ETSI TR 103 675 V1.1.1 (2020-12)

SmartM2M;
AI for IoT: A Proof of Concept

Reference

DTR/SmartM2M-103675

Keywords

architecture, artificial intelligence, IoT, oneM2M

Contents

Figures

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

The present document is addressing the development of a Proof of Concept based on three Use Cases analysed and selected in the associated ETSI TR 103 674 [i.1]. ETSI TR 103 674 [i.1] addresses the issues related to the introduction of AI into IoT systems and, as first priority, into the oneM2M architecture. ETSI TR 103 674 [i.1] has identified and described several Use Cases of which three are used for the development of the Proof of Concept described in the present document.

1 Scope

The following points are discussed:

- Description of the Use Case implemented as a Proof of Concept.
- Description of the implementation: architecture, oneM2M platform used, open source support, etc.
- Main findings regarding the impact on the oneM2M architecture.
- Lessons learned, guidelines and recommendations.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 103 674 (2020): "SmartM2M; Artificial Intelligence and the oneM2M architecture".

[i.2] Kaggle: "Real or Not? NLP with Disaster Tweets".

NOTE: Available at https://www.kaggle.com/c/nlp-getting-started.

[i.3] GloVe: "Global Vectors for Word Representation".

NOTE: Available at https://nlp.stanford.edu/projects/glove/.

[i.4] Readthedocs: "Gateway and backend configuration".

NOTE: Available at https://fiware-openmtc.readthedocs.io/en/latest/reference-doc/gateway-and-backend-configuration/index.html.

[i.5] GitHub: "Gateway and backend configuration".

NOTE: Available at https://github.com/OpenMTC/OpenMTC/blob/master/doc/reference-doc/gateway-and-backend-configuration.md#pluginsopenmtc-cse.

[i.6] ISO/IEC 2382-31:1997: "Information technology -- Vocabulary -- Part 31: Artificial intelligence -- Machine learning".

NOTE: This standard is withdrawn and revised by ISO/IEC 2382:2015.

[i.7] ETSI TR 103 306: "CYBER; Global Cyber Security Ecosystem".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

Artificial Intelligence (AI): refers to "a system's ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation"

Machine Learning (ML):

- Machine Learning is the process by which a functional unit improves its performance by acquiring new knowledge or skills, or by reorganizing existing knowledge or skills.

NOTE 1: Source ISO/IEC 2382-31 [i.6].

- Machine Learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence.

NOTE 2: Source Wikipedia.

oneM2M: Partnership Project (PP) on M2M launched by a number of SSOs including ETSI

open source license: type of license for computer software and other products that allows the source code, blueprint or design to be used, modified and/or shared under defined terms and conditions

NOTE: Examples of popular Open Source licenses are: Apache License 2.0, GNU General Public License (GPL) or Eclipse Public License.

Open Source Software (OSS): computer software that is available in source code form

NOTE: The source code and certain other rights normally reserved for copyright holders are provided under an Open Source license that permits users to study, change, improve and at times also to distribute the software.

Standards Development Organization (SDO): standards setting organization that has a formal recognition by international treaties, regulation, etc.

NOTE: In the present document, SSO is used equally for both Standards Setting Organization or Standards Developing Organizations (SDO).

Standards Setting Organization (SSO): any entity whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting or otherwise maintaining standards that address the interests of a wide base of users outside the standards development organization

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TR 103 306 [i.7] and the following apply:

AI	Artificial Intelligence
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
IoT	Internet of Things
ML	Machine Learning
OSS	Open Source Software
PoC	Proof of Concept

SDO	Standard Development Organization
SSO	Standards Setting Organization
UC	Use Case

4 AI and oneM2M: A Proof of Concept

4.1 The Proof of Concept Use Cases

The emergence of Artificial Intelligence (AI) and Machine Learning (ML) solutions applicable within IoT systems is an opportunity to introduce new functionalities as well as a potential challenge to the existing solutions.

The Proof of Concept (PoC) described in the present document is based on the collection and analysis of relevant use cases done in the associated ETSI TR 103 674 [i.1]. ETSI TR 103 674 [i.1] has made a detailed analysis of the potential impact of AI/ML on the IoT systems architectures and solutions, identified and described several Use Cases of which three are used for the development of the Proof of Concept described in the present document and identified the main validation objectives for the Proof of Concept described in the present document.

4.2 An implementation and the lessons learned

The present document is focused on the presentation of the PoC implementation: a description of the selected UC, the presentation of the technical solutions chosen for the implementation of the UC, and a discussion of the main impacts identified during the implementation.

4.3 Purpose and content of the present document

The present document addresses the implementation of a Proof of Concept (PoC) designed for an evaluation of the potential impact (in particular in terms of improvement) that Artificial Intelligence can bring to IoT systems, more specifically its impact on existing solutions such as the service layer architecture developed by the oneM2M Partnership Project. This PoC is based on the implementation of three Use Cases that has been also described in the associated ETSI TR 103 674 [i.1]. This PoC is using different implementations of a oneM2M platform with the addition of solutions coming, for the most part, from the Open Source Software community.

The target group for the present document is the community of people that is interested in the understanding of the challenges and the potential solutions of the introduction of AI in a concrete implementation of an IoT system.

Clause 5 presents the use cases selected and implemented for the Proof of Concept.

Clause 6 discusses the main elements of each the Use Cases: architecture, resources and attributes, message flow and main implementation characteristics.

Clause 7 presents some lessons learned from the above analysis.

5 Use Cases implemented for the Proof of Concept

5.0 Foreword

This clause is presenting the three Use Cases (UCs) selected for the Proof of Concept. After several Use Cases have been analysed in the associated ETSI TR 103 674 [i.1], three of them have been selected for the Proof of Concept. The rationale behind the selection of these Use Cases is to make sure that they can address a large span of situations, thus ensuring that they touch the largest possible set of issues.

Consequently, the PoC will be comprising three use cases:

- Use Case 1 on Fault detection will address more specifically measurements.

- Use Case 2 on Visual recognition will address more specifically images.
- Use Case 3 on occurrences classification will address more specifically textual content.

The PoC Use Cases have been implemented on two different oneM2M platforms and are available on the ETSI forge (https://labs.etsi.org/rep/iot/smartm2m-ai-for-iot-poc).

5.1 PoC UC1: Fault management and isolation for IoT field devices

Fault detection aims to identify defective states and conditions within computing systems, subsystems and components and ensure their proper functionality to reduce their rate of deterioration, hence better customer experience. There is a need to be an effective maintenance service in place to ensure that IoT devices are running at their best. The inputs of maintenance services are measurements reflecting the health state of the monitored item.

In this use case, an IoT module will be prototyped for fault detection and isolation of IoT device data in a smart building environment using both a rule-based fault detection and a self-learning fault detection algorithm based on e.g. statistics sliding window approach. The rule-based approach would be based on available manufacturer datasheets including rules if available. The self-learning algorithm is based on determining trend vectors and comparing such vectors with longer term historic data.

5.2 PoC UC2: Detection of patterns in video streams

Detection of patterns in video and camera streams enables users to identify scenes, objects, and situations in images uploaded to the service using aims visual recognition based on Artificial Intelligence and Machine learning. Subjects and objects contained in an image are automatically identified, organized and classified into logical categories in order to provide add high added value services in cities such as car vandalism and fire detection.

In this use case, an IoT module will be prototyped for images classification using machine learning and trained data. The IoT module supports multiple classifiers: predefined and custom models. A camera agent will be developed to quickly test the proposed prototype and simplify the integration with real devices within the city. The camera agent reads periodically images from the disk and push them to oneM2M platform. The images could be provided by a real camera or any other external sources.

5.3 PoC UC3: Language-based pattern recognition in social media/crowdsourced data for occurrences classification

In the Smart Cities environment, data comes in many formats from different sources. The common presented use case tends to be through sensors, but data obtained directly from citizens, through what they publish on social networks or share through other means (e.g. mobile apps to report incidents) can be very insightful and valuable. However, this crowd-sourced data is not structured, and its quality can be questionable (i.e. containing typos, incorrect grammar and incomplete location). As such, special care has to be taken in order to obtain insights from it since it can be a useful source of information about what is happening throughout the city in a relatively seamless way.

Within this use case, a prototype of a Common Service Function will be developed, which will provide the functionality of cleaning text data to be used in ML methods. With this functionality, and with the support of the OpenMTC implementation of oneM2M, a system will be developed to fetch data from Tweets or other sources, which can potentially be describing a disaster somewhere, automatically clean the data using the developed CSF, and feed this data into a neural network that classifies the provided text in regards if it is reporting a disaster or not.

6 Details of the Proof of Concept Implementation

6.1 PoC UC1: Fault management and isolation for IoT field devices

6.1.1 Architecture

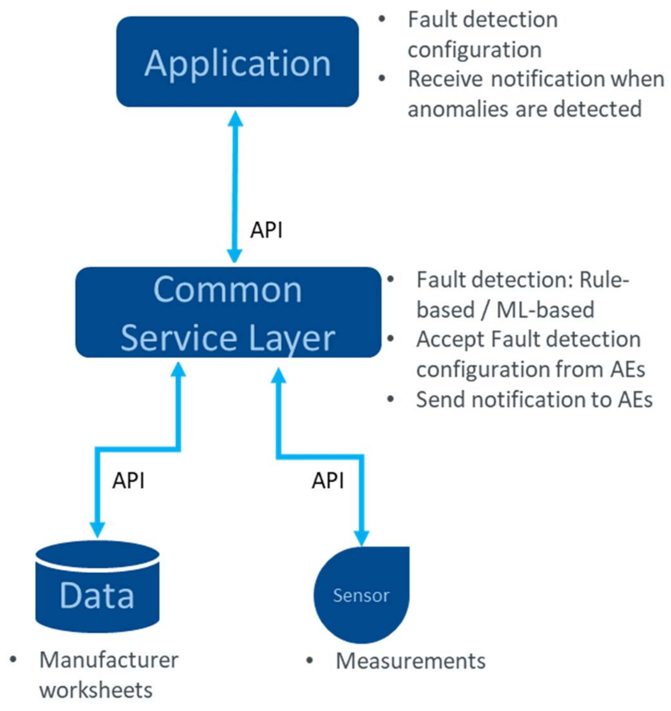

Application
• Fault detection configuration
• Receive notification when anomalies are detected
API
Common Service Layer
• Fault detection: Rule-based / ML-based
• Accept Fault detection configuration from AEs
• Send notification to AEs
API
API
Data
Sensor
• Manufacturer worksheets
• Measurements

Figure 1: High-Level Architecture of Use Case 1

6.1.2 Resources and attributes

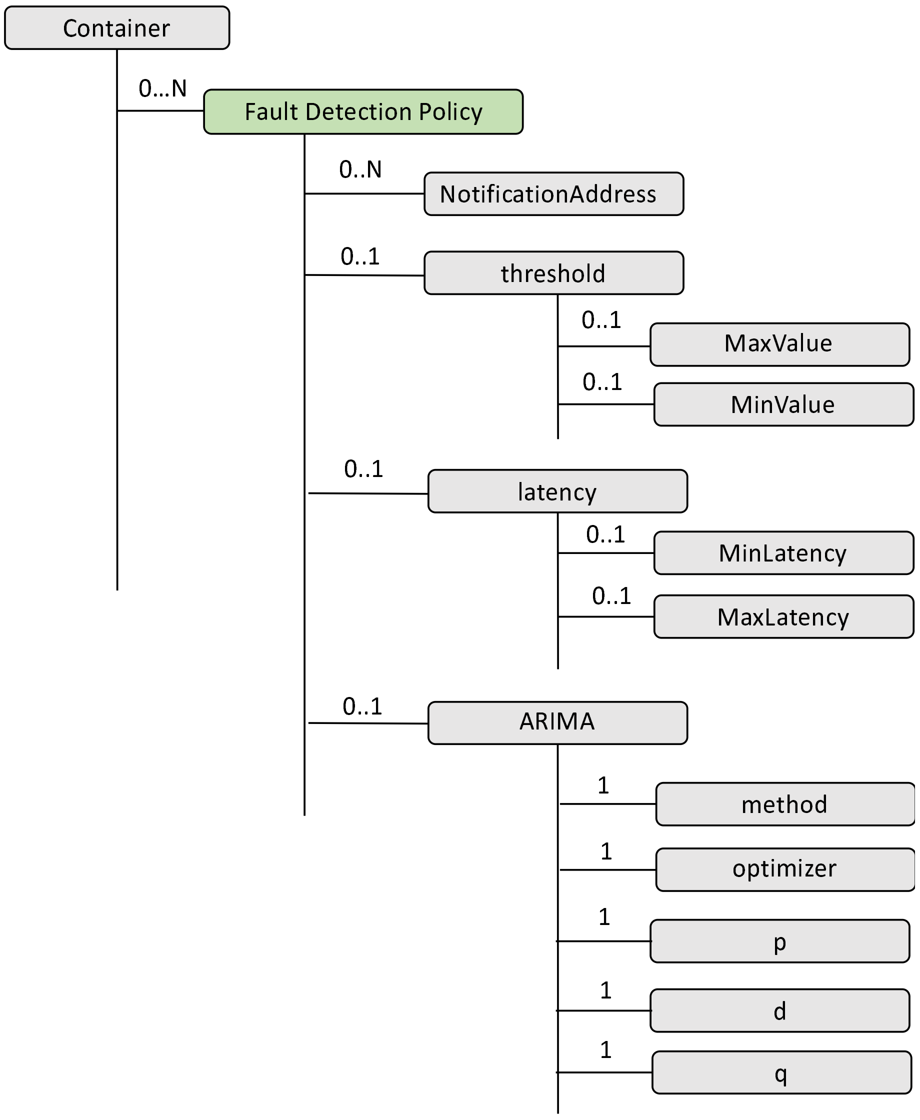


Figure 2: Fault detection policy

6.1.3 Message flow

Figure 3 depicts the entities and main components involved in the use case and the sequence of messages exchanged between them to carry out the functionality needed for fault management and isolation for IoT field devices in oneM2M platform. The blue boxes show the interactions that are achievable with oneM2M platform without any changes, while the green boxes show new interactions that do not currently exist in oneM2M and need to be specified and implemented for this use case.

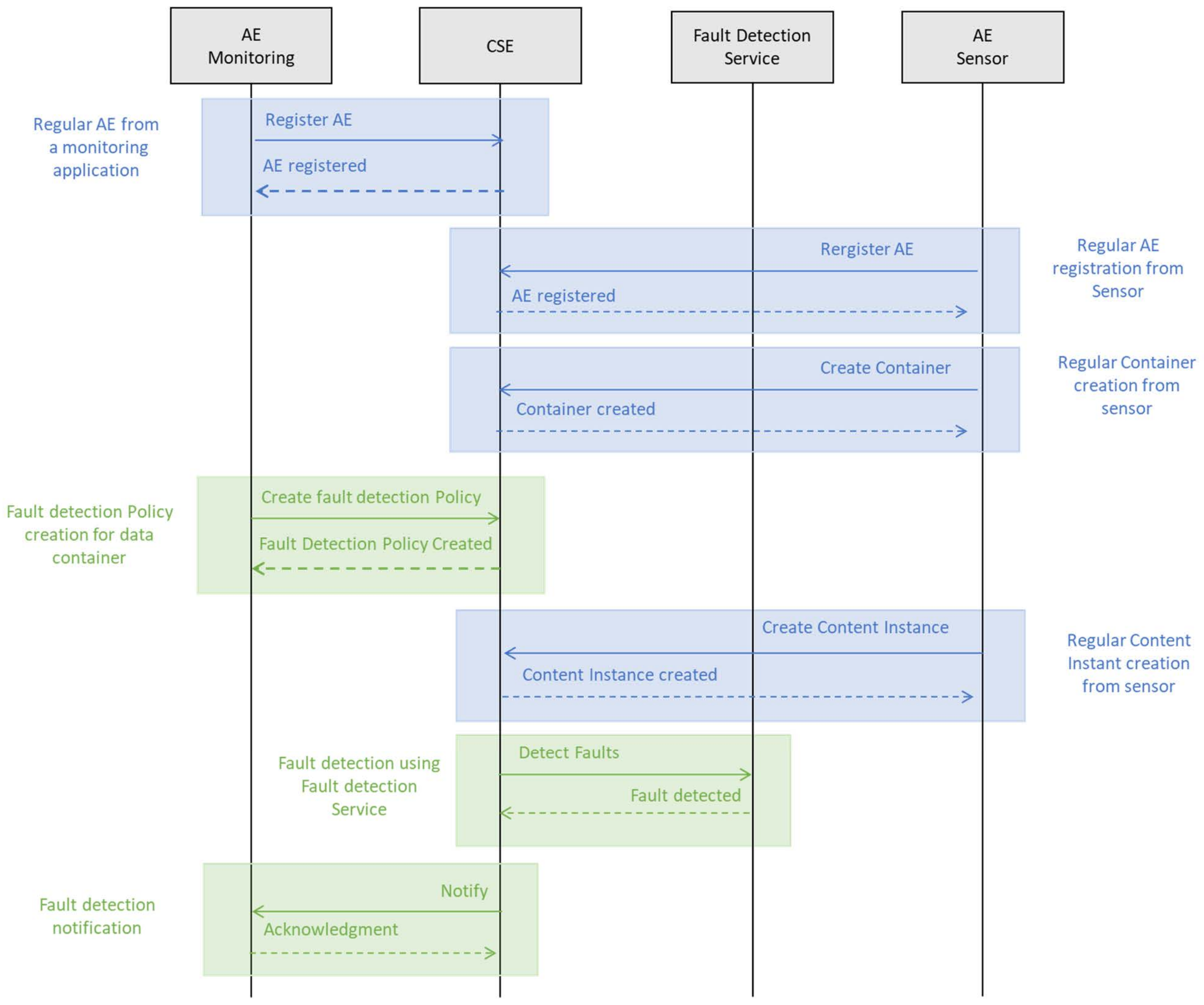

AE Monitoring
CSE
Fault Detection Service
AE Sensor
Regular AE from a monitoring application
Register AE
AE registered
Rergister AE
AE registered
Regular AE registration from Sensor
Create Container
Container created
Regular Container creation from sensor
Fault detection Policy creation for data container
Create fault detection Policy
Fault Detection Policy Created
Create Content Instance
Content Instance created
Regular Content Instant creation from sensor
Fault detection using Fault detection Service
Detect Faults
Fault detected
Fault detection notification
Notify
Acknowledgment

Figure 3: Sequence diagram of Use Case 1

6.1.4 Implementation

Register monitoring AE

HTTP Request

```
POST /server HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin : Cae-monitor
Content-Type: application/json;ty=2
Accept: application/json

{
    "m2m:ae": {
        "rn": "monitor",
        "rr": true,
             "nu":["http://127.0.0.1:4000"]
        "api": "company.org"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/sensor
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:ae": {
        "rn": "monitor",
        "ty": 2,
        "ri": "/CAE988694263",
        "pi": "/server",
        "ct": "20200223T194209",
        "lt": "20200223T194209",
        "acpi": ["/server/acp-69998272"],
        "et": "20180223T194209",
        "api": "company.org",
        "aei": "Cae-monitor",
        "rr": true,
             "nu":["http://127.0.0.1:4000"]
    }
}
```

Register sensor AE

HTTP Request

```
POST /server HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin : Cae-sensor
Content-Type: application/json;ty=2
Accept: application/json

{
    "m2m:ae": {
        "rn": "sensor",
        "rr": false,
        "api": "company.org"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/sensor
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:ae": {
        "rn": "sensor",
        "ty": 2,
        "ri": "/CAE988694263",
        "pi": "/server",
        "ct": "20200223T194209",
        "lt": "20200223T194209",
        "acpi": ["/server/acp-69998272"],
        "et": "20180223T194209",
        "api": "company.org",
        "aei": "Cae-sensor",
        "rr": false
    }
}
```

Create data Container inside sensor AE

HTTP Request

```
POST /server/sensor HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-sensor
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cnt": {
        "rn": "data",
        "mni": 100
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/sensor/data
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:cnt": {
        "rn": "data",
        "ty": 3,
        "ri": "/server/cnt-364492354",
        "pi": "/server/CAE267526506",
        "ct": "20200223T200338",
        "lt": "20200223T200338",
        "acpi": ["/server/acp-940743520"],
        "et": "20180223T200338",
        "st": 0,
        "mni": 100,
        "mbs": 10000,
        "mia": 0,
        "cni": 0,
        "cbs": 0
    }
}
```

Create Fault Detection Policy (threshold based) by monitor inside data Container of sensor AE

HTTP Request

```
POST /server/sensor/data HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-monitor
Content-Type: application/json;ty={type}
Accept: application/json

{
            "m2m:faultDetectionPolicy": {
                     "threshold" :{
                                "maxValue": 100,
                                "minValue: 10
                     },
                     "notificationURI": "/server/monitor"
            }
}
```

HTTP Response

```
201 Created
Content-Location:/server/sensor/data
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
      "faultDetectionPolicy": {
                     "threshold" :{
                                "maxValue": 100,
                                "minValue: 10
                     },
                     "notificationURI": "/server/monitor"
            }
}
```

Create Content Instance by sensor AE inside data Container

HTTP Request

```
POST /server/sensor/data HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-sensor
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cin": {
        "cnf": "application/text",
            "con": "200"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/sensor/data
Content-Type: application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
      "m2m:cin": {
           "rn": "cin_50288226",
        "ty": 4,
        "ri": "/server/cin-50288226",
        "pi": "/server/cnt-211484110",
        "ct": "20200223T205943",
        "lt": "20200223T205943",
        "st": 0,
        "cnf": "application/text",
        "cs": 2,
        "con":"200"
       }
}
```

Monitor AE receives fault detection alert from data published by sensor AE inside data Container

HTTP Request

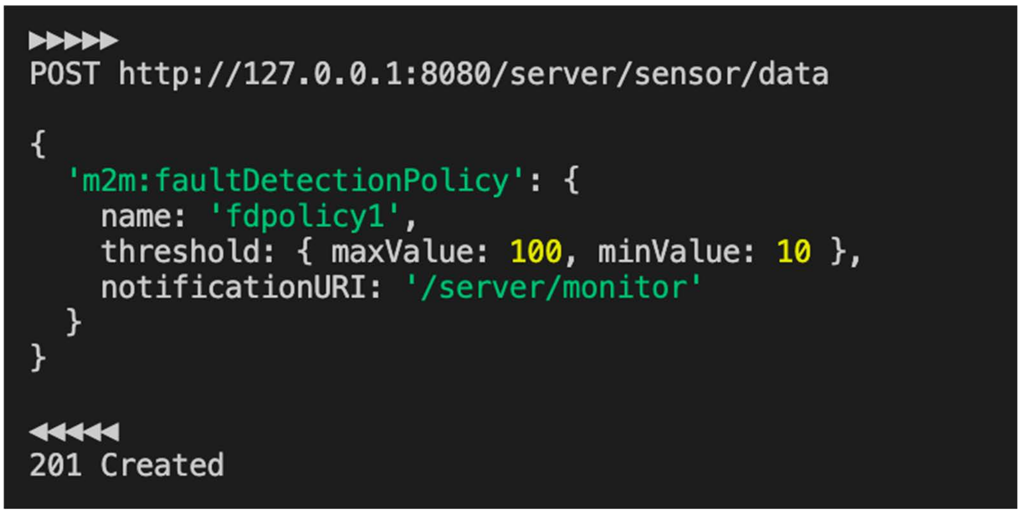
`POST / HTTP/1.1`
`Host:1270.0.1:4000`
`X-M2M-Origin: /server`
`Content-Type: application/json`
 	
`{`
`"m2m:sgn": {`
`    "nev": {`
`        "rep": {`
`            "m2m:cin": {`
`              "con": "200",`
`              "cnf": "text/plain:0"`
`            },`
`            "rss": "201"`
`          }`
`   },`
`   "sur": "/server/fdp-164504268"`
`}`
`}`

HTTP Response

`204 No content`
`X-M2M-Origin:/server`
`X-M2M-RSC:2004`

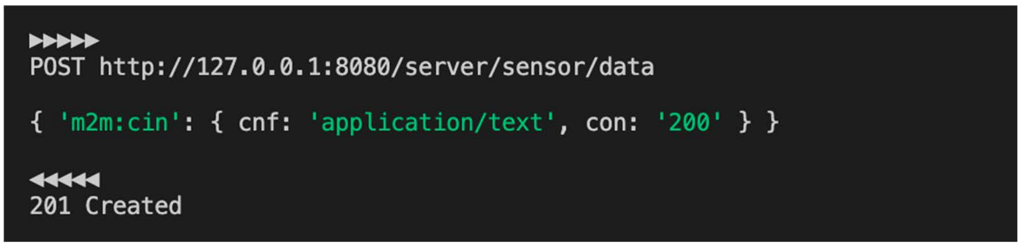
6.1.5 Demonstration

The Monitor AE creates the fault detection based on threshold policy on data container of sensor AE:



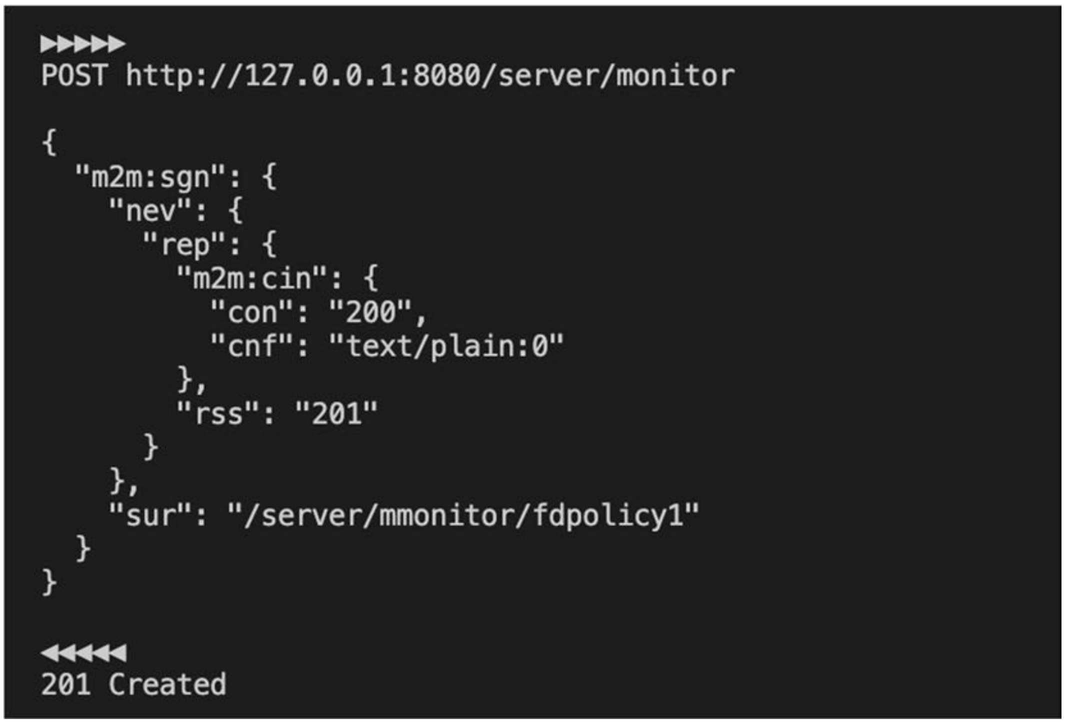
```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/sensor/data

{
  'm2m:faultDetectionPolicy': {
    name: 'fdpolicy1',
    threshold: { maxValue: 100, minValue: 10 },
    notificationURI: '/server/monitor'
  }
}

◀◀◀◀◀
201 Created
```

The Sensor AE publishes a measurement on data Container:



```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/sensor/data

{ 'm2m:cin': { cnf: 'application/text', con: '200' } }

◀◀◀◀◀
201 Created
```

The CSE detects faulty measurement using fault detection CSF and notify monitor AE accordingly:



```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/monitor

{
  "m2m:sgn": {
    "nev": {
      "rep": {
        "m2m:cin": {
          "con": "200",
          "cnf": "text/plain:0"
        },
        "rss": "201"
      }
    },
    "sur": "/server/mmonitor/fdpolicy1"
  }
}

◀◀◀◀◀
201 Created
```

6.2 PoC UC2: Detection of patterns in video streams

6.2.1 Architecture

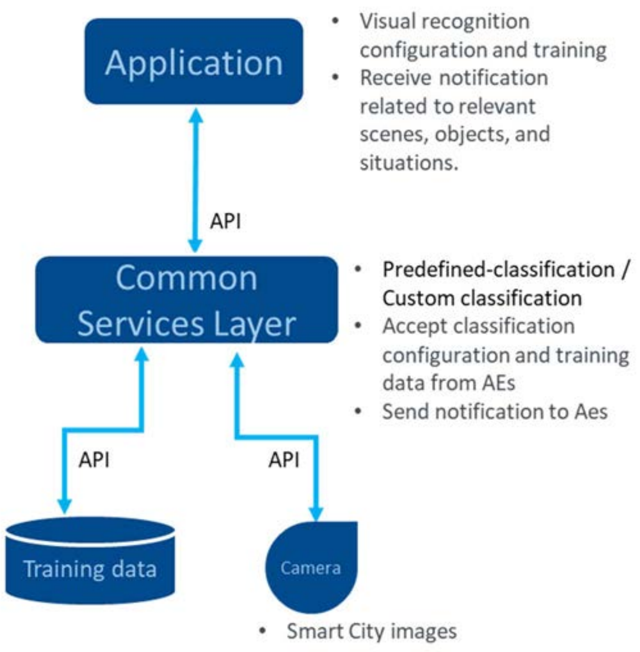

Application
• Visual recognition configuration and training
• Receive notification related to relevant scenes, objects, and situations.
API
Common Services Layer
• Predefined-classification / Custom classification
• Accept classification configuration and training data from AEs
• Send notification to Aes
API
API
Training data
Camera
• Smart City images

Figure 4: High-Level Architecture of Use Case 2

6.2.2 Resources and attributes

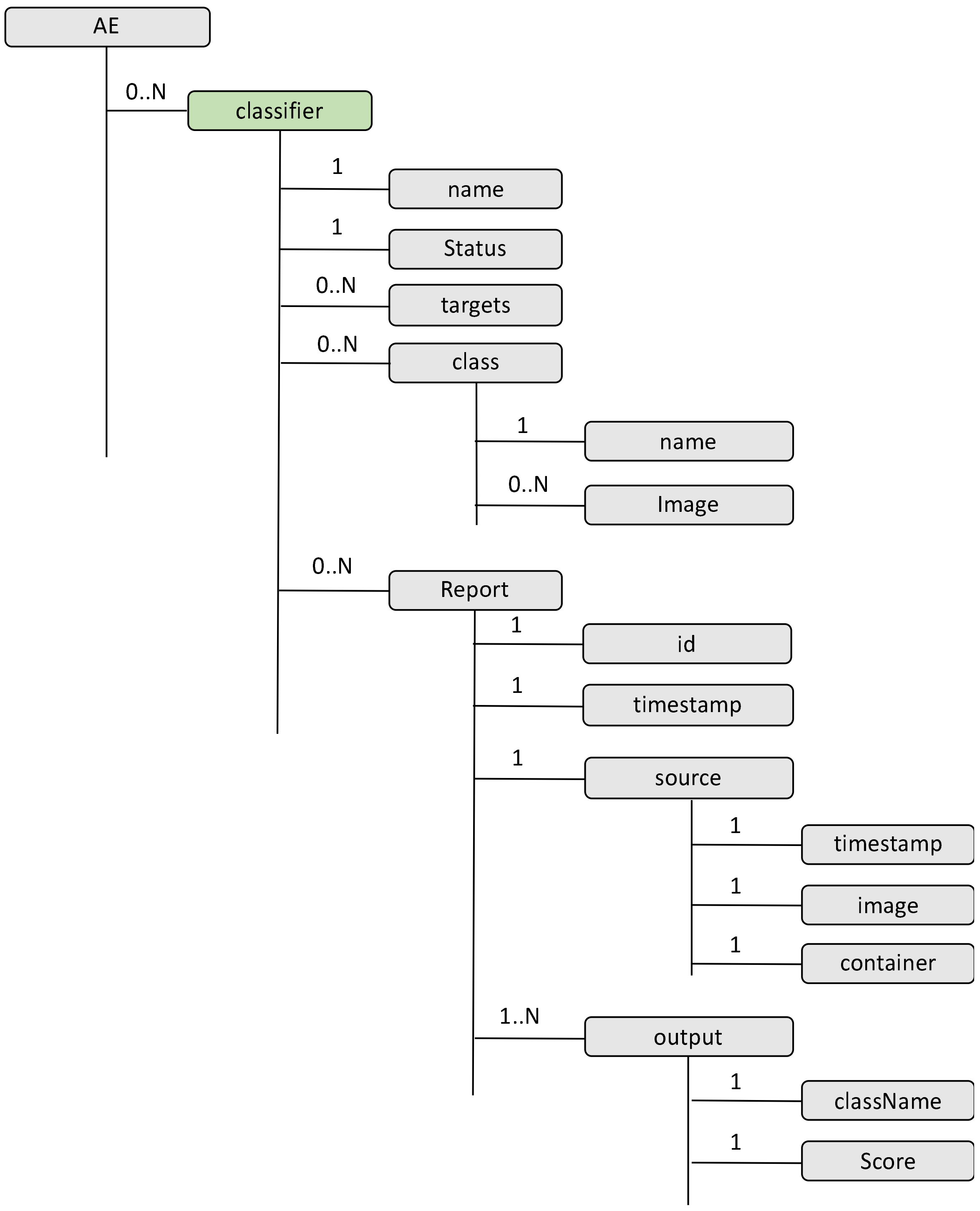


Figure 5

6.2.3 Message flow

Figure 6 depicts the entities and main components involved in this use case and the sequence of messages exchanged between them to carry out the functionality need for detection of patterns in video streams in oneM2M platform. The blue boxes show the interactions that are achievable with oneM2M platform without any changes, while the green boxes show new interactions that do not currently exist in oneM2M and need to be specified and implemented for this use case.

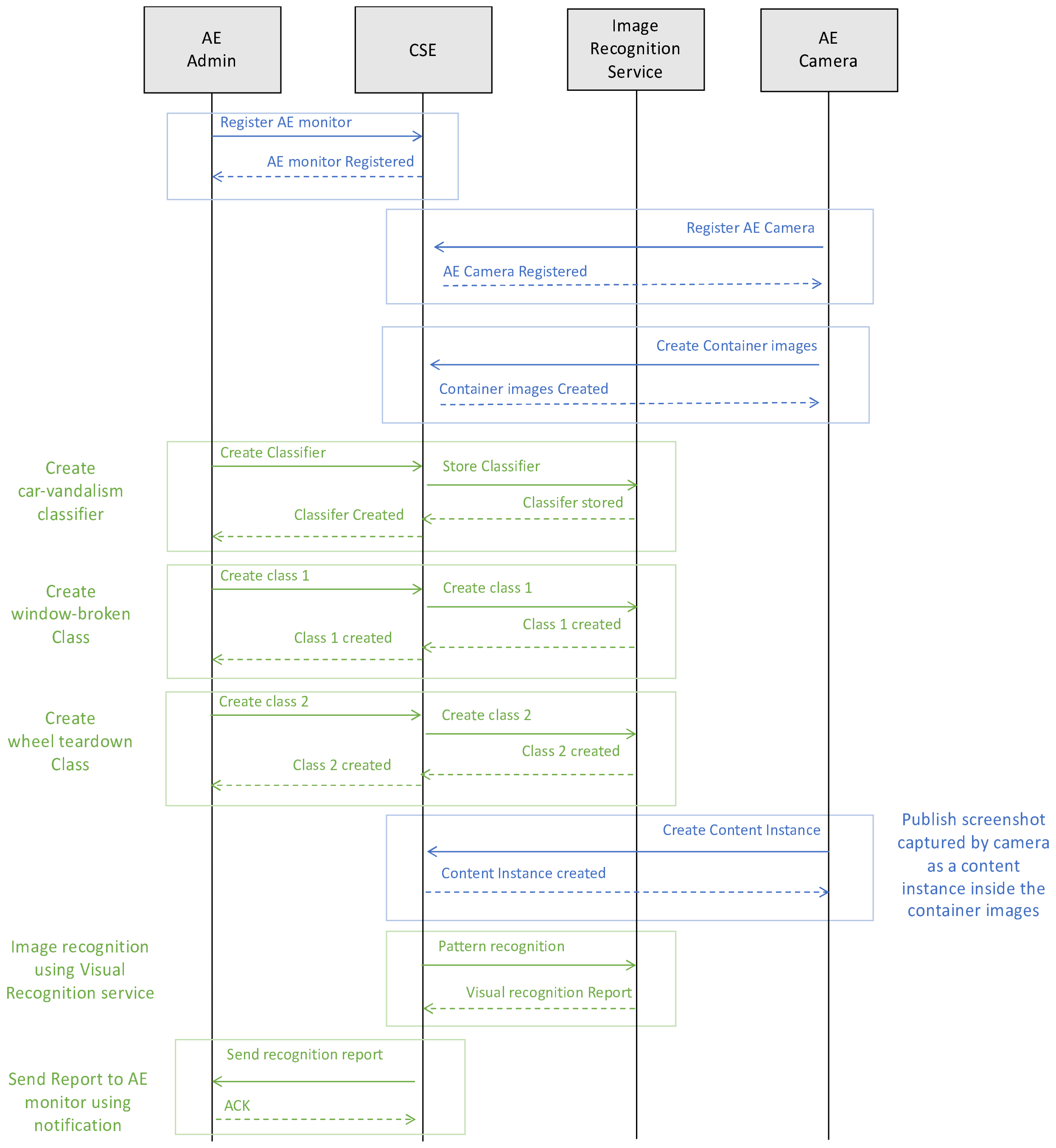


Figure 6: Sequence diagram of Use Case 2

6.2.4 Implementation

Register AE admin

HTTP Request

```
POST /server HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin : Cae-monitor
Content-Type: application/json;ty=2
Accept: application/json

{
    "m2m:ae": {
        "rn": "admin",
        "rr": true,
             "nu":["http://127.0.0.1:4000"]
        "api": "company.org"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/admin
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:ae": {
        "rn": "admin",
        "ty": 2,
        "ri": "/CAE988694263",
        "pi": "/server",
        "ct": "20200223T194209",
        "lt": "20200223T194209",
        "acpi": ["/server/acp-69998272"],
        "et": "20180223T194209",
        "api": "company.org",
        "aei": "Cae-monitor",
        "rr": true,
             "nu":["http://127.0.0.1:4000"]
    }
}
```

Register AE Camera

HTTP Request

```
POST /server HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin : Cae-camera
Content-Type: application/json;ty=2
Accept: application/json

{
    "m2m:ae": {
        "rn": "camera",
        "rr": false,
        "api": "company.org"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/camera
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:ae": {
        "rn": "camera",
        "ty": 2,
        "ri": "/CAE988694263",
        "pi": "/server",
        "ct": "20200223T194209",
        "lt": "20200223T194209",
        "acpi": ["/server/acp-69998272"],
        "et": "20180223T194209",
        "api": "company.org",
        "aei": "Cae-camera",
        "rr": false
    }
}
```

Create Container "images" inside camera AE

HTTP Request

```
POST /server/sensor HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-images
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cnt": {
        "rn": "images",
        "mni": 100
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/camera/images
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:cnt": {
        "rn": "images",
        "ty": 3,
        "ri": "/server/cnt-364492354",
        "pi": "/server/CAE267526506",
        "ct": "20200223T200338",
        "lt": "20200223T200338",
        "acpi": ["/server/acp-940743520"],
        "et": "20180223T200338",
        "st": 0,
        "mni": 100,
        "mbs": 10000,
        "mia": 0,
        "cni": 0,
        "cbs": 0
    }
}
```

Create Classifier car-vandalism

HTTP Request

```
POST /server/monitor HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-monitor
Content-Type: application/json;ty=XX
Accept: application/json

{
    "m2m:classifier": {
        "name": "car-vandalism",
           "status" : "active",
           "targets" : ["/server/camera/images"]
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/monitor/car-vandalism
Content-Type:application/json
X-M2M-Origin:/car-vandalism
X-M2M-RSC:2001

{
    "m2m:classifer": {
        "name": "car-vandalism",
           "status" : "active",
           "targets" : ["/server/camera/images"]
    }
}
```

Create window-broken class inside car-vandalism Classifier

HTTP Request

```
POST /server/monitor/car-vandalism HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-monitor
Content-Type: application/json;ty=XX
Accept: application/json

{
    "m2m:class": {
        "rn": "window-broken ",
           "images":["base64image1"," base64image2"," base64imageN"],
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/monitor/car-vandalism/window-broken
Content-Type:application/json
X-M2M-Origin:/car-vandalism
X-M2M-RSC:2001

{
    "m2m:classifer": {
        "name": "car-vandalism",
           "images":["base64image1"," base64image2"," base64imageN"]
    }
}
```

Create wheel-teardown class inside car-vandalism Classifier

HTTP Request

```
POST /server/monitor/car-vandalism HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-monitor
Content-Type: application/json;ty=XX
Accept: application/json

{
    "m2m:class": {
        "rn": "wheel-teardown",
           "images":["base64image1"," base64image2"," base64imageN"]
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/car-vandalism/wheel-teardown
Content-Type:application/json
X-M2M-Origin:/ae-admin
X-M2M-RSC:2001

{
    "m2m:class": {
        "rn": "wheel-teardown ",
           "images":["base64image1"," base64image2"," base64imageN"]
    }
}
```

Publish a car image taken by camera AE as contentInstance inside "images" container

HTTP Request

```
POST /server/camera/images HTTP/1.1
Host: http://127.0.0.1:8080
X-M2M-Origin: Cae-camera
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cin": {
        "cnf": "application/text",
           "con": "base64iamge"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/camera/images/cin_50288226
Content-Type: application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
      "m2m:cin": {
           "rn": "cin_50288226",
        "ty": 4,
        "ri": "/server/cin-50288226",
        "pi": "/server/cnt-211484110",
        "ct": "20200223T205943",
        "lt": "20200223T205943",
        "st": 0,
        "cnf": "application/text",
        "cs": 2,
        "con": base64iamge "
       }
}
```

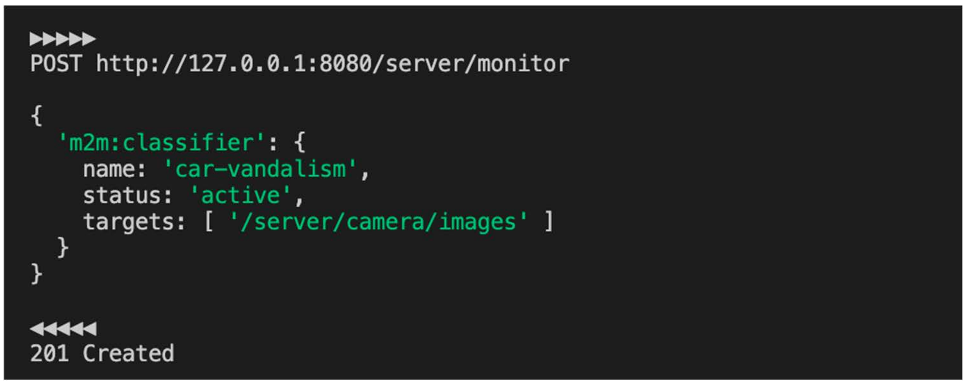
CSE creates a visual recognition report related to image published by the camera

HTTP Request
```
POST  /server/monitor/car-vandalism HTTP/1.1
Host:1270.0.1:8080
X-M2M-Origin: /server
Content-Type: application/json

"m2m:report" : {
       "id" : "rep123456",
       "output": [
             {
                 "className": "window-broken",
                 "score": 0.92,
              },
              {
                 "className": "wheel-teardown",
                 "score": 0.13
              }
       ],
       "source" : {
                "timestamp": "1538239798659",
                "image": "base64image",
                "container" : "/server/camera/images »
        }
}
```

HTTP Response
```
201 Created
X-M2M-Origin:/server/monitor/car-vandalism
X-M2M-RSC:2001

"m2m:report" : {
       "output": [
             {
                 "className": "window-broken",
                 "score": 0.92,
              },
              {
                 "className": "wheel-teardown",
                 "score": 0.13
              }
       ],
       "source" : {
                "timestamp": "1538239798659",
                "image": "base64image"
        }
}
```

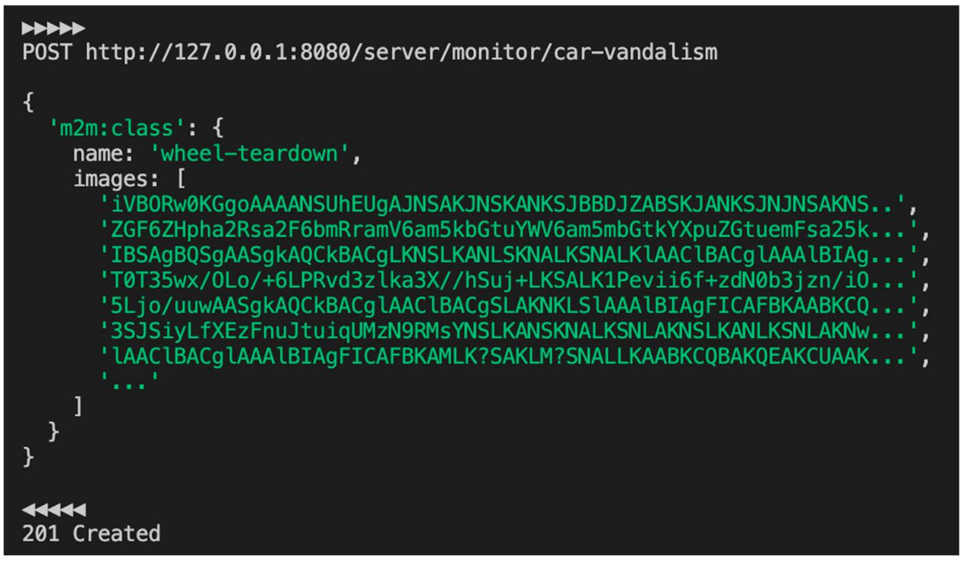
6.2.5 Demonstration

The Monitor AE creates a car-vandalism classifier targeting the images container of the camera AE:



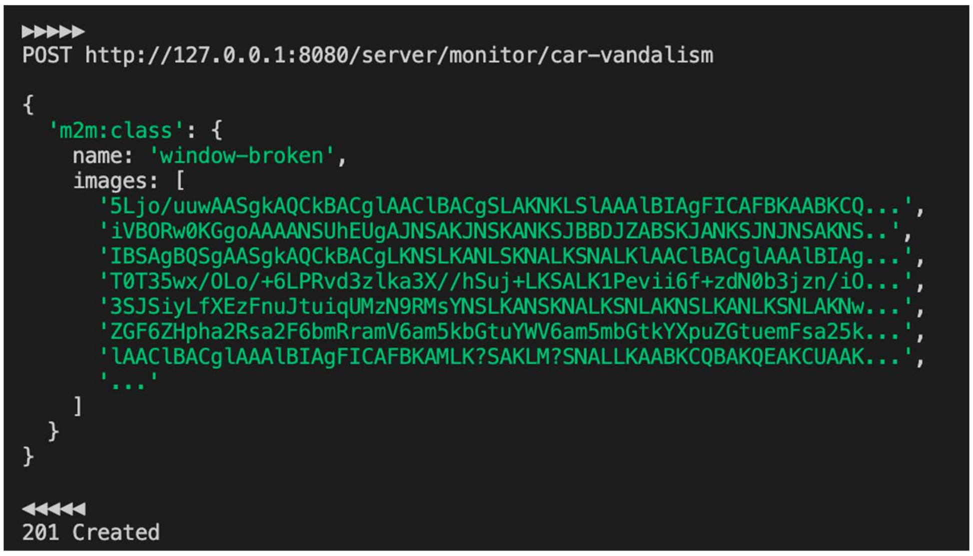
```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/monitor

{
  'm2m:classifier': {
    name: 'car-vandalism',
    status: 'active',
    targets: [ '/server/camera/images' ]
  }
}

◀◀◀◀◀
201 Created
```

The Monitor AE creates a window-broken class inside the car-vandalism classifier including a set of images of cars with broken windows:



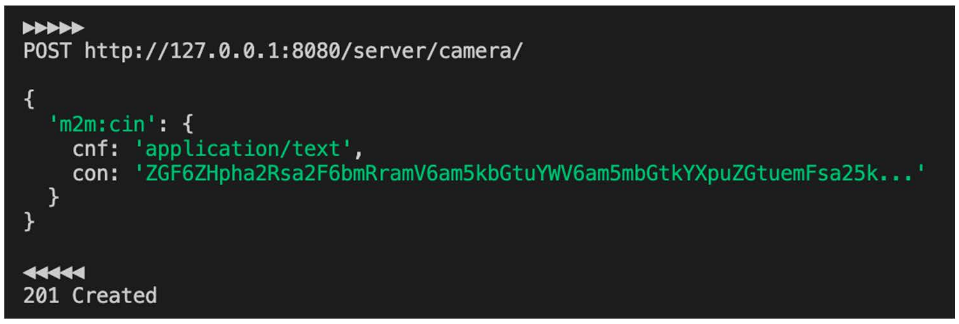
```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/monitor/car-vandalism

{
  'm2m:class': {
    name: 'wheel-teardown',
    images: [
      'iVBORw0KGgoAAAANSUhEUgAJNSAKJNSKANKSJBBDJZABSKJANKSJNJNSAKNS..',
      'ZGF6ZHpha2Rsa2F6bmRramV6am5kbGtuYWV6am5mbGtkYXpuZGtuemFsa25k...',
      'IBSAgBQSgAASgkAQCkBACgLKNSLKANLSKNALKSNALKlAAClBACglAAAlBIAg...',
      'T0T35wx/OLo/+6LPRvd3zlka3X//hSuj+LKSALK1Pevii6f+zdN0b3jzn/iO...',
      '5Ljo/uuwAASgkAQCkBACglAAClBACgSLAKNKLSlAAAlBIAgFICAFBKAABKCQ...',
      '3SJSiyLfXEzFnuJtuiqUMzN9RMsYNSLKANSKNALKSNLAKNSLKANLKSNLAKNw...',
      'lAAClBACglAAAlBIAgFICAFBKAMLK?SAKLM?SNALLKAABKCQBAKQEAKCUAAK...',
      '...'
    ]
  }
}

◀◀◀◀◀
201 Created
```

The Monitor AE creates a wheel teardown class inside the car-vandalism classifier including a set of images of cars with teardown wheels;



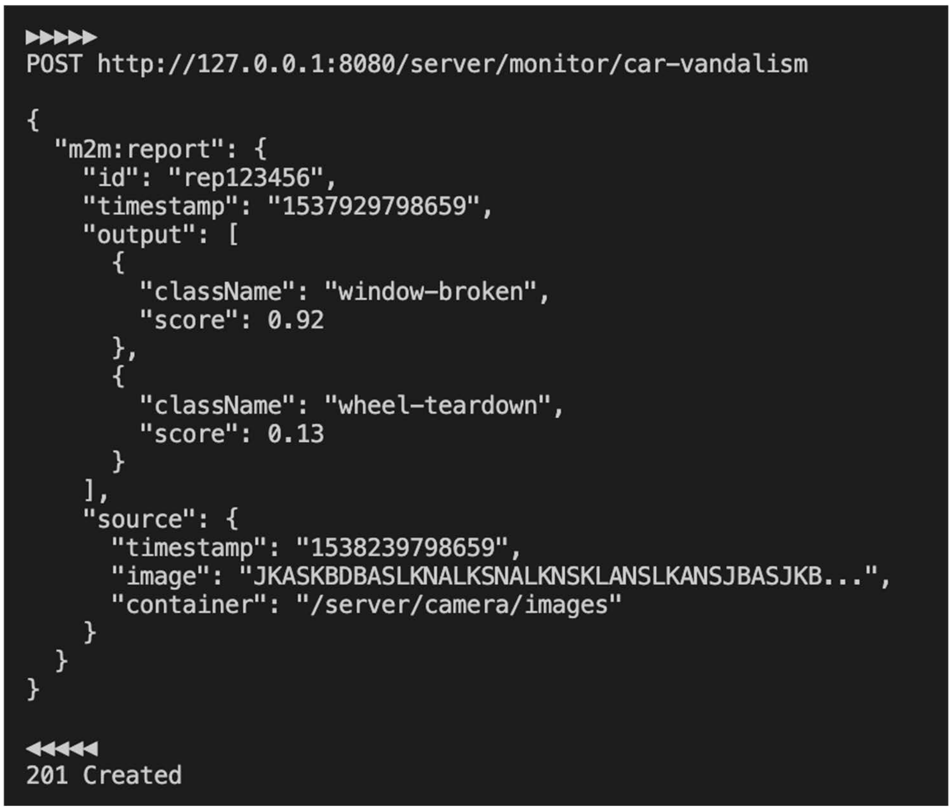
```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/monitor/car-vandalism

{
  'm2m:class': {
    name: 'window-broken',
    images: [
      '5Ljo/uuwAASgkAQCkBACglAAClBACgSLAKNKLSlAAAlBIAgFICAFBKAABKCQ...',
      'iVBORw0KGgoAAAANSUhEUgAJNSAKJNSKANKSJBBDJZABSKJANKSJNJNSAKNS..',
      'IBSAgBQSgAASgkAQCkBACgLKNSLKANLSKNALKSNALKlAAClBACglAAAlBIAg...',
      'T0T35wx/OLo/+6LPRvd3zlka3X//hSuj+LKSALK1Pevii6f+zdN0b3jzn/iO...',
      '3SJSiyLfXEzFnuJtuiqUMzN9RMsYNSLKANSKNALKSNLAKNSLKANLKSNLAKNw...',
      'ZGF6ZHpha2Rsa2F6bmRramV6am5kbGtuYWV6am5mbGtkYXpuZGtuemFsa25k...',
      'lAAClBACglAAAlBIAgFICAFBKAMLK?SAKLM?SNALLKAABKCQBAKQEAKCUAAK...',
      '...'
    ]
  }
}

◀◀◀◀◀
201 Created
```

The AE camera published an image in the its images container:



```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/camera/

{
  'm2m:cin': {
    cnf: 'application/text',
    con: 'ZGF6ZHpha2Rsa2F6bmRramV6am5kbGtuYWV6am5mbGtkYXpuZGtuemFsa25k...'
  }
}

◀◀◀◀◀
201 Created
```

The CSE performs visual recognition based on the car-vandalism classifier and create the corresponding report including the scores for window-broken and wheel teardown classes:



```
▶▶▶▶▶
POST http://127.0.0.1:8080/server/monitor/car-vandalism

{
  "m2m:report": {
    "id": "rep123456",
    "timestamp": "1537929798659",
    "output": [
      {
        "className": "window-broken",
        "score": 0.92
      },
      {
        "className": "wheel-teardown",
        "score": 0.13
      }
    ],
    "source": {
      "timestamp": "1538239798659",
      "image": "JKASKBDBASLKNALKSNALKNSKLANSLKANSJBASJKB...",
      "container": "/server/camera/images"
    }
  }
}

◀◀◀◀◀
201 Created
```

6.3 PoC UC3: Occurrences Classification

6.3.1 Architecture

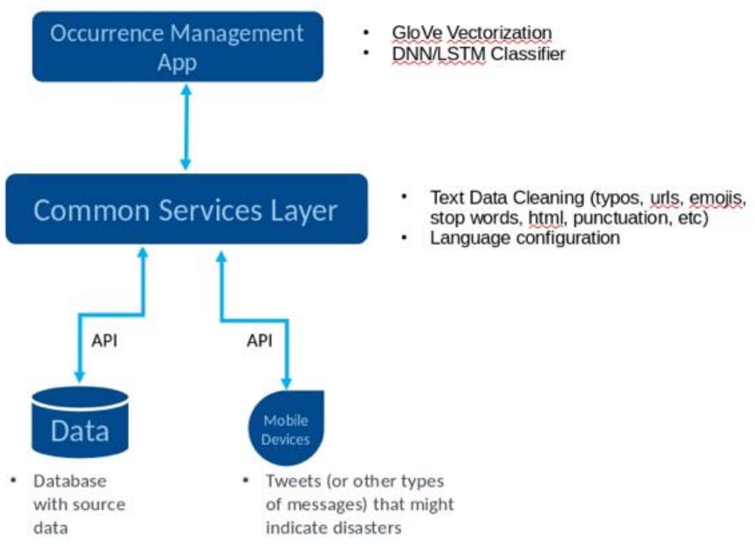

Occurrence Management App
• GloVe Vectorization
• DNN/LSTM Classifier
Common Services Layer
• Text Data Cleaning (typos, urls, emojis, stop words, html, punctuation, etc)
• Language configuration
API
API
Data
Mobile Devices
• Database with source data
• Tweets (or other types of messages) that might indicate disasters

Figure 7: High-Level Architecture of Use Case 3

The work in this use case is based on the existing open-source OpenMTC implementation of oneM2M. Data is obtained from a tweet dataset (in this particular case, using an open Kaggle dataset [i.2]) to train the model and from a mobile app where citizens can report occurrences and other incidents within the city. This text data is cleaned by a custom-made Common Service Function before being fed to the application entity where the classification models reside. When an incoming occurrence is classified as reporting a disaster, an alert is generated.

The Common Service Function to be developed uses popular Python libraries to do these text data cleaning processes (e.g. sklearn, nltk) which include fixing typos, removing stop words, punctuation, HTTP tags and URLs. It also provides the ability to be configured for a certain language, enabling these functionalities to a broader audience.

The trained classifier obtains the cleaned data and use word embeddings (such as GloVe [i.3]) to vectorize the incoming text and provide it to a Deep Neural Network classifier based on Long-Short Term Memory architecture.

6.3.2 Resources and attributes

The resources and attributes of the use case for text cleaning and occurrences classification are presented in figure 8.

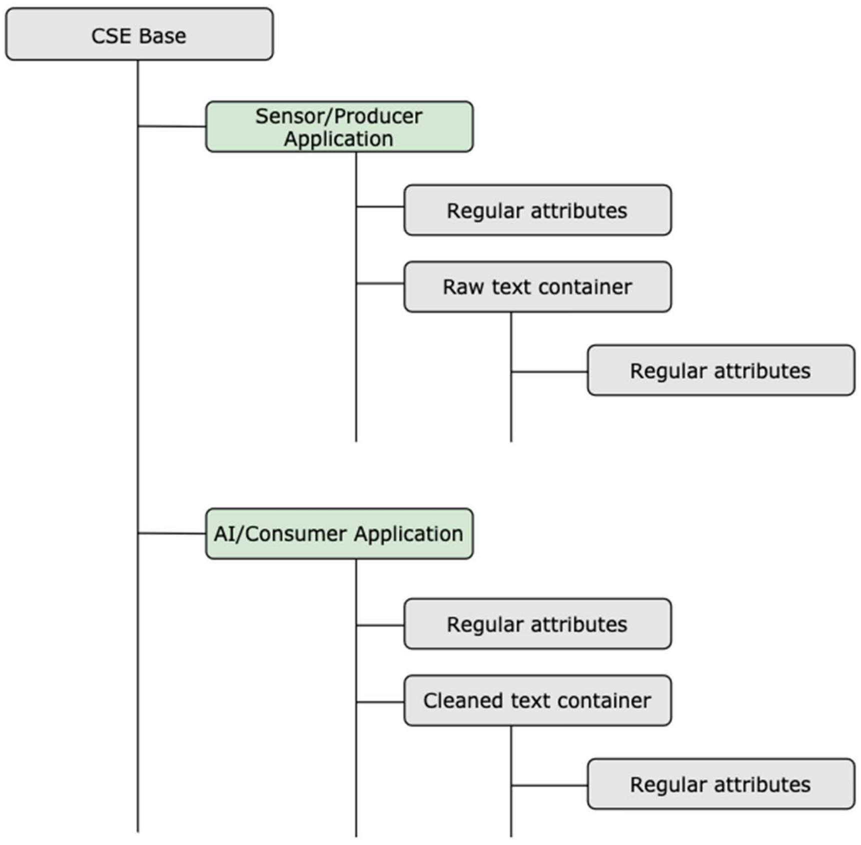


Figure 8

6.3.3 Message flow

Figure 9 represents a sequence diagram of the use case. The first two messages represent the instantiation of the Text Cleaning CSF. The Natural Language Processing (NLP) App, through OpenMTC's Connectivity Manager, provides the necessary configurations, including the language it is supposed to work with. Through this, the Text Cleaning CSF can expect incoming data and treat it accordingly.

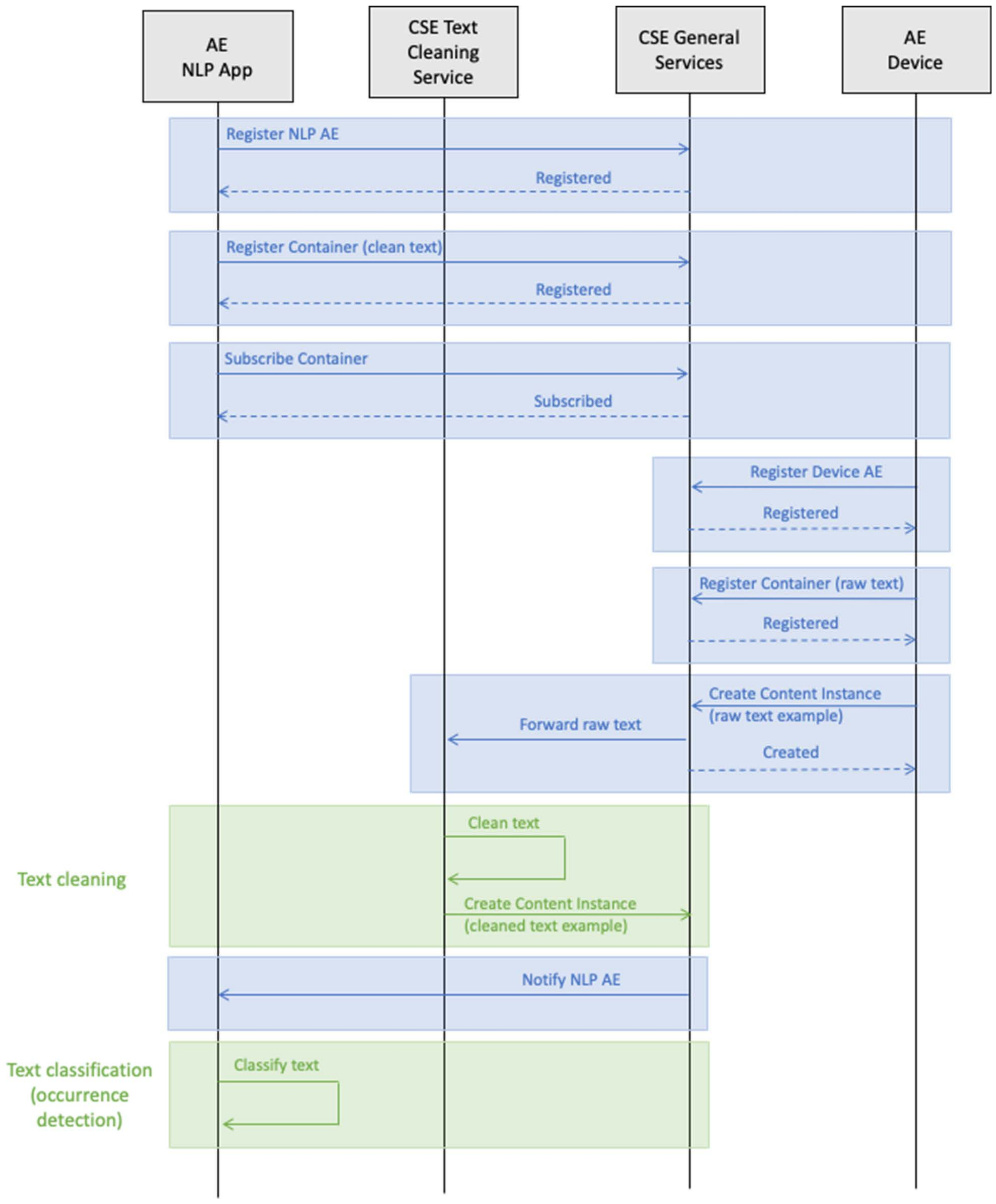


Figure 9: Sequence Diagram for Use Case 3

After that, the main flow of data is represented, where, through an application, citizens can provide written information regarding occurrences in their area, which are sent to OpenMTC, either through HTTP or MQTT. Within OpenMTC, the data is automatically processed by the Text Cleaning CSF and provided to the NLP App, where the classification algorithm processes the incoming data to determine if citizens are reporting disasters or not.

6.3.4 Implementation

Register NLP AE

HTTP Request

```
POST /server HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin : Cae-nlp_ae
Content-Type: application/json;ty=2
Accept: application/json

{
    "m2m:ae": {
        "rn": "nlp_ae",
        "rr": true,
                           "nu":["http://172.17.0.1:20371"]
        "api": "company.org"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/nlp_ae
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:ae": {
        "rn": "nlp_ae",
        "ty": 2,
        "ri": "/CAE988694263",
        "pi": "/server",
        "ct": "20200223T194209",
        "lt": "20200223T194209",
        "acpi": ["/server/acp-69998272"],
        "et": "20180223T194209",
        "api": "company.org",
        "aei": "Cae-nlp_ae",
        "rr": true,
                          "nu":["http://172.17.0.1:20371"]
    }
}
```

Register Device AE

HTTP Request

```
POST /server HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin : Cae-device
Content-Type: application/json;ty=2
Accept: application/json

{
    "m2m:ae": {
        "rn": "device",
        "rr": false,
        "api": "company.org"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/device
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:ae": {
        "rn": "device",
        "ty": 2,
        "ri": "/CAE988694263",
        "pi": "/server",
        "ct": "20200223T194209",
        "lt": "20200223T194209",
        "acpi": ["/server/acp-69998272"],
        "et": "20180223T194209",
        "api": "company.org",
        "aei": "Cae-device",
        "rr": false
    }
}
```

Create data_raw Container inside Device AE

HTTP Request

```
POST /server/device HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin: Cae-device
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cnt": {
        "rn": "data_raw",
                     "mni": 100
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/device/data_raw
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:cnt": {
        "rn": "data_raw",
        "ty": 3,
        "ri": "/server/cnt-364492354",
        "pi": "/server/CAE267526506",
        "ct": "20200223T200338",
        "lt": "20200223T200338",
        "acpi": ["/server/acp-940743520"],
        "et": "20180223T200338",
        "st": 0,
        "mni": 100,
        "mbs": 10000,
        "mia": 0,
        "cni": 0,
        "cbs": 0
    }
}
```

Create data_cleaned Container inside NLP AE

HTTP Request

```
POST /server/nlp_ae HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin: Cae-nlp_ae
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cnt": {
        "rn": "data_cleaned",
                     "mni": 100
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/nlp_ae/data_cleaned
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
    "m2m:cnt": {
        "rn": "data_cleaned",
        "ty": 3,
        "ri": "/server/cnt-364492354",
        "pi": "/server/CAE267526506",
        "ct": "20200223T200338",
        "lt": "20200223T200338",
        "acpi": ["/server/acp-940743520"],
        "et": "20180223T200338",
        "st": 0,
        "mni": 100,
        "mbs": 10000,
        "mia": 0,
        "cni": 0,
        "cbs": 0
    }
}
```

Subscribe the data_cleaned Container inside NLP AE

HTTP Request

```
POST /server/nlp_ae/data_cleaned HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin: Cae-nlp_ae
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:sub": {
        "rn": "data_cleaned",
                     "enc": {
                                   "net": 3
                           },
                           "nu": ["Cae_consumer"],
                           "nct": 1
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/sub-856593979
Content-Type:application/json
X-M2M-Origin:/server
X-M2M-RSC:2001
```

Create Content Instance by Device AE inside data_raw Container

HTTP Request

```
POST /server/device/data_raw HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin: Cae-device
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cin": {
        "cnf": "application/text",
                          "con": "{"text": "there is a fire", "target_label": "label"}"
    }
}
```

HTTP Response

```
201 Created
Content-Location:/server/device/data_raw
Content-Type: application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
      "m2m:cin": {
           "rn": "cin_50288226",
        "ty": 4,
        "ri": "/server/cin-50288226",
        "pi": "/server/cnt-211484110",
        "ct": "20200223T205943",
        "lt": "20200223T205943",
        "st": 0,
        "cnf": "application/text",
        "cs": 2,
        "con":"{"text": "there is a fire", "target_label": "label"}"
       }
}
```

Create Content Instance by Clean Text CSF inside data_cleaned Container

HTTP Request

```
POST /server/nlp_ae/data_cleaned HTTP/1.1
Host: http://127.0.0.1:8000
X-M2M-Origin: Cae-nlp_ae
Content-Type: application/json;ty=3
Accept: application/json

{
    "m2m:cin": {
        "cnf": "application/text",
                          "con": "{"text": "data", "cleaned_text": ["there", "fire"]}"
    }
}
```

HTTP Response
```
201 Created
Content-Location:/server/nlp_ae/data_cleaned
Content-Type: application/json
X-M2M-Origin:/server
X-M2M-RSC:2001

{
      "m2m:cin": {
           "rn": "cin_50288226",
        "ty": 4,
        "ri": "/server/cin-50288226",
        "pi": "/server/cnt-211484110",
        "ct": "20200223T205943",
        "lt": "20200223T205943",
        "st": 0,
        "cnf": "application/text",
        "cs": 2,
        "con":"{"text": "data", "cleaned_text": ["there", "fire"]}"
       }
}
```

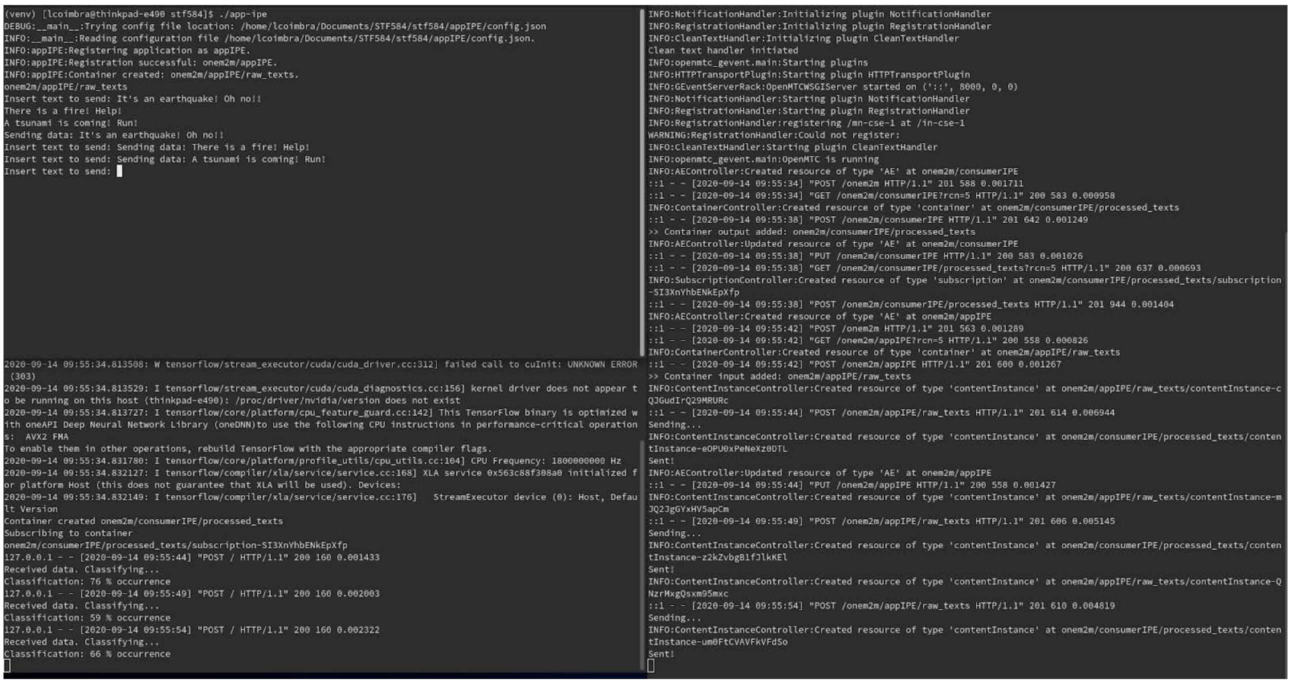
NLP AE receives alert from data published by the CSF inside data_cleaned Container

HTTP Request
```
POST / HTTP/1.1
Host:1270.0.1:8000
X-M2M-Origin: /server
Content-Type: application/json

{
"m2m:sgn": {
    "nev": {
        "rep": {
            "m2m:cin": {
              "con": "{"text": "data", "cleaned_text": ["there", "fire"]}",
              "cnf": "text/plain:0"
            },
            "rss": "201"
          }
   },
   "sur": "/server/fdp-164504268"
}
}
```

HTTP Response
```
204 No content
X-M2M-Origin:/server
X-M2M-RSC:2004
```

6.3.5 Demonstration

In figure 10, the full system is demonstrated (both AEs and the CSE).



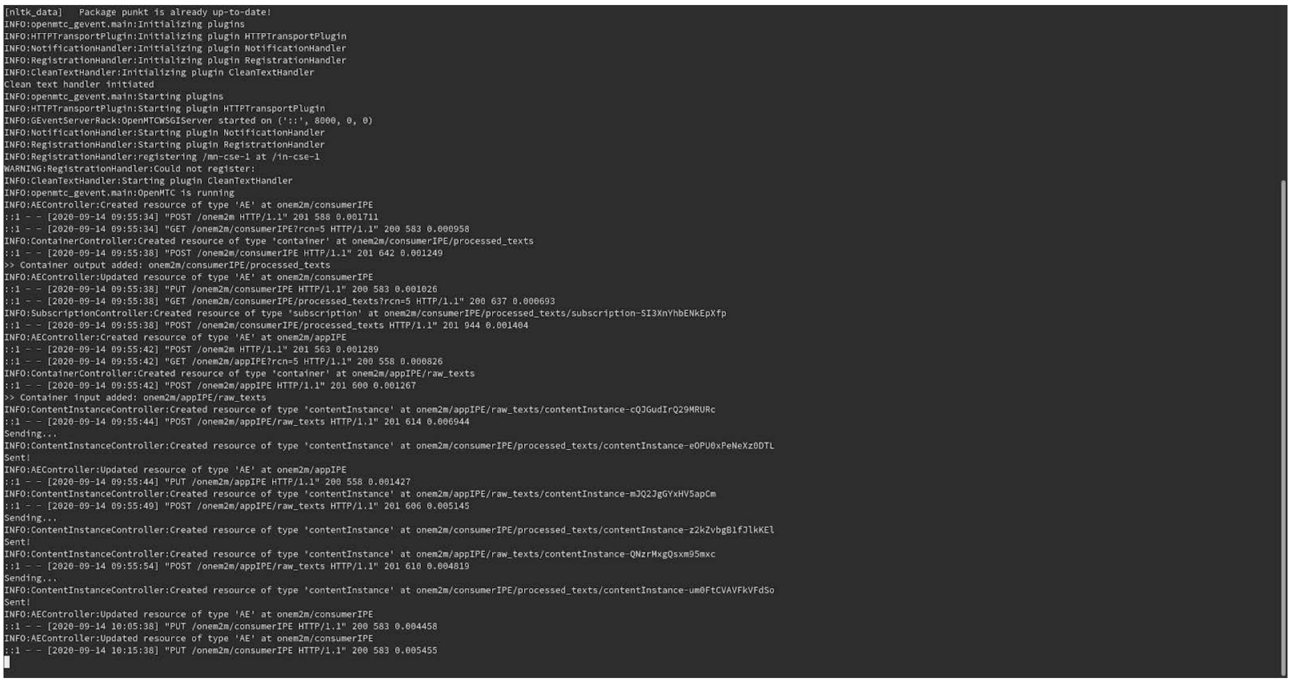
```
(venv) [lcoimbra@thinkpad-e490 stf584]$ ./app-ipe
DEBUG:__main__:Trying config file location: /home/lcoimbra/Documents/STF584/stf584/appIPE/config.json
INFO:__main__:Reading configuration file /home/lcoimbra/Documents/STF584/stf584/appIPE/config.json.
INFO:appIPE:Registering application as appIPE.
INFO:appIPE:Registration successful: onem2m/appIPE.
INFO:appIPE:Container created: onem2m/appIPE/raw_texts.
onem2m/appIPE/raw_texts
Insert text to send: It's an earthquake! Oh no!!
There is a fire! Help!
A tsunami is coming! Run!
Sending data: It's an earthquake! Oh no!!
Insert text to send: Sending data: There is a fire! Help!
Insert text to send: Sending data: A tsunami is coming! Run!
Insert text to send: 
```

```
2020-09-14 09:55:34.813508: W tensorflow/stream_executor/cuda/cuda_driver.cc:312] failed call to cuInit: UNKNOWN ERROR
 (303)
2020-09-14 09:55:34.813529: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear t
o be running on this host (thinkpad-e490): /proc/driver/nvidia/version does not exist
2020-09-14 09:55:34.813727: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized w
ith oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical operation
s:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2020-09-14 09:55:34.831780: I tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency: 1800000000 Hz
2020-09-14 09:55:34.832127: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x563c88f308a0 initialized f
or platform Host (this does not guarantee that XLA will be used). Devices:
2020-09-14 09:55:34.832149: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Defau
lt Version
Container created onem2m/consumerIPE/processed_texts
Subscribing to container
onem2m/consumerIPE/processed_texts/subscription-SI3XnYhbENkEpXfp
127.0.0.1 - - [2020-09-14 09:55:44] "POST / HTTP/1.1" 200 160 0.001433
Received data. Classifying...
Classification: 76 % occurrence
127.0.0.1 - - [2020-09-14 09:55:49] "POST / HTTP/1.1" 200 160 0.002003
Received data. Classifying...
Classification: 59 % occurrence
127.0.0.1 - - [2020-09-14 09:55:54] "POST / HTTP/1.1" 200 160 0.002322
Received data. Classifying...
Classification: 66 % occurrence
```

```
INFO:NotificationHandler:Initializing plugin NotificationHandler
INFO:RegistrationHandler:Initializing plugin RegistrationHandler
INFO:CleanTextHandler:Initializing plugin CleanTextHandler
Clean text handler initiated
INFO:openmtc_gevent.main:Starting plugins
INFO:HTTPTransportPlugin:Starting plugin HTTPTransportPlugin
INFO:GEventServerRack:OpenMTCWSGIServer started on ('::', 8000, 0, 0)
INFO:NotificationHandler:Starting plugin NotificationHandler
INFO:RegistrationHandler:Starting plugin RegistrationHandler
INFO:RegistrationHandler:registering /mn-cse-1 at /in-cse-1
WARNING:RegistrationHandler:Could not register:
INFO:CleanTextHandler:Starting plugin CleanTextHandler
INFO:openmtc_gevent.main:OpenMTC is running
INFO:AEController:Created resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 09:55:34] "POST /onem2m HTTP/1.1" 201 588 0.001711
::1 - - [2020-09-14 09:55:34] "GET /onem2m/consumerIPE?rcn=5 HTTP/1.1" 200 583 0.000958
INFO:ContainerController:Created resource of type 'container' at onem2m/consumerIPE/processed_texts
::1 - - [2020-09-14 09:55:38] "POST /onem2m/consumerIPE HTTP/1.1" 201 642 0.001249
>> Container output added: onem2m/consumerIPE/processed_texts
INFO:AEController:Updated resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 09:55:38] "PUT /onem2m/consumerIPE HTTP/1.1" 200 583 0.001026
::1 - - [2020-09-14 09:55:38] "GET /onem2m/consumerIPE/processed_texts?rcn=5 HTTP/1.1" 200 637 0.000693
INFO:SubscriptionController:Created resource of type 'subscription' at onem2m/consumerIPE/processed_texts/subscription
-SI3XnYhbENkEpXfp
::1 - - [2020-09-14 09:55:38] "POST /onem2m/consumerIPE/processed_texts HTTP/1.1" 201 944 0.001404
INFO:AEController:Created resource of type 'AE' at onem2m/appIPE
::1 - - [2020-09-14 09:55:42] "POST /onem2m HTTP/1.1" 201 563 0.001289
::1 - - [2020-09-14 09:55:42] "GET /onem2m/appIPE?rcn=5 HTTP/1.1" 200 558 0.000826
INFO:ContainerController:Created resource of type 'container' at onem2m/appIPE/raw_texts
::1 - - [2020-09-14 09:55:42] "POST /onem2m/appIPE HTTP/1.1" 201 600 0.001267
>> Container input added: onem2m/appIPE/raw_texts
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-c
QJGudIrQ29MRURc
::1 - - [2020-09-14 09:55:44] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 614 0.006944
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/conten
tInstance-eOPU0xPeNeXz0DTL
Sent!
INFO:AEController:Updated resource of type 'AE' at onem2m/appIPE
::1 - - [2020-09-14 09:55:44] "PUT /onem2m/appIPE HTTP/1.1" 200 558 0.001427
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-m
JQ2JgGYxHV5apCm
::1 - - [2020-09-14 09:55:49] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 606 0.005145
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/conten
tInstance-z2kZvbgB1fJlkKEl
Sent!
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-Q
NzrMxgQsxm95mxc
::1 - - [2020-09-14 09:55:54] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 610 0.004819
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/conten
tInstance-um0FtCVAVFkVFdSo
Sent!
```

Figure 10: An example of the overall occurrences classification use case system running with user-provided text content

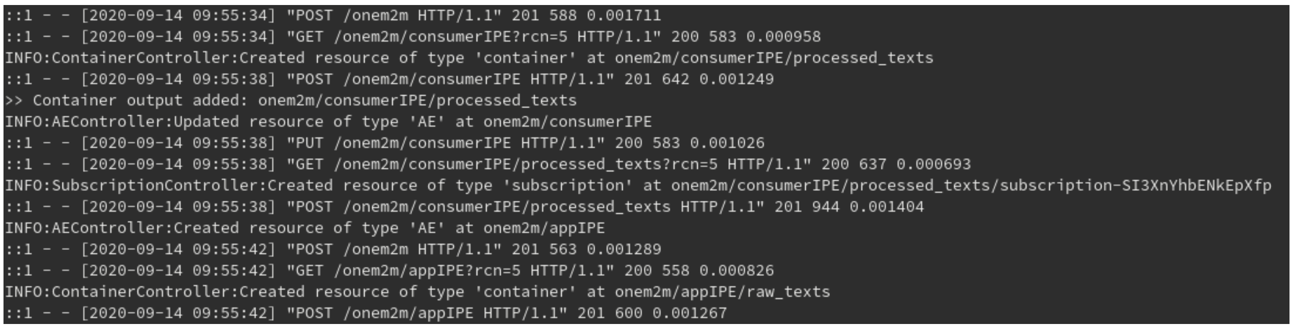
Figure 11 showcases the console logs printed by the CSE (instantiating the service and the containers).



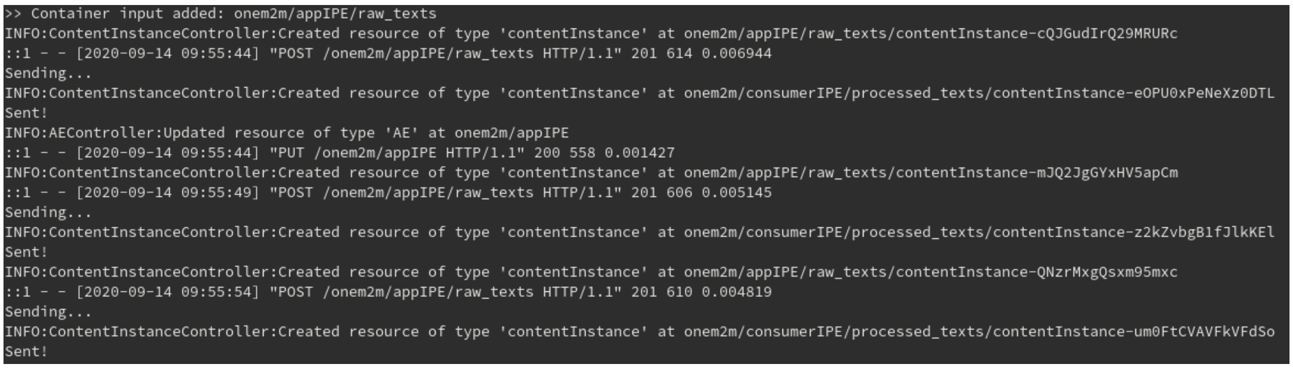
```
[nltk_data]   Package punkt is already up-to-date!
INFO:openmtc_gevent.main:Initializing plugins
INFO:HTTPTransportPlugin:Initializing plugin HTTPTransportPlugin
INFO:NotificationHandler:Initializing plugin NotificationHandler
INFO:RegistrationHandler:Initializing plugin RegistrationHandler
INFO:CleanTextHandler:Initializing plugin CleanTextHandler
Clean text handler initiated
INFO:openmtc_gevent.main:Starting plugins
INFO:HTTPTransportPlugin:Starting plugin HTTPTransportPlugin
INFO:GEventServerRack:OpenMTCWSGIServer started on ('::', 8000, 0, 0)
INFO:NotificationHandler:Starting plugin NotificationHandler
INFO:RegistrationHandler:Starting plugin RegistrationHandler
INFO:RegistrationHandler:registering /mn-cse-1 at /in-cse-1
WARNING:RegistrationHandler:Could not register:
INFO:CleanTextHandler:Starting plugin CleanTextHandler
INFO:openmtc_gevent.main:OpenMTC is running
INFO:AEController:Created resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 09:55:34] "POST /onem2m HTTP/1.1" 201 588 0.001711
::1 - - [2020-09-14 09:55:34] "GET /onem2m/consumerIPE?rcn=5 HTTP/1.1" 200 583 0.000958
INFO:ContainerController:Created resource of type 'container' at onem2m/consumerIPE/processed_texts
::1 - - [2020-09-14 09:55:38] "POST /onem2m/consumerIPE HTTP/1.1" 201 642 0.001249
>> Container output added: onem2m/consumerIPE/processed_texts
INFO:AEController:Updated resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 09:55:38] "PUT /onem2m/consumerIPE HTTP/1.1" 200 583 0.001026
::1 - - [2020-09-14 09:55:38] "GET /onem2m/consumerIPE/processed_texts?rcn=5 HTTP/1.1" 200 637 0.000693
INFO:SubscriptionController:Created resource of type 'subscription' at onem2m/consumerIPE/processed_texts/subscription-SI3XnYhbENkEpXfp
::1 - - [2020-09-14 09:55:38] "POST /onem2m/consumerIPE/processed_texts HTTP/1.1" 201 944 0.001404
INFO:AEController:Created resource of type 'AE' at onem2m/appIPE
::1 - - [2020-09-14 09:55:42] "POST /onem2m HTTP/1.1" 201 563 0.001289
::1 - - [2020-09-14 09:55:42] "GET /onem2m/appIPE?rcn=5 HTTP/1.1" 200 558 0.000826
INFO:ContainerController:Created resource of type 'container' at onem2m/appIPE/raw_texts
::1 - - [2020-09-14 09:55:42] "POST /onem2m/appIPE HTTP/1.1" 201 600 0.001267
>> Container input added: onem2m/appIPE/raw_texts
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-cQJGudIrQ29MRURc
::1 - - [2020-09-14 09:55:44] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 614 0.006944
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/contentInstance-eOPU0xPeNeXz0DTL
Sent!
INFO:AEController:Updated resource of type 'AE' at onem2m/appIPE
::1 - - [2020-09-14 09:55:44] "PUT /onem2m/appIPE HTTP/1.1" 200 558 0.001427
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-mJQ2JgGYxHV5apCm
::1 - - [2020-09-14 09:55:49] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 606 0.005145
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/contentInstance-z2kZvbgB1fJlkKEl
Sent!
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-QNzrMxgQsxm95mxc
::1 - - [2020-09-14 09:55:54] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 610 0.004819
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/contentInstance-um0FtCVAVFkVFdSo
Sent!
INFO:AEController:Updated resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 10:05:38] "PUT /onem2m/consumerIPE HTTP/1.1" 200 583 0.004458
INFO:AEController:Updated resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 10:15:38] "PUT /onem2m/consumerIPE HTTP/1.1" 200 583 0.005455
```

Figure 11: An example of the occurrences' classification CSE

Figure 12 showcases the CSE booting the service and registering the respective AEs needed for the use case.



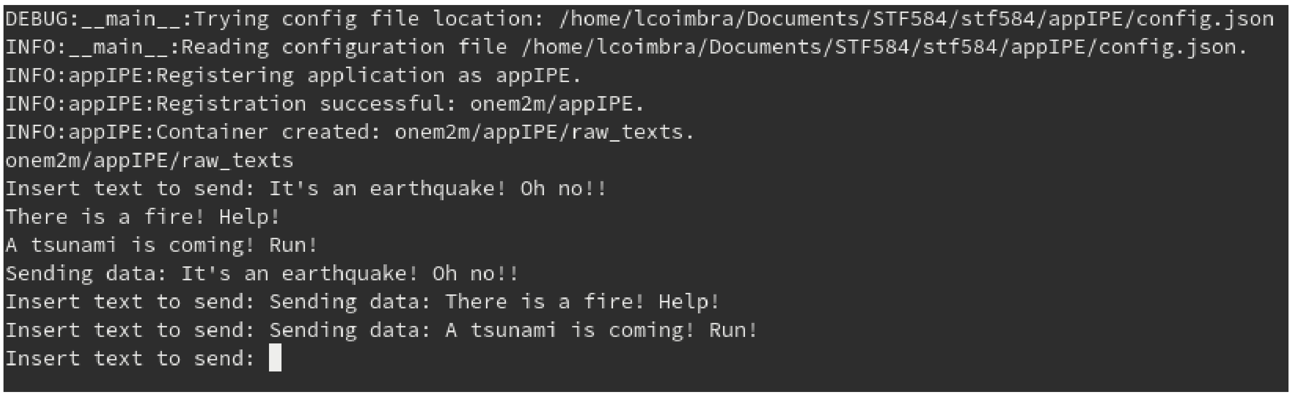
```
::1 - - [2020-09-14 09:55:34] "POST /onem2m HTTP/1.1" 201 588 0.001711
::1 - - [2020-09-14 09:55:34] "GET /onem2m/consumerIPE?rcn=5 HTTP/1.1" 200 583 0.000958
INFO:ContainerController:Created resource of type 'container' at onem2m/consumerIPE/processed_texts
::1 - - [2020-09-14 09:55:38] "POST /onem2m/consumerIPE HTTP/1.1" 201 642 0.001249
>> Container output added: onem2m/consumerIPE/processed_texts
INFO:AEController:Updated resource of type 'AE' at onem2m/consumerIPE
::1 - - [2020-09-14 09:55:38] "PUT /onem2m/consumerIPE HTTP/1.1" 200 583 0.001026
::1 - - [2020-09-14 09:55:38] "GET /onem2m/consumerIPE/processed_texts?rcn=5 HTTP/1.1" 200 637 0.000693
INFO:SubscriptionController:Created resource of type 'subscription' at onem2m/consumerIPE/processed_texts/subscription-SI3XnYhbENkEpXfp
::1 - - [2020-09-14 09:55:38] "POST /onem2m/consumerIPE/processed_texts HTTP/1.1" 201 944 0.001404
INFO:AEController:Created resource of type 'AE' at onem2m/appIPE
::1 - - [2020-09-14 09:55:42] "POST /onem2m HTTP/1.1" 201 563 0.001289
::1 - - [2020-09-14 09:55:42] "GET /onem2m/appIPE?rcn=5 HTTP/1.1" 200 558 0.000826
INFO:ContainerController:Created resource of type 'container' at onem2m/appIPE/raw_texts
::1 - - [2020-09-14 09:55:42] "POST /onem2m/appIPE HTTP/1.1" 201 600 0.001267
```



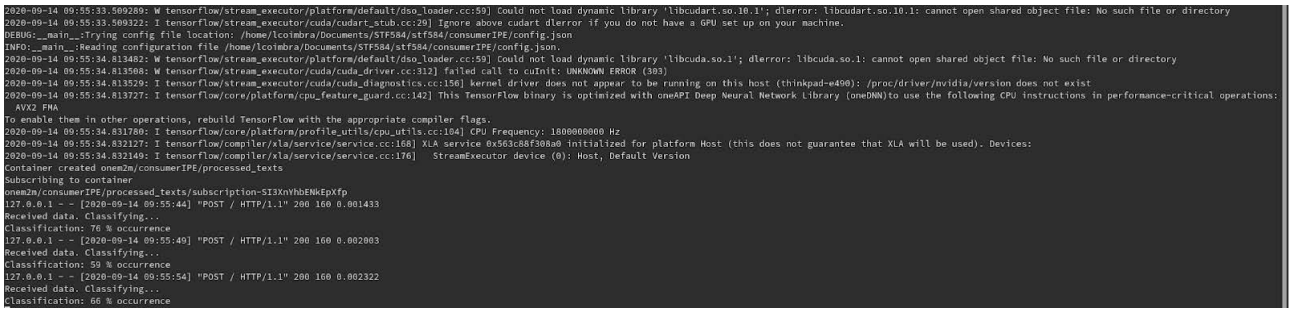
```
>> Container input added: onem2m/appIPE/raw_texts
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-cQJGudIrQ29MRURc
::1 - - [2020-09-14 09:55:44] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 614 0.006944
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/contentInstance-eOPU0xPeNeXz0DTL
Sent!
INFO:AEController:Updated resource of type 'AE' at onem2m/appIPE
::1 - - [2020-09-14 09:55:44] "PUT /onem2m/appIPE HTTP/1.1" 200 558 0.001427
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-mJQ2JgGYxHV5apCm
::1 - - [2020-09-14 09:55:49] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 606 0.005145
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/contentInstance-z2kZvbgB1fJlkKEl
Sent!
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/appIPE/raw_texts/contentInstance-QNzrMxgQsxm95mxc
::1 - - [2020-09-14 09:55:54] "POST /onem2m/appIPE/raw_texts HTTP/1.1" 201 610 0.004819
Sending...
INFO:ContentInstanceController:Created resource of type 'contentInstance' at onem2m/consumerIPE/processed_texts/contentInstance-um0FtCVAVFkVFdSo
Sent!
```

Figure 12: An example of the CSE creating the AE resources

In figure 13, the AE device is publishing information to the gateway and the rest of the system, and the other AE is consuming the cleaned text and classifying it.



```
DEBUG:__main__:Trying config file location: /home/lcoimbra/Documents/STF584/stf584/appIPE/config.json
INFO:__main__:Reading configuration file /home/lcoimbra/Documents/STF584/stf584/appIPE/config.json.
INFO:appIPE:Registering application as appIPE.
INFO:appIPE:Registration successful: onem2m/appIPE.
INFO:appIPE:Container created: onem2m/appIPE/raw_texts.
onem2m/appIPE/raw_texts
Insert text to send: It's an earthquake! Oh no!!
There is a fire! Help!
A tsunami is coming! Run!
Sending data: It's an earthquake! Oh no!!
Insert text to send: Sending data: There is a fire! Help!
Insert text to send: Sending data: A tsunami is coming! Run!
Insert text to send: 
```



```
2020-09-14 09:55:33.509289: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open shared object file: No such file or directory
2020-09-14 09:55:33.509322: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
DEBUG:__main__:Trying config file location: /home/lcoimbra/Documents/STF584/stf584/consumerIPE/config.json
INFO:__main__:Reading configuration file /home/lcoimbra/Documents/STF584/stf584/consumerIPE/config.json.
2020-09-14 09:55:34.813482: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or directory
2020-09-14 09:55:34.813508: W tensorflow/stream_executor/cuda/cuda_driver.cc:312] failed call to cuInit: UNKNOWN ERROR (303)
2020-09-14 09:55:34.813529: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (thinkpad-e490): /proc/driver/nvidia/version does not exist
2020-09-14 09:55:34.813727: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical operations:
  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2020-09-14 09:55:34.831780: I tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency: 1800000000 Hz
2020-09-14 09:55:34.832127: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x563c88f308a0 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2020-09-14 09:55:34.832149: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
Container created onem2m/consumerIPE/processed_texts
Subscribing to container
onem2m/consumerIPE/processed_texts/subscription-SI3XnYhbENkEpXfp
127.0.0.1 - - [2020-09-14 09:55:44] "POST / HTTP/1.1" 200 160 0.001433
Received data. Classifying...
Classification: 76 % occurrence
127.0.0.1 - - [2020-09-14 09:55:49] "POST / HTTP/1.1" 200 160 0.002003
Received data. Classifying...
Classification: 59 % occurrence
127.0.0.1 - - [2020-09-14 09:55:54] "POST / HTTP/1.1" 200 160 0.002322
Received data. Classifying...
Classification: 66 % occurrence
```

Figure 13: An example of the two AEs respectively publishing text to the gateway (above) and being notified for occurrences classification (below)

7 Lessons learned

7.1 Main lessons

Taking into account that it was designed for IoT use cases, the architecture of oneM2M adapts very nicely to AI use cases under this scope. Therefore, Machine Learning (ML) methods can be easily implemented directly over the data that is notified to the AEs subscribed to certain containers.

The complexity of implementing new CSFs or extending exiting ones highly depends on the selected platform architecture type: monolithic or micro-service architecture. While a monolithic oneM2M platform is a single unified unit implementing all the CSFs, a microservices oneM2M architecture breaks it down into a collection of smaller independent units. These units carry out each CSF process as a separate service. A microservice-based oneM2M architecture is much easier to extend compared to a monolith one because CSFs interact with each other's through well-defined APIs and are designed to be flexible, autonomous, scalable, fault tolerant, and leveraging the diversity in programming languages.

7.2 Implementation specific lessons

The implementation of new CSFs has led to the assessment that the oneM2M documentation is a reliable source of support for developers integrating a oneM2M platform like OpenMTC. Regardless of the documentation made available by the implementers, since it is compliant with the open standard, oneM2M documentation and examples have helped the team to better understand the technical details and workflows within OpenMTC.

With the purpose of disseminating these new CSFs within the open-source and AI community, there is the need for the team from OpenMTC to accept contributions into the master repository (following the pull requests approach) in order to appear in the official documentation portals like Readthedocs [i.4] or Github [i.5], which are the existing best options that could work as a catalogue of existing CSFs for OpenMTC.

Annex A: Change History

Date	Version	Information about changes
Mar. 2020	0.0.1	Initial draft version for internal STF work
May 2020	0.0.2	Inclusion of early descriptions of use case 3
June 2020	0.0.3	Inclusion of early descriptions of use cases 1 and 2
June 2020	0.0.4	Clean version for upload to SmartM2M portal
June 2020	0.1.0	Version uploaded on SmartM2M portal as Early Draft (identical to v0.0.4)
Sep. 2020	0.1.1	Additional information from sensinov on UC1 and UC2
Sep. 2020	0.1.2	Additional information from UbiWhere on UC3
Sep. 2020	0.1.3	Alignment and clean-up for upload (identical to stable draft v0.2.0)
Sep. 2020	0.2.1	Initial clean-up of stable draft
Sep. 2020	0.2.2	Additional contributions to Use Case 1
Sep. 2020	0.2.3	Additional contributions to Use Case 3
Sep. 2020	0.2.4	Additional contributions to Use Case 1 and 2
Oct. 2020	0.2.5	Alignment and cleanup for upload (identical to final draft v1.0.0)
Nov. 2020	1.1.1	Final Draft TBapproved, cleaned-up by editHelp! and reviewed byTechnical Officer okay

History

Document history		
V1.1.1	December 2020	Publication