Feature Specification for

SiESoCom Sensor Node

'S-Mote'

1.2.2

REVISION HISTORY

DATE	VERSION	INFO
MARCH 19, 2021	1.0	CREATION
MAY 5, 2021	1.1	ADD DATASCHEMA INFO IN SECTION 3.2.2
MAY 17, 2021	1.2	TOPICS AND DATA SCHEMA UPDATE
MAY 17, 2021	1.2.1	BASELINING 1.2 W/O MODS
JULY 2, 2021	1.2.2	CLEAN UP, COSMETICS.

CONTENTS

1 Introduction

Following a change of specification, the SiESoCom project will *not* use a Fraunhofer 'Krill' smart sensor any longer: The acoustic event inference and file capture system functions are now implemented as a M^3 multi-tier managed edge applications. As a result of this system topology change, the SiESoCom sensor requirements are simplified.

The SiESoCom sensor function is now limited to acquiring, framing and streaming the acoustic data into the SiESoCom setup message bus.

The sensor shall be able to support both (i) the M^3 device management data path and (ii) a custom-design acoustic application data path.

A single setup shall support more than one sensor.

Also, a formal data serialization process is introduced to manage the bandwidth on the application message bus

2 Embedded System Topology

The acoustic sensor nodes, like any M^3 device shall be connected to the M^3 cloud via an edge core which acts as both a gateway (protocol and physical interface mediation) and the host of edge processing.

The device management data path (in blue on fig 1 below) is used to commission, initialize and administrate the devices and gateways. This data path is supported by an M^3 agent that is integrated into the software of the devices , the gateways and the containerized applications.

The actual user data (in this case raw acoustic data and time series) shall be routed to the M^3 application data store and possibly to Siemens Data Lake via a custom data path. (in orange on fig 1 below). This data path is supported by containerized edge services and applications.

The diagram on next page depicts the SW/HW placement topology.

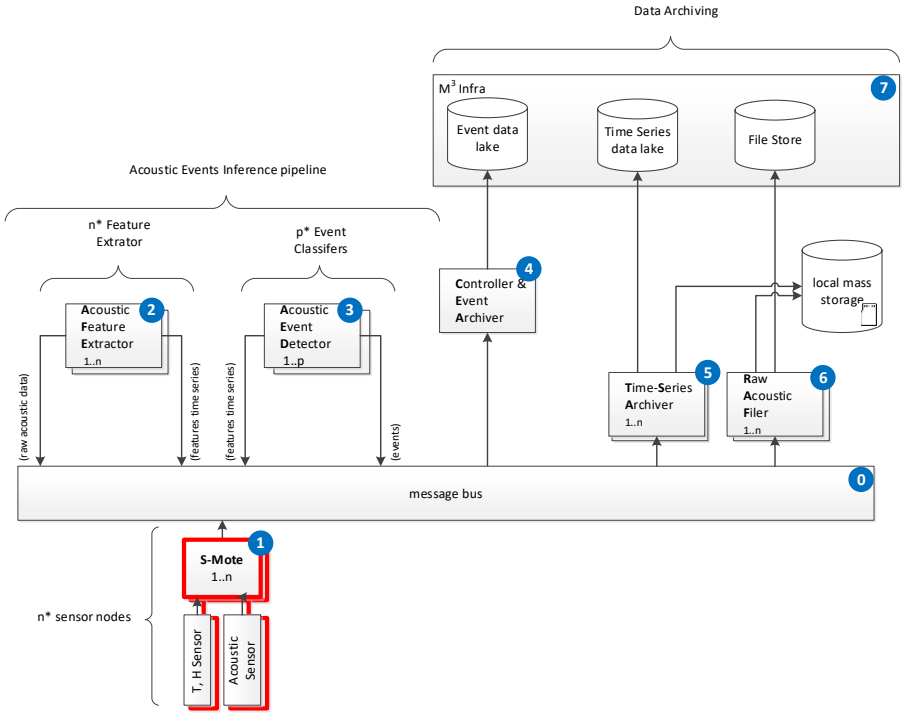


Fig: SiESoCom application topology – all tiers -

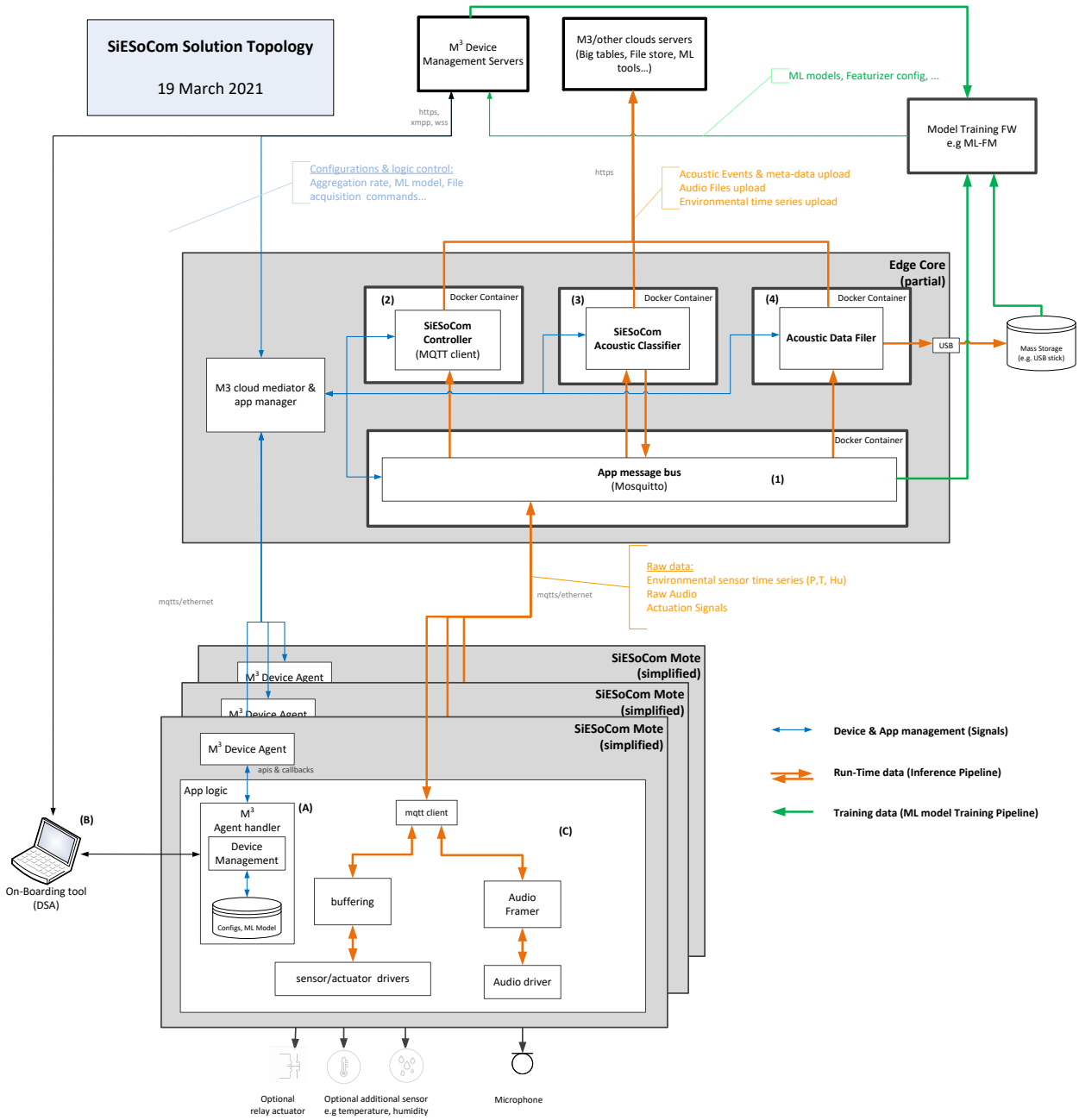


Fig 1: M³ for SiESoCom partitioning

The following software components are concerned by this project:

On the device:

(A) **M³ agent and its handler**: This component accepts cloud configurations and operation requests via the device management path (in blue in fig 1). The SiESoCom sensor shall be modeled in M³ cloud in terms of input configurations, output configurations and managed operations. The agent is provided as a compiled library for the target hardware.

(B) **M³ device on-boarding tool**: The secure device management of M³ solution requires the mote hardware to be on-boarded using a dedicated/secured M³ tools: Device Setup Application (DSA). This DSA tools securely connects to both the mote using either a serial link or a TCP socket (local IP address/port 9005) and to the M³ backend using operator credentials, possibly using 2FA. Once this secure bridge is established, DSA injects initialization data and credentials into the device.

(C) **Mote application logic**: In this topology, the mote application role is mostly to acquire and up-stream acoustic frames and, optionally sensor time series e.g. temperature, humidity... The mote application logic is therefore limited to acoustic data framing and time series acquisition.

On the M³ edge server(s):

(1) **Application Message Bus**: Implemented as an M³ managed MQTT broker. This is an existing, qualified M³ containerized application, supporting both plain and encrypted communication. It shall be used 'as is'.
(2) **SiESoCom Controller/Aggregator**: This containerized application is implemented by Mentor/Siemens. It is in charge of (i) managing the message bus accounts, (ii) aggregating time series (simple compression for now) and uploading them into M³ big tables, (iii) accept an upload acoustic events into M³ big table detected by the SiESoCom Acoustic Classifier(s).
(3) **SiESoCom Acoustic Classifier**: This containerized application implements the machine learning algorithm(s) in charge of detecting acoustic events from the raw audio stream provided by the various SiESoCom sensors via the message bus. This app and/or its trained models are provided by Fraunhofer IDMT. Mentor can provide a reference implementation as needed. Multiple instances of this application can co-exist on the same setup e.g. to detect different types of events.
(4) **Acoustic Data Filer**: This containerized application creates and stores raw acoustic files either on a local USB mass storage device (e.g. USB stick, SSD disk) or on M³ cloud-hosted file store. Mentor/Siemens provides this application.

Notice: As the SiESoCom project progresses, additional edge or central applications will be created. This topology reflects the first implementation only.

3 Mote Functional Specification

3.1 Mote Modeling Background

Class object & inheritance: All device parameters are modeled in M^3 class objects. M^3 managed device instances then inherit their parameters from a hierarchy of class objects.

In the case of device, the inheritance hierarchy is as follow:

1. **Device common configuration**: Parameters common to all M^3 devices: location, identification...
2. **Device type**: Parameters for M^3 devices of a given type: mote, printer, .. (empty for now). Here, the SiESoCom sensor will be modeled as a 'M3 Mote' type
3. **Device model**: Parameters for M^3 devices of a given model: Typically, the parameters that pertains to the hardware specifications of a device. Here, we create a 'SiESoCom' device model.
4. **Device firmware agent**: Parameters for M^3 device Agent: Device management connection credential and details (not to be customized)
5. **Device firmware**: Parameters for M^3 device application: Device management operations, networking reporting, application specific parameters. Typically, the parameters that pertains to the device software logic. Here, we create a 'SiESoCom mote FW – ethernet' firmware model.

Config parameter attributes: In general, M^3 device parameters have the following attributes (not exhaustive)

- Data direction: Defines if the parameters is from cloud-to-device ('in'), device-to-cloud ('out'), or both ('in/out'). The parameter can also be fully managed from the cloud ('backend only')
- Data Type: Int, float, string.
- Visibility: Defines how the parameter is rendered on the portal interface and whether it can be updated by an operator. The possible settings are 'View only', 'Private' (on user's passcode presentation), 'editable' (by admin user), 'hidden' (only for programmatic handling)
- Unit
- Range or discreet list of supported values: Defines a range ([8,56]) or a list of discrete values ({"Initializing", "Ready", "Error", ...]}) the parameter value can take.
- Default Value: The value for the parameter at the time the resource is instantiated.

Note: parameter pub/sub

- *In order to help the management of multi-container and multi-instance applications, any M³ instance parameters can 'subscribe' to any other M³ instance parameters: M³ backend will automatically notify all the subscribers when a subscribed value has changed. For example, a client application 'host IP' parameter can subscribe to the 'public IP' parameter of a server application to resolve its initial or updated server IP*

3.2 SiESoCom mote modeling

This sections below maps the parameters defining the 'SiESoCom mote' device and its embedded logic in term of **device type**, **device model** and **device firmware .** These parameters are subject to evolution.

A SiESoCom sensor is modeled as

- type : 'M3 Mote'
- model: 'SiESoCom'.
- Agent: Currently 'Linux Agent 2.0.2'
- Firmware: 'SiESoCom FW – ethernet'

Notice. If necessary, a WiFi variant can be created. For now, we assume ethernet only.

Other parameters are inherited from **'device common'**, and **'device agent'** class objets. However, they are mentioned for information and are not subject to customization.

3.2.1 Type modeling

SiESoCom mote if of Type 'M3 Mote'. This type have no config parameters of it own. This section is therefore empty.

3.2.2 Model modeling

The below table documents the "SiESoCom" model modeling requirements as an evolution/modification of the "M3 Generic Peripheral V2" model.

The SiESoCom mote has optional temperature & humidity sensors and a mandatory acoustic sensor. From the "M3 Generic Peripheral V2" model, we remove the pressure sensor, the actuator and we add a channel count on the acoustic sensor.

In addition, we indicate the name of the Data Schema associated to the sensed time series inside each sensor section, and the topic to which the stream is published. Both these new configs are out, readonly. They are meant to support the connection of these endpoints to other nodes.

▼ Model	
Setup Name	SiSoCom_1
Sensors	
Sensor 1	Microphone
State	on
Serial Number	001
Type	IDMT-microphone
Unit	db
Sampling Frequency (KHz)	48
Audio Sample Size (bit)	16
Audio Frame Payload Length	8192
Channel	Stereo
Gain (%)	100
Topic	SiSoCom_1/device/acoustic/CSP-B827EB24A797
Data Schema	
Sensor 2	Temperature
State	on
Serial Number	002
Type	Temperature
Unit	C
Sampling Frequency (Hz)	1
Reporting Count	1
Topic	SiSoCom_1/device/readings/CSP-B827EB24A797
Data Schema	
Sensor 3	Humidity
State	on
Serial Number	003
Type	Humidity
Unit	%
Sampling Frequency (Hz)	1
Reporting Count	1
Topic	SiSoCom_1/device/readings/CSP-B827EB24A797
Data Schema	

3.2.2.1 *Acoustic data schema*

For this model, the audio stream has no header and the data schema shall therefore be encoded as the following Avro schema:

```
Avro Schema / Binary frame 8192 - no header
{
```

```
    "type": "record",
    "name": "SiESoComAcousticFrame.8192.noheader",
    "namespace": "rc_data_schema_repo",
    "fields": [
        {"type" : "fixed" , "name" : "AudioFrame", "size" : 8192 }
    ]
}
```

Avro Schema / Binary frame 4096 - no header

```
{
    "type": "record",
    "name": "SiESoComAcousticFrame.4096.noheader",
    "namespace": "rc_data_schema_repo",
    "fields": [
        {"type" : "fixed" , "name" : "AudioFrame", "size" : 4096 }
    ]
}
```

Avro Schema / Binary frame 2048 - no header

```
{
    "type": "record",
    "name": "SiESoComAcousticFrame.2048.noheader",
    "namespace": "rc_data_schema_repo",
    "fields": [
        {"type" : "fixed" , "name" : "AudioFrame", "size" : 2048 }
    ]
}
```

Avro Schema / Binary frame 1024 - no header

```
{
    "type": "record",
    "name": "SiESoComAcousticFrame.1024.noheader",
    "namespace": "rc_data_schema_repo",
    "fields": [
        {"type" : "fixed" , "name" : "AudioFrame", "size" : 1024 }
    ]
}
```

3.2.2.2 *Temperature & Humidity time-series data schema:*

For this model, the temperature & Humidity time series are reported as an array of timetagged values and the data schema shall therefore be defined as the following Avro schema:

Example of a M3 sensor array of time tagged scalars

```
{
  "endPointName" : "Temperature",
  "Payload" : [ {
    "Date" : 1619699822940,
    "Value" : 51.262573
  }, {
    "Date" : 1619699823927,
    "Value" : 51.251892
  }, {
    "Date" : 1619699824940,
    "Value" : 51.273254
  } ]
```

```
}
```

Avro schema / array of timetagged scalars

```
{
  "name": "ScalarTimeSeries",
  "type": "record",
  "namespace": "rc_data_schema_repo",
  "fields": [
    {
      "name": "endPointName",
      "type": "string"
    },
    {
      "name": "Payload",
      "type": {
        "type": "array",
        "items": {
          "name": "Payload_record",
          "type": "record",
          "fields": [
            {
              "name": "Date",
              "type": "long"
            },
            {
              "name": "Value",
              "type": "float"
            }
          ]
        }
      }
    }
  ]
}
```

So, the value to set for the new Data Schema config parameter are:

Sensor1.Data Schema="SiESoComAcousticFrame.8192.noheader"

Sensor2.Data Schema="ScalarTimeSeries"

Sensor3.Data Schema="ScalarTimeSeries"

3.2.2.3 Topics

For this model, we explicitly indicate the topics to which for each sensor publish data, per the following rules:

Acoustic sensor:

<setup name>/device/acoustic/<device S/N>/<sensor S/N>

Where:

setup_name is the unique user-defeined name of the current setup

device SN is the commissioned serial number for the device.
sensor SN is the commissioned serial number for the device sensor

e.g. SiESoComSetup_1/device/acoustic/CSP-B827EB2DB0F1/001

T or Hu time series:

<setup name>device/readings/<device S/N>

Where:

setup_name is the unique user-defeined name of the current setup
device SN is the commissioned serial number for the device.
sensor SN is the commissioned serial number for the device sensor

e.g. SiESoComSetup_1/device/readings/CSP-B827EB2DB0F1/002

3.2.3 Firmware modeling

The below table documents the 'SiESoCom FW – ethernet' firmware modeling requirements.

Name	Default Value	Data Direction	Data Type
Networking			
Adapters	-	backendonly	string
eth0	-	backendonly	string
Type	-	out	string
Interface State	-	out	string
Current IP Address	-	out	string
MAC Address	-	out	string
Gateway Address	-	out	string
Netmask Address	-	out	string
Use DHCP	-	out	string
Error	-	out	string
Device Monitoring			
CPU/Memory Monitoring	on	in	string
Sampling Period (sec)	5	in	int
Reporting Period (sec)	30	in	int
Load Monitoring	on	in	string
Sampling Period (sec)	2	in	int
Reporting Period (sec)	30	in	int
Networking Monitoring	on	in	string
Sampling Period (sec)	3	in	int
Reporting Period (sec)	45	in	int
Disk Monitoring	on	in	string
Sampling Period (sec)	20	in	int
Reporting Period (sec)	60	in	int
Swap Monitoring	on	in	string
Sampling Period (sec)	2	in	int
Reporting Period (sec)	30	in	int
Application			
Mote State	Offline	out	string
Security Mode	-	in	string
Controller State	not started	in	string
Application Protocol	mqtt	backendonly	string
Physical Interfaces	ethernet	backendonly	string
Broker host	-	backendonly	string
IP Address	-	in	string
Port	-	in	string
State	not started	in	string
Image	-	backendonly	string
Version	-	out	string
State	Applied	out	string
Application Credentials			
Mote MQTT Account	-	backendonly	string
Username	-	in	string
Password	**********	in	string
TLS Certificates	-	backendonly	string
CA Cert File	**********	in	string
Cert File	**********	in	string
Key File	**********	in	string
Expiration Date	-	out	string

3.2.4 Common & Agent parameters

As mentioned above, the following config parameters, inherited from 'Common' and 'Agent' class objects are mentioned for information and are not subject to customization

Common Config Parameter

A ‘SiESocom’ device, like every other M^3 device will inherit the following ‘common’ parameters:

Agent Connection Status:

Status (out)
First Connection Date (backend only)

Note: The agent is maintaining its own state machine as follow:

Initially the agent is in 'blank' state, which means the system has not been initialized and cannot connect to M³ cloud. In this state, the agent is awaiting an AOIM client connection to be initialized and injected connection information. Nothing can be done except connect a genuine AOIM client application and carry-out the device initialization.

After a successful initialization, the agent establishes a secure connection to the M³ mediator and starts relaying device management requests from/to M³ cloud to/from the mote application.

>> IMPORTANT: On device 'detach', the agent returns to 'blank' and notifies the application so that it can clean-up for detach. The app is then asked to reboot the device. The agent will then await a new AOIM initialization connection.

Please refer to our ref. code for further details.

Device Identification

Serial Number (backend only)
Name (backend only)
Batch (backend only, optional)
Manufacturer (backend only)
Manufacturing date (backend only)
Type (backend only)
Model (backend only)
Firmware Model Version (backend only)
Hardware Version Batch (backend only)
Info (backend only, optional)
Installation date (backend only)

These identification parameter values are set by the operator at the time the device is commissioned.

Device Location

Site (backend only)
Country (backend only)
- City (backend only)
- Postal Code (backend only)
- Latitude (out, optional)
- Longitude (out, optional)

These location parameter values are set by the operator at the time the device is commissioned. The optional GPS coordinates can be set by the device

Agent Config Parameter

A 'SiESoCom' device, like every other M^3 device will inherit the following 'agent parameters:

Device Management

Auto Reboot (backend only)
- Auto reboot period
- Auto reboot UTC time

>> IMPORTANT: Auto reboot schedule is managed by the agent, but the actual reboot is an application-controlled operation. Please refer to our ref. code for further details

Agent Credential

Agent MQTT Account (backend only)
- Username (in)
- Password (in)

Note: The Agent account credential are automatically set by the backend logic at the time a commissioned device is attached to a particular gateway.

Agent

C2AM interface (backend only)
- Application protocol (backend only, set to mqtt in this case)
- Phy (backend only, set to Ethernet in our case)
- Version (out)

3.3 Mote Commissioning and Initialization.

Before the mote device can be used, it has to be commissioned and initialized.

This procedure involves the following steps:

1. Mote is **commissioned** in M^3 cloud, using M^3 portal: A cloud instance ('avatar') of the device is created with all the parameters defined in section 3.1.
2. The commissioned instance is **attached** to a M^3 edge server instance (gateway). This is also a backend operation that 'links' a mote device to its driving gateway.

3. A 'blank' physical device (i.e. a physical device loaded with the SiESoCom firmware image but without M^3 operational credentials) is connected to a secure device setup application (DSA) hosted on a PC via a serial interface or TCP AOIM. (see Fig 1)
4. An authorized M^3 user logs in to M^3 backend from the DSA application and selects the device to initialize.
5. The DSA application injects into the mote agent a series of credentials stored/generated on its instance M^3 avatar that will allow the mote agent (and therefore the mote device) to (i) authenticate the M^3 cloud, (ii) be authenticated by M^3 cloud, (iii) equipped with operational credentials (TLS certificate) and (iv) provided with its host device management connection details.
6. The agent then transition from 'blank' to 'operational' and connects to the Device Management mediator. The mote application is then provided with all its device management parameters and starts, as defined in section 3.4.1

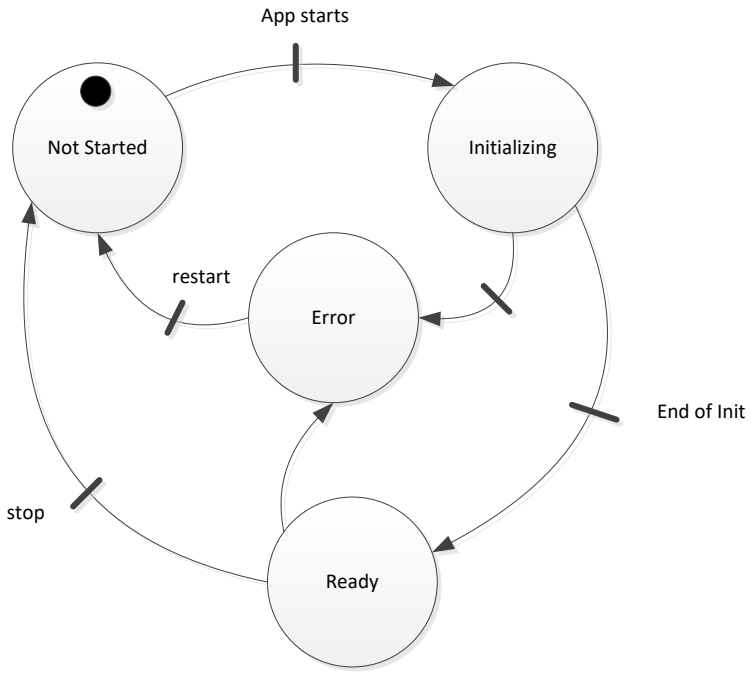
3.4 SiESoCom Mote Application Specifications

3.4.1 Application states

By design rule, M^3 devices and managed applications shall maintain and report the following states to M^3 backend: "Not Started", "Initializing", "Ready" and "Error".

The SiESoCom application firmware shall therefore manage these states.

Note: The compliance to this state values will guarantee seamless synchronization between the various contributors of an M^3-managed IOT setup.



“Not Started”

This is the default state, set by the backend at the time the device is commissioned.

“Initializing”

This is the first state the mote application shall report when its boots up.

In the 'Initializing' state, the application shall:

1. Initialize the M^3 agent.
2. Register call-back function to the M^3 agent that is responsible for device management. Call-back functions are invoked by the M^3 agent to pass any requests from the platform to the mote as well as commit changes from the mote to the platform.
3. Wait for the acknowledgement of the M^3 agent initialization via a callback confirming that the M^3 agent is initialized.
4. Get all mote configuration parameters.
5. Acknowledge configuration parameters through the agent interface.
6. Load inference model
7. Establish a connection to the application broker.
8. Transition to Ready State

“Ready”

This is the regular operational state of the application. In this state, the mote shall accept and handle configuration update signals at any time. This events can be (but are not limited to):

- Configuration update request e.g.
  - Update audio frame size, sampling freq, sample width
  - turn on/off the microphone.
  - deploy a new model
- Operational mode change
  - Turn on or off sensors.
- Connectivity event:
  - Application broker or controller transitioning to 'Initializing', 'Ready' or 'Error'.
  - Mote attach or detach request

“Error”

The application shall transition to the Error state in case of any application error. For example, the loss of its application broker or the loss of a container controller preventing normal operation to continue.

3.4.2 Operational modes

In the "Ready" state, the SiESoCom device shall accept the settings of the config parameters, in particular the on/off of each individual sensor: Temperature, Humidity, and Acoustic.

3.4.3 Other Application Logic Elements

Mote detach signal

The “mote detach” is a signal that requires a special handling from the mote sw:

When receiving this signal, the mote application shall clear its config and execute a soft reboot. The agent will then returns the device to its blank state and await a new initialization connection via its AOIM interface. Please refer to reference implementation for details.

3.5 Application data path

As explained in section 1 and 2, the application data path is supported by a containerized message bus (MQTT broker).

3.5.1 Broker host and client account

The MQTT broker host information (IP address and port) and client MQTT Account (username and password) that the SiESoCom application shall use to connect to the message bus are supplied to the application via subscribed device management parameters, as defined in section 3.1.2

The concern parent parameters are ‘Broker Host’ and ‘Mote MQTT Account’

These values are automatically set by the M^3 backend ‘parameter pub/sub’ service: At start up the SiESoCom device application shall read these value from its configuration and establish a connection to the broker.

Notes:

- *The M³ managed MQTT broker can be configured to accept plain text or encrypted communication. It is highly recommended to use encrypted communication. A subscribed parameter provides this information to the mote application that can then connect to the broker with the required level of security*
- *A Last Will and Testament message shall be registered to the MQTT broker to manage the handling of mote disconnection. See reference implementation for details.*

3.5.2 MQTT Topics and payload

Publishing Keep Alive:

Topic: The mote shall publish its current application state every 45 sec to "<setup name>/device/state/<device_sn>" in order to prevent the connection to time-out

Where:

setup_name is the unique user-defeined name of the current setup
device_sn is the commissioned serial number for the device.

e.g. SiESoComSetup_1/device/state/CSP-B827EB2DB0F1

Payload Example:

{"State":"Online"
}

Publishing acoustic frames:

Topic: The mote shall publish acoustic frames per , it's defined data schema on section 3.2.2 on the topic:

<setup name>/device/acoustic/<device S/N>/<sensor S/N>

Where:

setup_name is the unique user-defeined name of the current setup
device SN is the commissioned serial number for the device.
sensor SN is the commissioned serial number for the device sensor

e.g. SiESoComSetup_1/device/acoustic/CSP-B827EB2DB0F1/001

Payload Example:

22FF 21FF 9CFF 3E00 B700 0801 9C01 0202
0502 BE01 5B01 E400 5200 1B00 EBFF 9BFF
25FF 2CFF D3FE BFFE D1FE 8AFE 6EFE 2DFE
2FFE 66FE E3FE 1EFF 07FF 45FF 7EFF B9FF
4B00 ED00 6B01 C601 4901 7800 F3FF ….BDFF

Publishing time series:

Topic: The mote shall publish temperature & humidity time series per their defined data schema on section 3.2.2 on the topic: <setup name>device/readings/<device S/N>

Where:

setup_name is the unique user-defeined name of the current setup
device SN is the commissioned serial number for the device.
sensor SN is the commissioned serial number for the device sensor

e.g. SiESoComSetup_1/device/readings/CSP-B827EB2DB0F1/002

Payload Example:


```
{
 "endPointName" : "Temperature",
 "Payload" : [ {
  "Date" : 1619699822940,
  "Value" : 51.262573
 }, {
  "Date" : 1619699823927,
  "Value" : 51.251892
 }, {
  "Date" : 1619699824940,
  "Value" : 51.273254
 } ]
}
```