodes, as defined in [I-D.ietf-core-sid].

Constrained Device: A device with limited processing, memory, and energy resources, as characterized in [RFC7228].

Device: A piece of equipment containing one or more transducers.

Transducer: an interface between the analog and digital world. Transducers are sensors reporting values or actuators having an action on the physical world.

Quantity: values manipulated by transducers.

Value: other information stored in the datastore including quantities

3. The coreconf-m2m YANG Module

The device acts as a CoAP server implementing CORECONF and exposes its state through a set of Transducers. A Transducer represents either a sensor (providing measurements) or an actuator (accepting commands). Each Transducer type is identified by a YANG identity. When multiple Transducers of the same type are present on the device, they are distinguished by an instance identifier.

3.1. Module Overview

For simplicity reasons, we regroup in a single YANG module two elements. The first element is a set of Transducer identities, which in a real deployment would typically reside in a separate device-specific module. In this example, 12 transducers are defined, corresponding to the weather station ATMOS41 (see https://metergroup.com/fr/products/atmos-41/).

The second element defines the data structures:

* a container including a device description and a list of associated transducers. Each transducer is identified by an identityref and an instance number, allowing several transducers of the same type. Each transducer contains:
  - a quantity measured from the field or set by the client.
  - a list of statistics (mean, variance, min, max), resettable via RPC.
  - alert parameters: threshold values (minimum and maximum) triggering a notification when the measured quantity crosses them.
  - time series parameters: collection settings controlling when a series of samples is sent (number of samples, duration, message size).
* an RPC, now limited to resetting all statistics
* two notification types: one to collect time-series history, and one to alert when a quantity reaches a minimum or maximum threshold.

Figure 1 gives an overview of the module YANG tree:

```
module: coreconf-m2m
  +--ro state
  |  +--ro uptime?   uint64
  +--rw characteristics
  |  +--rw name?         string
  |  +--rw version?      string
  |  +--rw identifier?   string
  +--rw transducers
     +--rw transducer* [type id]
        +--rw type                        identityref
        +--rw id                          uint8
        +--rw unit?                       string
        +--rw nature?                     transducer-nature
        +--rw precision?                  int8
        +--ro quantity
        |  +--ro value?              int64
        |  +--ro timestamp?          uint64
        |  +--ro u-timestamp?        uint32
        |  +--ro timestamp-source?   enumeration
        |  +--ro statistics
        |     +--ro min?            int64
        |     +--ro max?            int64
        |     +--ro mean?           int64
        |     +--ro median?         int64
        |     +--ro stdev?          uint64
        |     +--ro sample-count?   uint64
        +--rw notification-parameters
        |  +--rw history
        |  |  +--ro active?        boolean
        |  |  +--rw step?          uint32
        |  |  +--rw precision?     uint8
        |  |  +--rw max-samples?   uint32
        |  |  +--rw time-period?   uint32
        |  |  +--rw encoding?      encoding-type
        |  |  +--rw max-payload?   uint32
        |  +--rw sensor-alert
        |     +--ro active?       boolean
        |     +--rw t-min?        int32
        |     +--rw t-max?        int32
        |     +--rw hysteresis?   uint8
        |     +--rw dampening?    uint32
        +---x reset-stats

  rpcs:
    +---x reset-stats
```

```
          notifications:
            +---n history
            |  +--ro last?          boolean
            |  +--ro time-series* [type id]
            |     +--ro type        identityref
            |     +--ro id          uint8
            |     +--ro values*     int64
            |     +--ro internal
            |        +--ro last-update?     uint64
            |        +--ro start-time?      uint64
            |        +--ro messages-sent?   uint64
            +---n sensor-alert
               +--ro target* [type id]
                  +--ro type     identityref
                  +--ro id       uint8
                  +--ro value?   int64
```

Figure 1: coreconf-m2m module tree

3.2. State and Characteristics Sub-Tree

These two sub-trees contain global parameters generic to the device, such as uptime. They are still under development and their content is not yet fully defined.

3.3. transducer sub-tree

Transducers sub-tree contains the list of transducers (i.e. sensors, actuators) maintained by the device. A transducer is identified by two elements:

* a “type” identityref giving the nature of the transducer,

* an “id” giving the instance number allowing several transducers of the same type.

At this level, three other leaves are defined:

* “unit” is a string giving the nature of the quantity. This value may be taken from SenML maintained by IANA. The use of a string, instead of identityref is intentional. Units are short names. YANG Identity may be larger and less flexible.

* “nature” reveals if a transducer is a sensor, from which the quantity can be read, or an actuator where the quantity can be written.

* “precision” : is the number of digits after the decimal point. Quantity values are integers multiplied by 10^precision. It can be noted that precision can also be negative for very large values.

These five leaves form the first level of the transducers tree. The recommended approach is to query with depth=0, which returns only the transducer list with its identifying leaves (type, id, unit, nature, precision). This lightweight exchange is sufficient for resource discovery and minimizes traffic on constrained networks.

A query with depth=1 is also possible and returns transducers together with their associated quantity values in a single message. However, the response may be significantly larger, as it includes timestamp and statistics data that the client may not need. On low-power links, it is therefore preferable to perform discovery and value retrieval as two separate queries.

3.3.1. quantity

The “quantity” sub-tree pushes the leaves to a deeper level, to avoid being retrieved during the resource discovery phase. Quantity contains:

* the value adjusted with the precision to be an integer,

* the timestamp in seconds and microseconds,

* the entity in charge of the timestamp, which can be the device itself or the receiver.

Under “quantity” , the sub-level “statistics” includes major statistic for a specific transducer. Locally computed statistics. Statistics may be erased for each transducer by calling the “reset-stats” action, or globally for all transducers with the “reset-stats” RPC.

3.3.2. Notification Parameters

The model supports two kinds of notifications:

* “sensor-alert” will send a notification when the measured quantity reaches one or two limits, minimal and maximal, or goes back to a value between these two bounds. To avoid fluctuations, two mechanisms are in place:

  - “hysteresis” defines a percentage, by default 5% around the limit, so if a maximum limit is set to 100, an alert message will be triggered when the quantity is higher than 105 and

another alert will be sent when the quantity becomes lower than 95%. The value is sent in the notification message, so the client is able to know the state of the alert.

- “dampening” limits the number of messages sent.

* “history” builds time series:

- “step” parameter defines at which interval samples are taken.

- “precision” allows overriding the quantity precision defined in the transducer. By default the precision is the one associated with the transducer.

- “encoding” indicates how information is stored in the time series:

  o “direct” : all the values are stored with the precision.

  o “delta” : the first value is the reference and the following ones are the difference with the previous. This allows limiting the size of the message since CBOR encodes small numbers more efficiently.

- A notification is sent when a number of measurements is reached, either:

  o the number of samples in the time series reaches “max-samples” ,

  o “max-payload” is based on the size of the time series. Since small numbers take less space than large numbers in CBOR, a time series may contain a different number of samples.

  o the “time-period” after which the collection should be sent.

The read-only active flag in both notification types is set when one or more clients observe a notification. Parameters are common to all observations of a particular transducer.

3.4. Starting Notifications

A client wishing to initiate a notification MAY first send an iPATCH to set up notification parameters. Parameters set during an active notification are immedialty applied.

Notification is started by sending a FETCH+Observe request on the notification stream resource (/s), with a body identifying the SID of the transducer to observe. Multiple clients may observe the same transducer simultaneously; each receives an independent copy of every notification.

4. CORECONF Overview in the M2M Context

4.1. CoAP Methods Mapping

CORECONF defines mappings for all CoAP methods, but this document uses only two:

* FETCH is used instead of GET to retrieve values from the YANG Data Model. Unlike GET, FETCH carries a body specifying the exact SIDs to retrieve, enabling precise and bandwidth-efficient queries. Combined with the CoAP Observe option, FETCH also serves to subscribe to notification streams.

* iPATCH is used instead of PUT or POST to modify quantities and notification parameters. It supports partial updates: only the specified nodes are modified, leaving others unchanged. Setting a node to an empty value with iPATCH is the preferred way to clear a parameter, making DELETE unnecessary for datastore modifications.

The recommended CoAP Content-Formats for all exchanges are:

* Content-Format 141 (application/yang-fetch+cbor) for FETCH request bodies, which carry the list of SIDs to retrieve.

* Content-Format 142 (application/yang-data+cbor;id=sid) for all response bodies and iPATCH payloads, where data nodes are identified by their SID.

Using these two content formats ensures maximum interoperability with CORECONF implementations and keeps the payloads as compact as possible. Limiting exchanges to a small number of well-known packet formats also benefits SCHC compression [RFC8724]: the fewer distinct header patterns in use, the more efficiently SCHC rules can compress the CoAP headers, reducing overhead on the most constrained links.

FETCH and iPATCH requests MUST be sent as Non-Confirmable (NON) CoAP messages. This leaves the application free to implement its own retransmission strategy and timer management, which is essential on constrained networks where the default CoAP confirmable retransmission behavior may be inappropriate or wasteful.

For notification streams (Observe), Confirmable (CON) messages MAY be used. A CON notification allows the server to detect that the observer is no longer reachable when no ACK is received, and to cancel the Observe subscription accordingly. A RST response from the client is also a valid way to signal that the subscription should be terminated.

5. CORECONF Traffic

The following examples show some CoAP messages between a client and a device (the CoAP server). In the example, the device is an ATMOS41 weather station, able to measure 12 parameters. An identity is associated with each parameter and a unique SID, as illustrated in Figure 2:

```
identity solar-radiation {
  base transducer-type;
  description "Solar radiation measurement (W/m2).";
}

identity precipitation {
  base transducer-type;
  description "Precipitation measurement (mm).";
}

identity air-temperature {
  base transducer-type;
  description "Air temperature measurement (°C).";
}
```

Figure 2: Excerpt of YANG identity definitions for the ATMOS41 transducers (see Appendix for the complete module)

5.1. Resource Discovery

The client does not know the transducers managed by the device. It sends a FETCH on “/transducers/transducer” with a depth of 0, as shown in Figure 3.

```
CoAP Request:
  Non-Confirmable, FETCH, MID:47709
    Token: 8ed4
    Opt #1: Uri-Path: c
    Opt #2: Content-Format: 141 (application/yang-fetch+cbor)
    Opt #3: Uri-Query: d=0
    Opt #4: Accept: 142 (application/yang-data+cbor;id=sid)

  Payload: 5 bytes
    1A 000186DF  # unsigned(100063) : /transducers/transducer

CoAP Response:
  Non-Confirmable, 2.05 Content, MID:39441
    Token: 8ed4
    Opt #1: Content-Format: 142 (application/yang-data+cbor;id=sid)

  Payload: 220 bytes
    {100063:
      [{33: 100008, 1: 0, 17: 1, 34: "W/m2"},
       {33: 100006, 1: 0, 17: 3, 34: "mm"},
       {33: 100009, 1: 0, 17: 0, 34: ""},
       {33: 100002, 1: 0, 17: 1, 34: "km"},
       {33: 100013, 1: 0, 17: 1, 34: "deg"},
       {33: 100015, 1: 0, 17: 2, 34: "m/s"},
       {33: 100014, 1: 0, 17: 2, 34: "m/s"},
       {33: 100010, 1: 0, 17: 1, 34: "deg"},
       {33: 100001, 1: 0, 17: 1, 34: "degC"},
       {33: 100012, 1: 0, 17: 3, 34: "kPa"},
       {33: 100003, 1: 0, 17: 2, 34: "kPa"},
       {33: 100007, 1: 0, 17: 1, 34: "%RH"}]}
```

Figure 3: Resource discovery: FETCH request and response

In the request, the payload 1A 000186DF is the CBOR encoding of the unsigned integer 100063, which is the SID of “/transducers/transducer”. In the response, the outer key 100063 identifies the list, and each entry is a CBOR map whose keys are delta SIDs relative to the list SID, as defined in [RFC9254]. The values encode the transducer type (as an identity SID), instance id, precision, and unit string.

The client may translate the identityref values to the names defined in the YANG module. Figure 4 shows the resulting transducer table:

```
 1  solar-radiation      W/m2  [type='solar-radiation'][id='0']
 2  precipitation        mm    [type='precipitation'][id='0']
 3  strike-count               [type='strike-count'][id='0']
 4  average-distance     km    [type='average-distance'][id='0']
 5  wind-direction       deg   [type='wind-direction'][id='0']
 6  wind-speed           m/s   [type='wind-speed'][id='0']
 7  wind-gust            m/s   [type='wind-gust'][id='0']
 8  tilt                 deg   [type='tilt'][id='0']
 9  air-temperature      degC  [type='air-temperature'][id='0']
10  vapor-pressure       kPa   [type='vapor-pressure'][id='0']
11  barometric-pressure  kPa   [type='barometric-pressure'][id='0']
12  relative-humidity    %RH   [type='relative-humidity'][id='0']
```

Figure 4: Decoded transducer list from resource discovery response

5.2. Querying a quantity

A specific quantity may be requested through a FETCH. Figure 5 shows the exchange for the current value of air-temperature.

```
CoAP Request:
  Non-Confirmable, FETCH, MID:47710
    Token: 8ed5
    Opt #1: Uri-Path: c
    Opt #2: Content-Format: 141 (application/yang-fetch+cbor)
    Opt #3: Accept: 142 (application/yang-data+cbor;id=sid)

  Payload: 12 bytes
    [100092, 100001, 0]

CoAP Response:
  Non-Confirmable, 2.05 Content, MID:39442
    Token: 8ed5
    Opt #1: Content-Format: 142 (application/yang-data+cbor;id=sid)

  Payload: 8 bytes
    {100092: 112}
```

Figure 5: FETCH request and response for air-temperature current value

The FETCH body [100092, 100001, 0] is a CORECONF instance-identifier: the first element is the SID of the requested leaf, followed by the list key values identifying the transducer (type=100001 for air-temperature, id=0). The response value 112, combined with precision=1, decodes to 11.2°C.

Figure 6 shows the FETCH for the full statistics table of the same transducer:

```
CoAP Request:
  Non-Confirmable, FETCH, MID:47711
    Token: 8ed6
    Opt #1: Uri-Path: c
    Opt #2: Content-Format: 141 (application/yang-fetch+cbor)
    Opt #3: Accept: 142 (application/yang-data+cbor;id=sid)

  Payload: 12 bytes
    [100082, 100001, 0]

CoAP Response:
  Non-Confirmable, 2.05 Content, MID:39443
    Token: 8ed6
    Opt #1: Content-Format: 142 (application/yang-data+cbor;id=sid)

  Payload: 24 bytes
    {100082: {4: 79, 1: 119, 2: 94, 3: 103, 6: 12, 5: 53}}
```

Figure 6: FETCH request and response for air-temperature statistics

The FETCH body [100082, 100001, 0] requests the quantity sub-tree (SID 100082) for air-temperature (100001), instance 0. In the response, the inner map keys are delta SIDs relative to 100082, each encoding the difference between consecutive SIDs to minimize CBOR size. The six values correspond to the statistics leaves: min, max, mean, median, stdev, and sample-count, all scaled by the transducer precision.

The client decodes the response and displays the statistics as shown in Figure 7:

```
  [9] Statistics — air-temperature:
    min:     7.9 degC
    max:     11.9 degC
    mean:    9.4 degC
    median:  10.3 degC
    σ :       1.2 degC
    n:       53
```

Figure 7: Decoded statistics for air-temperature

5.3. Notification

The client first sends an iPATCH to configure the history notification parameters for the solar-radiation transducer, as shown in Figure 8.

```
CoAP Request:
  Non-Confirmable, iPATCH, MID:47712
    Token: 8ed7
    Opt #1: Uri-Path: c
    Opt #2: Content-Format: 142 (application/yang-data+cbor;id=sid)

  Payload: 28 bytes
    {[100066, 100008, 0]: {100066: {6: 5000, 4: 3, 2: 1}}}

CoAP Response:
  Non-Confirmable, 2.04 Changed, MID:39444
    Token: 8ed7
```

Figure 8: iPATCH to configure history notification parameters for solar- radiation

The iPATCH key [100066, 100008, 0] is an instance-identifier targeting the notification-parameters node (SID 100066) for solar-radiation (SID 100008), instance 0. The value {100066: {6: 5000, 4: 3, 2: 1}} sets three history parameters using delta SIDs relative to 100066: step=5000 ms (one sample every 5 seconds), max-samples=3, and encoding=delta (value 1).

The client then initiates an Observe subscription with a FETCH on the notification stream resource /s, as shown in Figure 9.

```
CoAP Request:
  Non-Confirmable, FETCH, MID:47713
    Token: 8ed8
    Opt #1: Observe: 0
    Opt #2: Uri-Path: s
    Opt #3: Content-Format: 141 (application/yang-fetch+cbor)
    Opt #4: Accept: 142 (application/yang-data+cbor;id=sid)

  Payload: 12 bytes
    [100044, 100008, 0]

CoAP Response (subscription acknowledgment):
  Non-Confirmable, 2.05 Content, MID:39445
    Token: 8ed8
    Opt #1: Observe: 0
    Opt #2: Content-Format: 142 (application/yang-data+cbor;id=sid)

  Payload: 1 byte
    {}

CoAP Notification:
  Non-Confirmable, 2.05 Content, MID:39446
    Token: 8ed8
    Opt #1: Observe: 1
    Opt #2: Content-Format: 142 (application/yang-data+cbor;id=sid)

  Payload: 43 bytes
    {100042: {2: [{6: 100008, 1: 0, 7: [1043, 82, 82, 362, 316, 318, 226, 94, -147, -16]}
]}}
```

Figure 9: FETCH+Observe subscription on ‘/s‘ and history notification for solar-radiation

The FETCH body [100044, 100008, 0] subscribes to the history time-series stream (SID 100044) for solar-radiation. The server acknowledges with an empty map {}. In the notification, the outer key 100042 identifies the history notification; the inner structure uses delta SIDs to encode type, id, and the values list. The values [1043, 82, 82, ...] use delta encoding: 1043 is the absolute reference (104.3 W/m), and each subsequent value is the difference from the previous one, yielding a compact representation of slowly-varying measurements.

The client decodes the delta-encoded time-series values and displays the result as shown in Figure 10:

```
[1] solar-radiation: 104.3 W/m2  (16:50:29)
[1] solar-radiation: 112.5 W/m2  (16:52:29)
[1] solar-radiation: 120.7 W/m2  (16:54:29)
[1] solar-radiation: 156.9 W/m2  (16:56:29)
[1] solar-radiation: 188.5 W/m2  (16:58:29)
[1] solar-radiation: 220.3 W/m2  (17:00:29)
[1] solar-radiation: 242.9 W/m2  (17:02:29)
[1] solar-radiation: 252.3 W/m2  (17:04:29)
[1] solar-radiation: 237.6 W/m2  (17:06:29)
[1] solar-radiation: 236.0 W/m2  (17:08:29)
```

Figure 10: Decoded solar-radiation time-series from history notification

6. SID Allocation

The SID range 100000100399 is used as an example throughout this document. Official values MUST be assigned by IANA prior to publication. The complete SID file is provided in Figure 12.

7. Security Considerations

CORECONF operations over CoAP MUST be secured using either DTLS [RFC6347] or OSCORE [RFC8613]. In M2M scenarios where a central manager is absent, the trust model requires particular attention.

8. IANA Considerations

8.1. YANG Module Registration

This document registers the following YANG module in the "YANG Module Names" registry [RFC7950]:

Name	Namespace	Prefix	Reference
coreconf-m2m	urn:ietf:params:xml:ns:yang:coreconf-m2m	cm2m	This document

Table 1

8.2. SID Range Allocation

This document requests the SID range starting at entry-point TBD with a size of 100, as defined in [I-D.ietf-core-sid].

Transducer identites MUST be sperated from this document and defined in anjother YANG module.

9. References

9.1. Normative References

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

[RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <https://www.rfc-editor.org/info/rfc8824>.

[RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/info/rfc9254>.

[RFC9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <https://www.rfc-editor.org/info/rfc9363>.

[I-D.ietf-core-comi] Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and C. Bormann, "CoAP Management Interface (CORECONF)",

Work in Progress, Internet-Draft, draft-ietf-core-comi-21, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-21>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <https://www.rfc-editor.org/info/rfc8376>.

[RFC9179] Hopps, C., "A YANG Grouping for Geographic Locations", RFC 9179, DOI 10.17487/RFC9179, February 2022, <https://www.rfc-editor.org/info/rfc9179>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC7396] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396, DOI 10.17487/RFC7396, October 2014, <https://www.rfc-editor.org/info/rfc7396>.

[RFC8428] Jennings, C., Shelby, Z., Arkko, J., Keranen, A., and C. Bormann, "Sensor Measurement Lists (SenML)", RFC 8428, DOI 10.17487/RFC8428, August 2018, <https://www.rfc-editor.org/info/rfc8428>.

[RFC9232] Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", RFC 9232, DOI 10.17487/RFC9232, May 2022, <https://www.rfc-editor.org/info/rfc9232>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[I-D.ietf-core-sid]
Veillette, M., Pelov, A., Petrov, I., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-24, 22 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-sid-24>.

[I-D.ietf-core-yang-cbor]
Veillette, M., Petrov, I., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-20, 11 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-yang-cbor-20>.

[I-D.gudi-t2trg-senml-as-coreconf]
G, M., Toutain, L., Fernandez, A., and J. BONNIN, "SenML is CORECONF (almost)", Work in Progress, Internet-Draft, draft-gudi-t2trg-senml-as-coreconf-00, 7 October 2024, <https://datatracker.ietf.org/doc/html/draft-gudi-t2trg-senml-as-coreconf-00>.

[I-D.birkholz-yang-core-telemetry]
Birkholz, H. and E. Voit, "Concise YANG Telemetry", Work in Progress, Internet-Draft, draft-birkholz-yang-core-telemetry-01, 16 July 2018, <https://datatracker.ietf.org/doc/html/draft-birkholz-yang-core-telemetry-01>.

[OMA-LwM2M]
"Lightweight Machine to Machine Technical Specification: Core", 2020, <https://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

Appendix A. Complete YANG Module

```
module coreconf-m2m {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:coreconf-m2m";
  prefix m2m;

  import ietf-geo-location {
    prefix geo;
    reference "RFC 9179";
  }

  organization "METER Group, Inc.";
  contact
    "Technical Support
     https://www.metergroup.com/";

  description
    "YANG data model for a generic M2M CoMI weather station.
     This model defines the operational state data for sensor readings.";

  revision 2026-05-07 {
    description
      "Fix typedef naming conflict (transducer-type renamed to transducer-nature)
       and missing closing brace in enum unknown.";
  }

  revision 2026-03-29 {
    description
      "Move container statistics inside container quantity.";
  }

  revision 2026-03-25 {
    description
      "Add dampening leaf to sensor-alert notification parameters.";
  }

  revision 2026-03-23 {
    description
      "Make active leaf config false (reflects observe subscription state).
       Add must constraint t-min < t-max on sensor-alert.
       Add timestamp-source enum in quantity to indicate timestamp origin
       (source = timestamped by the sensor, receiver = timestamped on receipt).";
  }

  revision 2026-03-22 {
    description
      "Rename branch measurement to transducer
```

```
       (container, list, identity, notification).";
  }

  revision 2026-03-08 {
    description
      "Module renamed to coreconf-m2m for generic M2M CoMI use.";
  }

  revision 2026-03-02 {
    description
      "Major update for CoMI compliance:
       - Added multi-sensor support (id key).
       - Measurements are now configurable (writable).
       - Added standard deviation (stdev).
       - Added uptime to state.
       - Removed tilt from fixed state (now a measurement type).
       - Removed legacy RPCs (subscribe, get-stats) in favor of CoMI/Observe.
       - Cleaned up measurement lists and notifications.";
  }

  revision 2026-02-27 {
    description
      "Initial revision for testing.";
  }

  identity transducer-type {
    description
      "Base identity for all measurement types.";
  }

  identity solar-radiation {
    base transducer-type;
    description "Solar radiation measurement (W/m2).";
  }

  identity precipitation {
    base transducer-type;
    description "Precipitation measurement (mm).";
  }

  identity air-temperature {
    base transducer-type;
    description "Air temperature measurement (°C).";
  }

  identity relative-humidity {
    base transducer-type;
    description "Relative humidity measurement (%).";
```

```
  }

  identity barometric-pressure {
    base transducer-type;
    description "Barometric pressure measurement (kPa).";
  }

  identity vapor-pressure {
    base transducer-type;
    description "Vapor pressure measurement (kPa).";
  }

  identity wind-speed {
    base transducer-type;
    description "Horizontal wind speed measurement (m/s).";
  }

  identity wind-direction {
    base transducer-type;
    description "Wind direction measurement (degrees).";
  }

  identity wind-gust {
    base transducer-type;
    description "Wind gust measurement (m/s).";
  }

  identity strike-count {
    base transducer-type;
    description "Lightning strike count.";
  }

  identity average-distance {
    base transducer-type;
    description "Average lightning distance (km).";
  }

  identity tilt {
    base transducer-type;
    description "Sensor tilt measurement (degrees).";
  }

  identity x-orientation {
    base transducer-type;
    description "X-axis orientation (raw accelerometer data).";
  }

  identity y-orientation {
```

```
    base transducer-type;
    description "Y-axis orientation (raw accelerometer data).";
  }

  identity north-wind-speed {
    base transducer-type;
    description "North wind speed component (m/s).";
  }

  identity east-wind-speed {
    base transducer-type;
    description "East wind speed component (m/s).";
  }

  typedef encoding-type {
    type enumeration {
      enum direct {
        value 0;
        description "Value is encoded directly as an integer.";
      }
      enum delta {
        value 1;
        description "Value is encoded as a delta from the previous value.";
      }
    }
    description "Encoding used for the measurement values.";
  }

  typedef transducer-nature {
    type enumeration {
      enum sensor {
        value 0;
        description "Physical sensor; quantity can be read.";
      }
      enum actuator {
        value 1;
        description "Actuator; quantity can be written.";
      }
      enum sensor-actuator {
        value 2;
        description "Both sensor and actuator.";
      }
      enum internal {
        value 3;
        description "Internal operational parameter.";
      }
      enum unknown {
        value 255;
```

```
        description "Unknown.";
      }
    }
    description "Nature of a transducer.";
  }

  grouping transducers-list {
    list time-series {
      key "type id";
      description
        "List of measurements included in this notification.";
      leaf type {
        type identityref {
          base transducer-type;
        }
        description "The type of measurement.";
      }
      leaf id {
        type uint8;
        description "Instance identifier for this measurement type.";
      }
      leaf-list values {
        type int64;
        ordered-by user;
        description "List of encoded measurement values.";
      }
    }
  }

  container state {
    config false;
    description "Operational state data of the weather station.";
    leaf uptime {
      type uint64;
      units "seconds";
      description "Time elapsed since the last boot.";
    }
  }

  container characteristics {
    description "Static descriptive information about the device.";
    leaf name {
      type string;
      description "Human-readable name of the device.";
    }
    leaf version {
      type string;
      description "Firmware or software version of the device.";
```

```
    }
    leaf identifier {
      type string;
      description "Unique identifier of the device (e.g., serial number or EUI).";
    }
    uses geo:geo-location {
      refine "geo-location/reference-frame/geodetic-system/geodetic-datum" {
        default "wgs-84";
      }
    }
  }

  container transducers {
    description
      "Current measurement values accessible via polling or configuration.";
    list transducer {
      key "type id";
      description "List of available measurements.";
      leaf type {
        type identityref {
          base transducer-type;
        }
        description "The type of measurement.";
      }
      leaf id {
        type uint8;
        description
          "Instance identifier to distinguish multiple sensors of the same type.";
      }
      leaf unit {
        type string;
        description "Unit of measurement (e.g., 'C', 'm/s', '%').";
      }
      leaf nature {
        type transducer-nature;
        description
          "Indicates whether this is a sensor, actuator, computed value,
           or internal parameter.";
      }
      leaf precision {
        type int8;
        default 1;
        description
          "Number of decimal places. Real value = raw_value * 10^-precision.";
      }
      container quantity {
        config false;
        description "Operational state for this transducer.";
```

```
        leaf value {
          type int64;
          description "The current raw integer value of the transducer.";
        }
        leaf timestamp {
          type uint64;
          units "seconds";
          description "Epoch timestamp (seconds) of the last transducer update.";
        }
        leaf u-timestamp {
          type uint32;
          units "microseconds";
          description "Microsecond fraction of the timestamp (0..999999).";
        }
        leaf timestamp-source {
          type enumeration {
            enum source {
              value 0;
              description "Timestamp generated by the sensor itself.";
            }
            enum receiver {
              value 1;
              description "Timestamp applied by the receiver upon arrival.";
            }
          }
          description "Origin of the timestamp associated with this measurement.";
        }
        container statistics {
          config false;
          description "Read-only accumulated statistics for this measurement.";
          leaf min {
            type int64;
            description "Minimum value observed since last reset.";
          }
          leaf max {
            type int64;
            description "Maximum value observed since last reset.";
          }
          leaf mean {
            type int64;
            description "Mean value observed since last reset.";
          }
          leaf median {
            type int64;
            description "Estimated median value observed since last reset.";
          }
          leaf stdev {
            type uint64;
```

```
            description "Standard deviation since last reset.";
          }
          leaf sample-count {
            type uint64;
            description "Number of samples used for statistics calculation.";
          }
        }
      }
      container notification-parameters {
        description
          "Configuration parameters controlling how this transducer
           is reported in CoMI notification streams.";
        container history {
          description "Parameters for the history time-series stream.";
          leaf active {
            type boolean;
            config false;
            description
              "True when a FETCH+Observe subscription is active on ‘/s‘
               for this transducer history stream.";
          }
          leaf step {
            type uint32;
            units "milliseconds";
            description "Time interval between samples in the notification stream.";
          }
          leaf precision {
            type uint8;
            default 1;
            description
              "Number of decimal places for values encoded in time-series
               notifications.";
          }
          leaf max-samples {
            type uint32;
            description "Maximum number of samples kept in the time-series buffer.";
          }
          leaf time-period {
            type uint32;
            units "seconds";
            description "Duration of the time window covered by the history buffer.";
          }
          leaf encoding {
            type encoding-type;
            default "delta";
            description "Encoding used for the values list in notifications.";
          }
          leaf max-payload {
```

```
            type uint32;
            units "bytes";
            description
              "Maximum payload size in bytes. When exceeded, the notification
               is sent early. 0 means no size limit.";
          }
        }
        container sensor-alert {
          description "Parameters for threshold-based sensor-alert notifications.";
          must "not(t-min and t-max) or t-min < t-max" {
            error-message "t-min must be strictly less than t-max.";
          }
          leaf active {
            type boolean;
            config false;
            description
              "True when a FETCH+Observe subscription is active on ‘/s‘
               for this transducer sensor-alert stream.";
          }
          leaf t-min {
            type int32;
            description "Minimum threshold. Alert raised when value falls below.";
          }
          leaf t-max {
            type int32;
            description "Maximum threshold. Alert raised when value exceeds.";
          }
          leaf hysteresis {
            type uint8 {
              range "0..100";
            }
            default 5;
            units "percent";
            description
              "Hysteresis applied to threshold crossings, as a percentage.";
          }
          leaf dampening {
            type uint32;
            default 0;
            units "milliseconds";
            description
              "Minimum time between two consecutive sensor-alert notifications.
               0 means no dampening.";
          }
        }
      }
      action reset-stats {
        description
```

```
          "Reset accumulated statistics for this transducer.";
      }
    }
  }

  rpc reset-stats {
    description
      "Reset accumulated statistics for all transducers.";
  }

  notification history {
    description
      "History time-series notification sent to Observe subscribers.";
    leaf last {
      type boolean;
      default "false";
      description "True if this is the last notification for the current series.";
    }
    uses transducers-list {
      augment "time-series" {
        container internal {
          description "Internal operational state for this time-series entry.";
          leaf last-update {
            type uint64;
            units "seconds";
            description "Epoch timestamp of the last sample appended.";
          }
          leaf start-time {
            type uint64;
            units "seconds";
            description "Epoch timestamp of the first sample.";
          }
          leaf messages-sent {
            type uint64;
            description "Number of notification messages sent since subscription.";
          }
        }
      }
    }
  }

  notification sensor-alert {
    description "Alert notification when a value exceeds a threshold.";
    list target {
      key "type id";
      description "Measurement instances that triggered this alert.";
      leaf type {
        type identityref {
```

```
          base transducer-type;
        }
      }
      leaf id {
        type uint8;
      }
      leaf value {
        type int64;
        description "The measurement value that triggered this alert.";
      }
    }
  }
}
```

Figure 11: Complete coreconf-m2m YANG module

Appendix B. SID File (CSV)

The following table lists the SID assignments for the coreconf-m2m module (assignment range: 100000100399, revision 2026-03-29).

```
SID,Namespace,Identifier
100000,module,coreconf-m2m
100001,identity,air-temperature
100002,identity,average-distance
100003,identity,barometric-pressure
100004,identity,east-wind-speed
100005,identity,north-wind-speed
100006,identity,precipitation
100007,identity,relative-humidity
100008,identity,solar-radiation
100009,identity,strike-count
100010,identity,tilt
100011,identity,transducer-type
100012,identity,vapor-pressure
100013,identity,wind-direction
100014,identity,wind-gust
100015,identity,wind-speed
100016,identity,x-orientation
100017,identity,y-orientation
100018,data,/coreconf-m2m:characteristics
100019,data,/coreconf-m2m:characteristics/geo-location
100020,data,/coreconf-m2m:characteristics/geo-location/height
100021,data,/coreconf-m2m:characteristics/geo-location/latitude
100022,data,/coreconf-m2m:characteristics/geo-location/longitude
100023,data,/coreconf-m2m:characteristics/geo-location/reference-frame
100024,data,/coreconf-m2m:characteristics/geo-location/reference-frame/alternate-system
100025,data,/coreconf-m2m:characteristics/geo-location/reference-frame/astronomical-body
```

```
100026,data,/coreconf-m2m:characteristics/geo-location/reference-frame/geodetic-system
100027,data,/coreconf-m2m:characteristics/geo-location/reference-frame/geodetic-system/co
ord-accuracy
100028,data,/coreconf-m2m:characteristics/geo-location/reference-frame/geodetic-system/ge
odetic-datum
100029,data,/coreconf-m2m:characteristics/geo-location/reference-frame/geodetic-system/he
ight-accuracy
100030,data,/coreconf-m2m:characteristics/geo-location/timestamp
100031,data,/coreconf-m2m:characteristics/geo-location/valid-until
100032,data,/coreconf-m2m:characteristics/geo-location/velocity
100033,data,/coreconf-m2m:characteristics/geo-location/velocity/v-east
100034,data,/coreconf-m2m:characteristics/geo-location/velocity/v-north
100035,data,/coreconf-m2m:characteristics/geo-location/velocity/v-up
100036,data,/coreconf-m2m:characteristics/geo-location/x
100037,data,/coreconf-m2m:characteristics/geo-location/y
100038,data,/coreconf-m2m:characteristics/geo-location/z
100039,data,/coreconf-m2m:characteristics/identifier
100040,data,/coreconf-m2m:characteristics/name
100041,data,/coreconf-m2m:characteristics/version
100042,data,/coreconf-m2m:history
100043,data,/coreconf-m2m:history/last
100044,data,/coreconf-m2m:history/time-series
100045,data,/coreconf-m2m:history/time-series/id
100046,data,/coreconf-m2m:history/time-series/internal
100047,data,/coreconf-m2m:history/time-series/internal/last-update
100048,data,/coreconf-m2m:history/time-series/internal/messages-sent
100049,data,/coreconf-m2m:history/time-series/internal/start-time
100050,data,/coreconf-m2m:history/time-series/type
100051,data,/coreconf-m2m:history/time-series/values
100052,data,/coreconf-m2m:reset-stats
100053,data,/coreconf-m2m:reset-stats/input
100054,data,/coreconf-m2m:reset-stats/output
100055,data,/coreconf-m2m:sensor-alert
100056,data,/coreconf-m2m:sensor-alert/target
100057,data,/coreconf-m2m:sensor-alert/target/id
100058,data,/coreconf-m2m:sensor-alert/target/type
100059,data,/coreconf-m2m:sensor-alert/target/value
100060,data,/coreconf-m2m:state
100061,data,/coreconf-m2m:state/uptime
100062,data,/coreconf-m2m:transducers
100063,data,/coreconf-m2m:transducers/transducer
100064,data,/coreconf-m2m:transducers/transducer/id
100065,data,/coreconf-m2m:transducers/transducer/nature
100066,data,/coreconf-m2m:transducers/transducer/notification-parameters
100067,data,/coreconf-m2m:transducers/transducer/notification-parameters/history
100068,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/active
100069,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/encoding
100070,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/max-payl
oad
100071,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/max-samp
les
100072,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/precisio
n
100073,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/step
```

```
100074,data,/coreconf-m2m:transducers/transducer/notification-parameters/history/time-per
iod
100075,data,/coreconf-m2m:transducers/transducer/notification-parameters/sensor-alert
100076,data,/coreconf-m2m:transducers/transducer/notification-parameters/sensor-alert/act
ive
100077,data,/coreconf-m2m:transducers/transducer/notification-parameters/sensor-alert/dam
pening
100078,data,/coreconf-m2m:transducers/transducer/notification-parameters/sensor-alert/hys
teresis
100079,data,/coreconf-m2m:transducers/transducer/notification-parameters/sensor-alert/t-m
ax
100080,data,/coreconf-m2m:transducers/transducer/notification-parameters/sensor-alert/t-m
in
100081,data,/coreconf-m2m:transducers/transducer/precision
100082,data,/coreconf-m2m:transducers/transducer/quantity
100083,data,/coreconf-m2m:transducers/transducer/quantity/statistics
100084,data,/coreconf-m2m:transducers/transducer/quantity/statistics/max
100085,data,/coreconf-m2m:transducers/transducer/quantity/statistics/mean
100086,data,/coreconf-m2m:transducers/transducer/quantity/statistics/median
100087,data,/coreconf-m2m:transducers/transducer/quantity/statistics/min
100088,data,/coreconf-m2m:transducers/transducer/quantity/statistics/sample-count
100089,data,/coreconf-m2m:transducers/transducer/quantity/statistics/stdev
100090,data,/coreconf-m2m:transducers/transducer/quantity/timestamp
100091,data,/coreconf-m2m:transducers/transducer/quantity/timestamp-source
100092,data,/coreconf-m2m:transducers/transducer/quantity/u-timestamp
100093,data,/coreconf-m2m:transducers/transducer/quantity/value
100094,data,/coreconf-m2m:transducers/transducer/reset-stats
100095,data,/coreconf-m2m:transducers/transducer/reset-stats/input
100096,data,/coreconf-m2m:transducers/transducer/reset-stats/output
100097,data,/coreconf-m2m:transducers/transducer/type
100098,data,/coreconf-m2m:transducers/transducer/unit
```

Figure 12: SID assignments for coreconf-m2m (CSV format)

Acknowledgments

This work has been supported by the SCHC Chair from IMT Atlantique and Afnic.

Author’s Address

Laurent Toutain
IMT Atlantique
Email: laurent.toutain@imt-atlantique.fr