ETSI TR 118 568 V5.0.1 (2023-08)

**AI enablement to oneM2M
(oneM2M TR-0068 version 5.0.1)**

Reference

DTR/oneM2M-000068

Keywords

artificial intelligence, IoT, M2M

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Partnership Project oneM2M (oneM2M).

1 Scope

The present document is analysing existing AI/ML technologies that can be resourced into oneM2M architecture. The present document is also investigating potential AI/ML service use cases that use data collected in the oneM2M system. The study on existing AI/ML technologies and use cases are further analysed in the present document to understand what features are supported and unsupported by the oneM2M system. Based on the result of the present document, it will identify potential requirements and key features to enable AI/ML in the oneM2M system.

2 References

2.1 Normative 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.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

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 necessary for the application of the present document.

Not applicable.

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] oneM2M Drafting Rules.

[i.2] "Large scale labelled video data augmentation for semantic segmentation in driving scenarios", Ignas Budvytis, Patrick Sauer, Thomas Roddick, Kesar Breen, Roberto Cipolla; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 230-237.

[i.3] Last mile from Wikipedia.

[i.4] Metaverse from Wikipedia.

[i.5] "The Metarverse Is Coming And It's A Very Big Deal" from Forbes.

[i.6] ETSI TR 103 674 (V1.1.1): "SmartM2M; Artificial Intelligence and the oneM2M architecture".

[i.7] ETSI TR 103 675 (V1.1.1): "SmartM2M; AI for IoT: A Proof of Concept".

[i.8] ETSI TR 103 778: "SmartM2M; Use cases for cross-domain data usability of IoT devices".

[i.9] RDM-2021-0082R01: "ML/AI Use cases from STF 601".

[i.10] Directive 2011/24/EU of the European Parliament and of the Council of 9 March 2011 on the application of patients' rights in cross-border healthcare.

[i.11] ASSIST-IoT deliverable (D3.2): "Use Cases Manual & Requirements and Business Analysis".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

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

AI	Artificial Intelligence
DSS	Decision Support System
IoT	Internet of Things
ML	Machine Learning

4 Conventions

The key words "Shall", "Shall not", "May", "Need not", "Should", "Should not" in the present document are to be interpreted as described in the oneM2M Drafting Rules [i.1].

5 Introduction

Today's business world is transforming with the adoption of the Internet of Things (IoT). IoT is instrumental in capturing a tremendous amount of data from multiple sources. However, managing the vast variety of data originating from an immense number of IoT devices complicates data collection, processing, and analysis. Realizing the full potential of future IoT services will require an investment in new technologies. The convergence of Artificial Intelligence (AI) and Machine Learning (ML) with IoT can redefine how industries, businesses, and economies function. AI/ML-enabled IoT gives rise to intelligent machines that exhibit smart behaviour and support decision-making with little to no human interference. Combining these two streams benefits ordinary people and specialists alike. While IoT focuses on devices interacting via the internet, AI/ML enables these devices to learn from their data and experiences.

Therefore, oneM2M needs to investigate the necessary features to support AI/ML capabilities in the oneM2M architecture and service layers. The present document analyses existing AI technologies (including Machine Learning) that can be integrated into the oneM2M architecture. The present doument also explores potential AI/ML service use cases that utilize IoT data. The study of AI/ML technologies and use cases is further examined to understand which features are currently supported and which are not by the oneM2M system. Consequently, the present document introduces potential requirements for AI/ML features in the oneM2M system.

6 AI/ML Technologies

6.1 Overview of AI/ML

Artificial Intelligence (AI) is the ability of a computer program to learn and think. Everything can be considered Artificial intelligence if it involves a program performing functions that the intelligence of a human can do.

AI is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from experience.

On the other hand, Machine Learning (ML) is a type of AI that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. ML algorithms use historical data as input to predict new output values.

Advantages of using AI/ML

The advantages of AI applications are enormous and can revolutionize any professional sector. Below are a few of those:

- AI/ML drives down the time taken to perform a task. It enables multi-tasking and eases the workload for existing resources.
- AI/ML enables the execution of previously complex tasks without high cost outlays.
- AI/ML operates 24x7 without interruption or breaks and has no downtime.
- AI/ML augments the capabilities of differently-abled individuals.
- AI/ML has mass-market potential as it can be deployed across industries.
- AI/ML facilitates decision-making by making the process faster and smarter.

Applied areas

AI/ML is used in various fields of technology that require automation and intelligence. Its fields of application are various such as Natural Language Generation, Speech Recognition, Machine Learning Platforms, Virtual Agents, Decision Management, AI Optimized Hardware, Deep Learning Platforms, Robotic Process Automation.

Importance of AI/ML

AI/ML technology is crucial because it enables human capabilities - understanding, reasoning, planning, communication, and perception - to be undertaken by software increasingly effectively, efficiently, and at low cost.

Artificial intelligence can enhance things as simple as household appliances to medical neural networks that can diagnose diseases or perform operations. As society changes and embraces a more automated lifestyle, new jobs will be created to monitor, enhance, and repair these automated machines.

Data and AI/ML

ML is the link that connects Data Science and AI. That is because it is the process of learning from data over time. AI is the tool that helps an intelligent service gets results and solutions for specific problems. On the other hand, machine learning is what helps in achieving that goal using data.

IoT and AI/ML

AI-enabled IoT creates intelligent machines that simulate smart behaviour and supports decision making with little or no human interference. While IoT deals with managing devices and collecting data, AI/ML enables IoT to provide intelligent services using the collected data.

6.2 AI/ML and IoT

6.2.1 Steps for AI/ML

Machine Learning gives devices the ability to learn from their experiences and improve themself without doing any coding. Machine Learning is the study of making machines more human-like in their behaviour and decisions by allowing them the ability to learn and develop their own programs. This can be done with minimum human intervention, i.e. no explicit programming. The learning process is automated and improved based on the experiences of the machines throughout the process. Good quality data is fed to the machines, and different algorithms are used to build ML models to train the machines on this data. The choice of algorithm depends on the type of data and activity that needs to be automated.

In traditional programming, the input data and a program are fed into a machine to generate output. When it comes to machine learning, input data and expected output are fed into the machine during the learning phase, and it works out a program for itself. To understand this better, refer to the Figure 6.2.1-1.

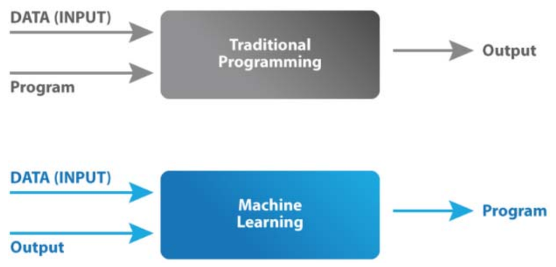

Figure 6.2.1-1: Traditional programming vs. machine learning

Several common terminologies of ML are follows:

- **Model:** A machine learning model is the mathematical representation of a real-world process. A machine learning algorithm along with the training data builds a machine learning model.
- **Feature:** A feature is a measurable property or parameter of the dataset.
- **Feature Vector:** It is a set of multiple numeric features. This is used as an input to the machine learning model for training and prediction purposes.
- **Training:** An algorithm takes a set of data known as "training data" as input. The learning algorithm finds patterns in the input data and trains the model for expected results (target). The output of the training process is the machine learning model.
- **Prediction:** Once the machine learning model is ready, it can be fed with input data to provide a predicted output.
- **Target (Label):** The value that the machine learning model has to predict is called the target or label.
- **Overfitting:** When a massive amount of data trains a machine learning model, it tends to learn from the noise and inaccurate data entries. Here the model fails to characterize the data correctly.
- **Underfitting:** It is the scenario when the model fails to decipher the underlying trend in the input data. It destroys the accuracy of the machine learning model. In simple terms, the model or the algorithm does not fit the data well enough.

In order to perform ML, there are Seven Steps to take as follows:

Step 1: Gathering data

For the purpose of developing our machine learning model, the first step would be to gather relevant data. This step is very crucial as the quality and quantity of gathered data will have a direct impact to a model for prediction. Mistakes such as choosing the incorrect features or focusing on limited types of entries for the dataset may render the model completely ineffective.

Step 2: Data preparation

The next step is data preparation where the data is located in a place and then prepared for the use in the ML training. The data preparation step split the datasets into 2 parts. The larger part (~80 %) is used for training the model while the smaller part (~20 %) is used for evaluation purposes. This is important because using the same datasets for both training and evaluation would not give a fair assessment of the model's performance in real world scenarios. Apart from the data split, additional steps are taken to refine the datasets. This could include removing duplicate entries, discarding incorrect readings etc.

Step 3: Choosing a model

The next step is to select a model among the many that researchers and data scientists have created over the year. There are various existing models developed by data scientists which can be used for different purposes. These models are designed with different goals in mind. For instance, some models are more suited to dealing with texts while another model may be better equipped to handle images.

Step 4: Training

At the heart of the machine learning process is the training of the model. Bulk of the "learning" is done at this stage. The training step requires patience and experimentation. It is also useful to have knowledge of the field where the model would be implemented. For instance, if a machine learning model is to be used for identifying high risk clients for an insurance company, the knowledge of how the insurance industry operates would expedite the process of training as more educated guesses can be made during the iterations. Training can prove to be highly rewarding if the model starts to succeed in its role. This process then repeats and each cycle of updating is called one training step.

Step 5: Evaluation

With the model trained, it needs to be tested to see if it would operate well in real world situations. That is why the part of the dataset created for evaluation is used to check the model's proficiency. This puts the model in a scenario where it encounters situations that were not a part of its training. Evaluation becomes highly important when it comes to commercial applications. Evaluation allows data scientists to check whether the goals they set out to achieve were met or not. If the results are not satisfactory then the prior steps need to be revisited so that root cause behind the model's underperformance can be identified and, subsequently, rectified. If the evaluation is not done properly then the model may not excel at fulfilling its desired commercial purpose.

Step 6: Parameter tuning

If the evaluation is successful, the next step is to perform parameter. This step tries to improve upon the positive results achieved during the evaluation step. There were a few parameters that were implicitly assumed when the training was done. Another parameter included is the learning rate that defines how far the line is shifted during each step, based on the information from the previous training step. These values all play a role in the accuracy of the training model, and how long the training will take.

Step 7: Prediction

ML basically answers questions using data. Therefore, the final step of the machine learning process is answer a question using prediction based on a developed model. This is the stage where the model is to be ready for practical applications.

6.2.2 Data Augmentation

AI/ML algorithms have become a key standard for most vision and machine learning issues. Despite its general use and high performance for many applications, they have some disadvantages. A big problem with AI/DL methods is the size of the dataset to be used for training. Appropriate training methods require a large dataset. However, a large dataset may not be available for all problems. In this case, appropriate method refer as data augmentation is used to obtain a larger dataset from original dataset.

Data augmentation techniques artificially generate different versions of a real dataset by adding slightly modified copies of already existing data or newly created artificial data from existing data to increase its size. Data augmentation is useful to improve the performance and outcomes of machine learning models by forming new and different examples to train datasets. If the dataset in a machine learning model is rich and sufficient, the model performs better and is more accurate.

These techniques enable machine learning models to be more robust by creating variations that the model may see in the real world. Without IoT platforms, AI applications directly take the dataset from the source devices such as CCTV and drones. Applications then apply data augmentation techniques to make a larger dataset. As many AI/ML applications use data augmentation techniques, they can reduce their tasks to manage training datasets by introducing data augmentation functions to IoT platforms. AI/ML applications take the small dataset of images and use data augmentation functions to transform the source images to different sizes by zooming in or zooming out, flipping them vertically or horizontally or changing the brightness or rotating whatever makes sense for the object, as shown in Figure 6.2.2-1.

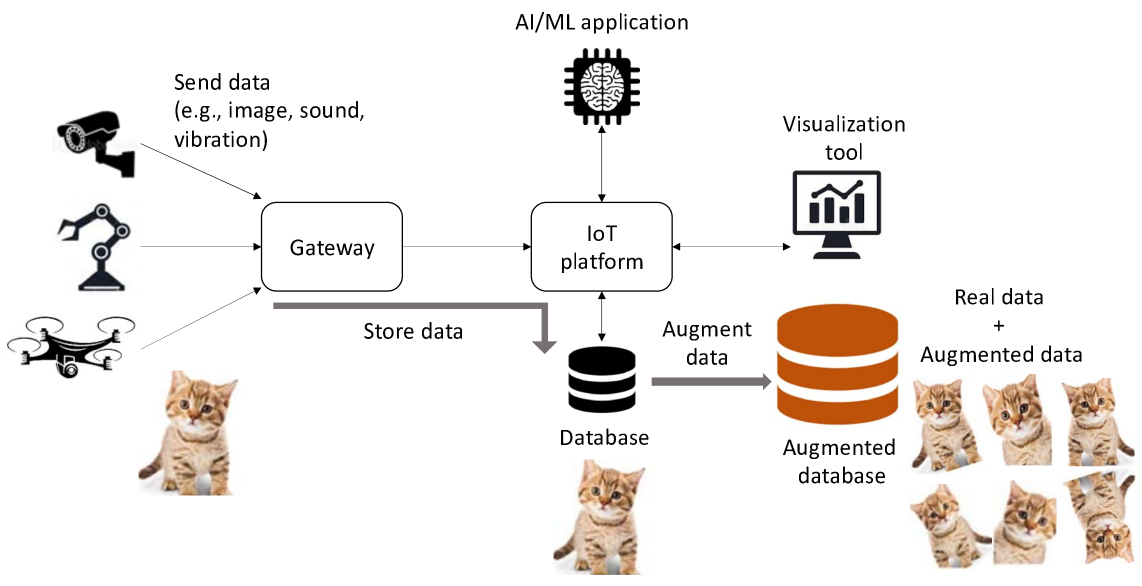

Figure 6.2.2-1: Data augmentation and IoT

Benefits of data augmentation include:

- Improving model prediction accuracy
  - adding more training data into the models
  - preventing data scarcity for better models
  - reducing data overfitting and creating variability in data
  - increasing generalization ability of the models
  - helping resolve class imbalance issues in classification
  - more
- Reducing costs of collecting and labelling data

There exist several basic but powerful data augmentation techniques that are widely used to increase the amount of dataset. Followings are several well known data augmentation techniques:

1) **Adding noise:** For blurry images, adding noise on the image can be useful. By "salt and pepper noise", the image looks like consisting of white and black dots.

2) **Cropping:** A section of the image is selected, cropped and then resized to the original image size.

3) **Flipping:** The image is flipped horizontally and vertically. In flipping, the pixels are rearranged while protecting features of image. Vertical flipping is not meaningful for some photos, but it can be useful for example cosmology or microscopic photos.

4) **Rotation:** The image is rotated by a degree between 0 and 360 degree. Every rotated image will be unique in the model. For example, the selected image can be rotated 45 degree. The value can from 359 ~ -359. In the random rotate case, two values, for example, 30, and 90 can be entered. Then a random degree between the two input parameters can be selected to rotate the image. Total number of images can be decided. In this case, the target image can be rotated to generate the given number of images. For example, input parameters A degree, B degree, 100 means that generates 100 images through rotating the source image between A and B degrees.

5) **Scaling:** The image is scaled outward and inward. An object in new image can be smaller or bigger than in the original image by scaling. For example, the image can be scaled below to 100 % to 50 % of the image height/width. In this case, a random value between 100 % ~ 50 % will be selected to resize the image.

6) **Translation:** The image is shifted into various areas along the x-axis or y-axis, so neural network looks everywhere in the image to capture it.

7) **Brightness:** The brightness of the image is changed and new image will be darker or lighter. This technique allows the model to recognize image in different lighting levels. The brightness of the image can be changed using different contrast. Depending on the selected contrast various input parameters should be selected, for example, 1. Contrast options, 2. Gamma contrast, 3. Sigmoid contrast, 4. Linear contrast, 5. Various contrast filter can be used.

8) **Contrast:** The contrast of the image is changed and new image will be different from luminance and color aspects. The following image's contrast is changed randomly.

9) **Color space transformations:** Change RGB color channels, intensify any color. The color of image is changed by new pixel values.

10) **Kernel filters (sharpen or blur an image):** These are a very popular techniques in image processing to sharpen and blur images. These filters work by sliding an n × n matrix across an image with either a Gaussian blur filter, which will result in a blurrier image, or a high contrast vertical or horizontal edge filter which will result in a sharper image along edges.

11) **Random Erasing (delete a part of the initial image):** It retains the overall structure of the object, only occluding some parts of object. Areas are re-assigned with random values, which can be viewed as adding noise to the image.

12) **Mixing images:** Basically, this is a technique to mix images with one another. Might be counterintuitive but it work.

7 AI/ML Use Cases using IoT data

7.1 Use case #1 - Data Augmentation for Autonomous Driving

7.1.1 Description

AI/ML provides good solutions to various domains that need image classification recognition, such as CCTV and Home security cameras. Similarly, autonomous driving technologies use AI/ML to detect objects around a vehicle. Typically such AI/ML requires a vast amount of datasets to have an accurate result. However, many AI/ML services for autonomous driving still operates on limited size datasets. Even there exist many datasets to use, labelling each image by hand takes around an hour. Therefore, AI/ML services for autonomous driving use techniques for data augmentation to acquire a large number of datasets from limited size datasets.

Applying data augmentation techniques to collected datasets require high computing power and large data storage. The autonomous driving application typically does not have such resources. IoT platforms serve the autonomous driving application can provide a common function for data augmentation. The application can offload their data augmentation tasks (e.g. applying image processing and storing large datasets) to IoT platforms. As many IoT services using AI/ML techniques require data augmentation function, the introduction of data augmentation function to IoT platforms benefits many intelligent IoT services.

Several reasons of data augmentation interest in autonomous driving are:

- Limited size available dataset, e.g. road images, various objects
- The use of data augmentations helped to improve the results
- Sharing data is not easy due to data privacy regulations

7.1.2 Source

Large scale labelled video data augmentation for semantic segmentation in driving scenarios [i.2].

7.1.3 Actors

This use case contains actors as follows:

- Autonomous driving application: an application using developed AI/ML model
- AI/ML data management application: an application managing AI/ML image data
- AI-enabled IoT platform: An IoT platform collects image data from various sources

AI/ML data management application builds a model for autonomous driving, and autonomous driving application uses the developed AI/ML model to make a decision on various driving situations such as when to stop, turn on/off engine.

7.1.4 Pre-conditions

- The AI-enabled IoT platform holds a set of good quality image data for autonomous driving
- The AI-enabled IoT platform provides features to handle requested image augment techniques
- The amount of collected source images to build a model is not enough

7.1.5 Triggers

- AI/ML data management application requests to augment data to build AI/ML model for autonomous driving

7.1.6 Normal Flow

Figure 7.1.9-1 illustrates the high-level flows of data augmentation for AI/ML use case, which consists of the following steps:

- Step 1: The AI/ML management application sends a request to the AI-enabled IoT platform to augment source images. The request may include the following information:
  - Source images.
  - Data augmentation techniques to apply.
  - Additional information for a selected data augmentation technique, for example, the number of images to generate after applying the data augmentation technique.
- Step 2: The AI-enabled IoT platform analyses the received request and stores retrieved information internally. Then the AI-enabled IoT platform applies the selected data augmentation technique and generates a set of augmented images. The IoT platform stores generated images with their own identifiers.
- Step 3: The AI-enabled IoT platform returns the result to the AI/ML management application. The results may include a summary of the requested data augmentation, for example, the number of generated images, links to access such augmented images.
- Step 4: The autonomous driving application uses augmented dataset to build a model for autonomous driving, object detection, driving control etc. If the AI-enable IoT platform supports the generation of a model for autonomous driving, the application can download and use the model. Otherwise, the autonomous driving application build its own model using the augmented dataset.

7.1.7 Alternative Flow

None.

7.1.8 Post-conditions

The AI-enabled IoT platform has dataset for the source images and augmented images from the source.

7.1.9 High Level Illustration

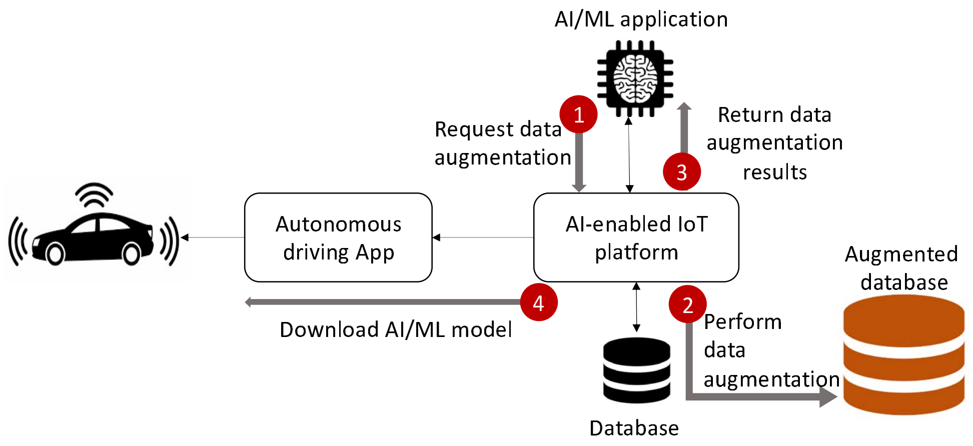

Figure 7.1.9-1: Data augmentation for autonomous driving

7.1.10 Potential Requirements

1) The oneM2M System will be able to handle data augmentation requests for AI/ML purposes.
2) The oneM2M System will be able to generate augmented data resources from a given source data and data augmentation technique.
3) The oneM2M System will be able to manage data for AI/ML purposes such as model training and augmentation of training dataset.

7.2 Use case #2 - Last Mile Delivery

7.2.1 Description

Last Mile is a term used in supply chain management and transportation planning to describe the last leg of a journey comprising the movement of people and goods from a transportation hub to a final destination. For example, when moving from your home (departure) to your office (destination), the travel between your home and your office through public transportation is First Mile, the short distance to your office is Last Mile [i.3].

In logistics, the first mile covers the area where the goods are sent from the provider, the hub terminal, and the sub terminal. Then, the last mile logistics service provider carries goods from the sub terminal to the consumer directly. Robots and autonomous vehicles are the ones that logistics companies are developing for the Last Mile delivery service.

IoT and AI/ML technologies are considered playing a pivotal role in this area, Last Mile Delivery. IoT platform controlling and managing robots for last mile delivery collects billions of gigabytes of structured and unstructured data everyday. AI/ML technologies harness this dataset to make build a good model for various decisions, which performed by human being.

In the last mile delivery use case, a last mile robot picks up goods to deliver and navigates to the customer. The robot will need to learn to operate in more complex and varied environments with minimal or no human intervention. This requires extensive computing power and storages. This ranges from gathering data, defining object classes, labelling the data, and training a selected ML model in many environments and conditions.

In order to make ML models work as expected, continuous maintaining of trained models is essential. Typically data is going to change over time. Even in the same building for the last mile delivery, the number of people and objects located in the building are changed over time. This means that a ML model built a week ago may not provide accurate predictions for a last mile robot. Therefore, continuous training of a model using new updated data is an essential part of the last mile delivery service using AI/ML.

Therefore, such complex tasks are expected to be offloaded to an edge or cloud IoT platform where higher computing power and huge data storage can be supported. Also IoT platforms hold other data that can also be used for training. In this use case, IoT platform can provide the following functions:

- Manage structured and unstructured data for training
- Update trained model using new inputs everyday
- Classify AI/ML data into two parts, i.e. training and validating

7.2.2 Source

"Last Mile: The last leg of a journey" [i.3].

7.2.3 Actors

- Last mile delivery robot: a robot picks up and delivers goods.
- Last mile delivery application: an application controls the last mile delivery robot, e.g. configuring locations for pick up and deliver.
- AI-enabled IoT platform: An IoT platform stores data from the robot, classifies and manages data for training.

7.2.4 Pre-conditions

- The last mile delivery robot is energy and computing power constrained so that heavy computational tasks should be performed in IoT platforms.
- The AI-enabled IoT platform holds a set of good quality training data for the last mile delivery service.

7.2.5 Triggers

- Last mile delivery application configures delivery information to the robot and requests to start delivery.

7.2.6 Normal Flow

Figure 7.2.9-1 illustrates the high-level flows of the last mile delivery use case, which consists of the following steps:

- Step 1: A new delivery order is digitally entered to an IoT platform:
    - Target location.
    - Pick up location.
    - Identifier of goods.
    - Goals to achieve (e.g. time, shortest path, security).
    - Geo-fense information.
- Step 2: The IoT platform selects a robot and notifies the order to deliver.
- Step 3: The last mile delivery robot retrieves the order and loads the goods from the pick up location.
- Step 4: The robot starts the delivery. While the robot routes to the destination, it capture data. The data is sent to the IoT platform for managing and processing for training. Such data can be used for updating the trained model to understand edges, many classes of fixed objects, path structures, etc.
- Step 5: The robot delivers the goods to the destination place correctly, then returns to the stand-by location.

7.2.7 Alternative Flow

None.

7.2.8 Post-conditions

The AI-enabled IoT platform has dataset for continuous training of the delivery AI/ML model.

7.2.9 High Level Illustration

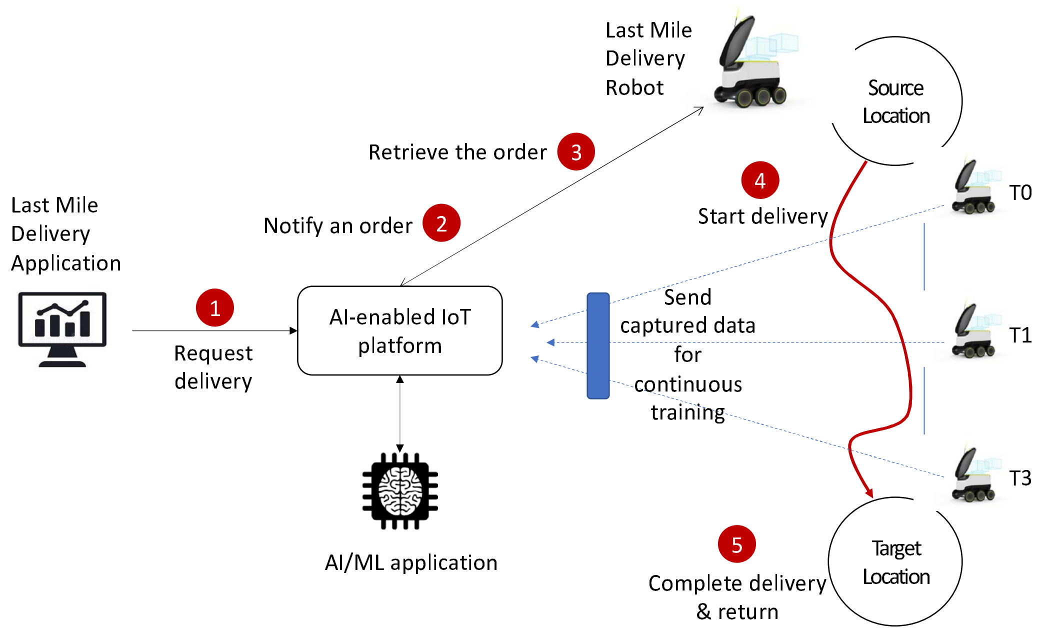

Figure 7.2.9-1: Conceptual diagram of the last mile delivery

Figure 7.2.9-2 shows a possible resources structure that shows how AI/ML models can be managed in the oneM2M system. For example, available AI/ML models can be stored in the availableAIModel container. Trained AI/ML models after re-learning can be stored in the aiModel container resource.

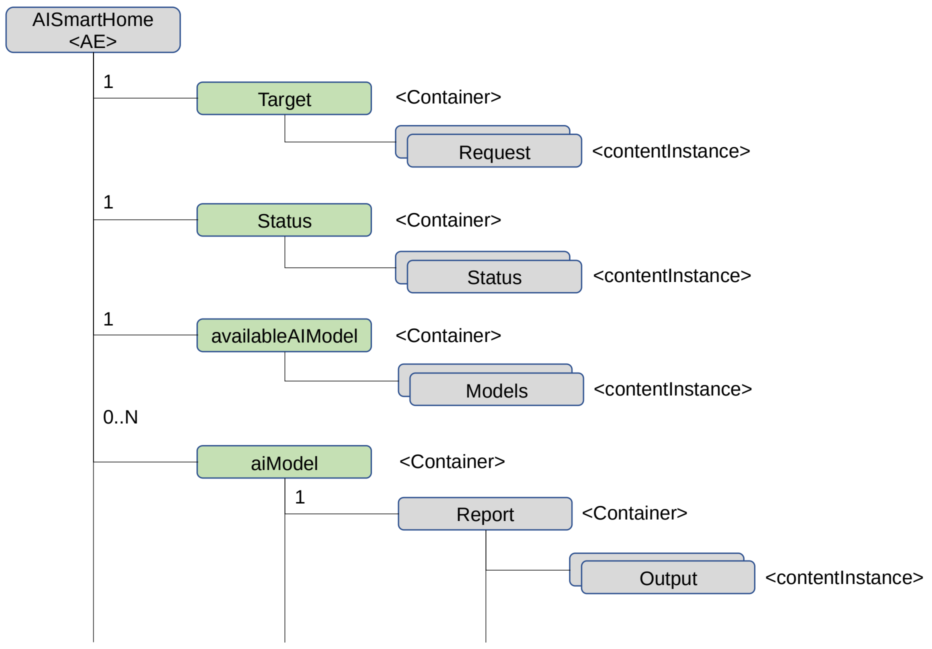

Figure 7.2.9-2: A possible resource structure to manage AI/ML models in oneM2M

In the case of IoT services, such as Personal Mobility Device (PDM), learning data is collected from hundreds or thousands of IoT devices, not just one or two. If these IoT devices use AI features, then AI models specialized for each device should be trained. In this case, information about which IoT device the result of the trained AI model corresponds to is required. Therefore, the resource tree structure example shown in Figure 7.2.9-2 above can be updated and designed by adding the 'Source' resource under the 'aiModel' resource, as shown in Figure 7.2.9-3.

The summary of each resource is as follows:

- 'Target': A resource used by users to request and cancel real-time artificial intelligence services.
- 'Status': A resource used to inform users that their AI service request and cancellation request have been registered normally.
- 'availableAIModel': A resource that holds the list of available AI models.
- 'aiModel': A resource that is automatically created whenever a new AI model is created using training data and available selected AI models.
- 'Source': A resource containing the source information of the AI model created after learning.
- 'Report': A resource used to store information about the inference results of an AI model.

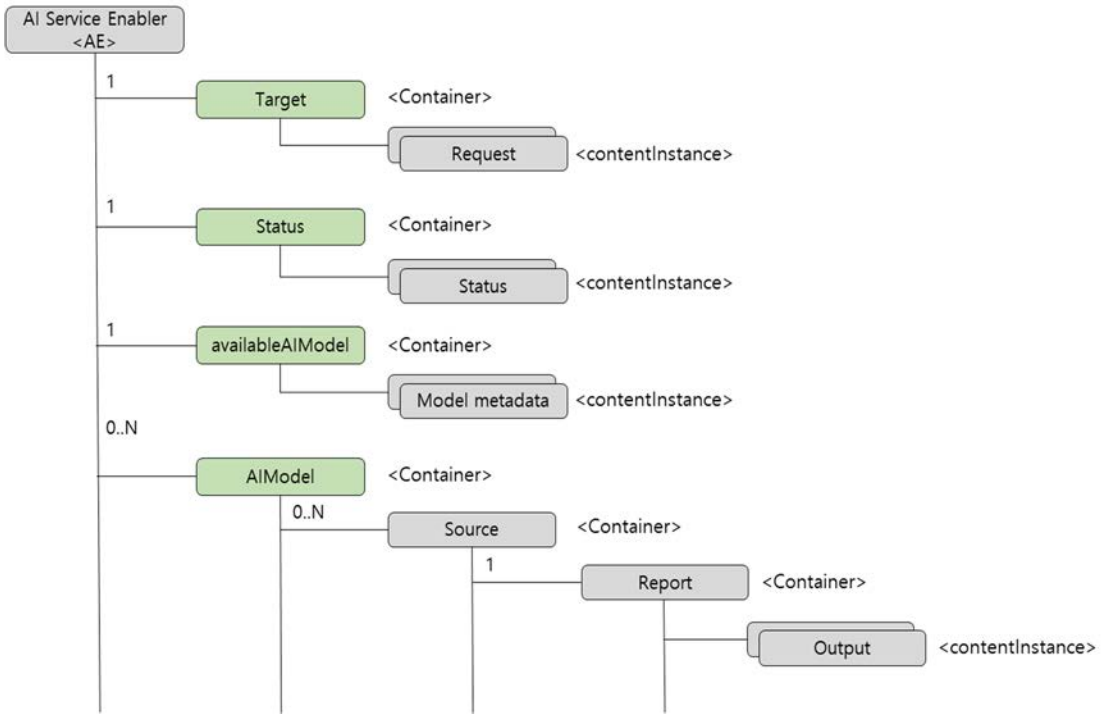

Figure 7.2.9-3: A possible resource structure to manage AI/ML models from multiple sources in oneM2M

7.2.10 Potential Requirements

1) The oneM2M System will be able to manage structured and unstructured data for training, for example, preprocessing data, describing data and inferring meaning.

2) The oneM2M System will be able to update trained AI/ML model according to continuous measuring data e.g. location, time series and historical data.

3) The oneM2M System will be able to provide a classification function (e.g. split data into two parts, training and validating) in supervised Machine Learning.

7.3 Use case #3 - Smart Virtual Store using Metaverse

7.3.1 Description

The word "Metaverse" is made up of the prefix "meta" (meaning beyond) and the stem "verse" (a back-formation from "universe"); the term is typically used to describe the concept of a future iteration of the internet, made up of persistent, shared, 3D virtual spaces linked into a perceived virtual universe [i.4]. The metaverse in a broader concept refer to realize virtual worlds using IoT, AI and Augmented Reality (AR)/Virtual Reality (VR) [i.5].

A metaverse-based online store where stores in the real world are created as digital twins in the metaverse virtual space, and users visit a virtual store in the metaverse space to purchase preferred products. For real-time synchronization between the real-world and the virtual stores in the metaverse, various smart sensors are used to sense real-world products intelligently. The edge node at the real world store loads a trained AI/ML model and infers products' information. The retrieved product data is then transferred to the IoT platform for real-time synchronization.

A user can now purchase products from a virtual store in the metaverse. The purchase info in the metaverse is notified to the administrator and the purchased product is delivered to the user.

7.3.2 Source

"The Metarverse Is Coming And It's A Very Big Deal" [i.5].

7.3.3 Actors

- User: a user who shops products in the metaverse virtual store.
- Edge node: an IoT edge node perform machine learning model to retrieve shopping information from sensors deployed in the real world store.
- AI-enabled IoT platform: An IoT platform synchronizes data between the real-world and virtual stores.

7.3.4 Pre-conditions

- The IoT edge node is loaded with a AI/ML model to detect products from the real world.

7.3.5 Triggers

- Users do virtual shopping in the metaverse virtual store and purchase preferred products.

7.3.6 Normal Flow

Figure 7.3.9-1 illustrates the high-level flows of the metaverse virtual store use case, which consists of the following steps:

- Step 1: The Edge node collects data from sensors in the real world store.
- Step 2: The Edge node infers product information from the collected data.
- Step 3: The Edge node send inferred product data to the IoT platform.
- Step 4: The virtual store application in the metaverse retrieves information from the IoT platform about the products in the real world.
- Step 5: A user picks up products from the virtual store. The purchased information is sent to the IoT platform.
- Step 6: An admin application gets a notification for the purchase, and delivers real products to the user.

7.3.7 Alternative Flow

None.

7.3.8 Post-conditions

The AI-enabled IoT platform synchronizes product data in the real world and virtual stores.

7.3.9 High Level Illustration

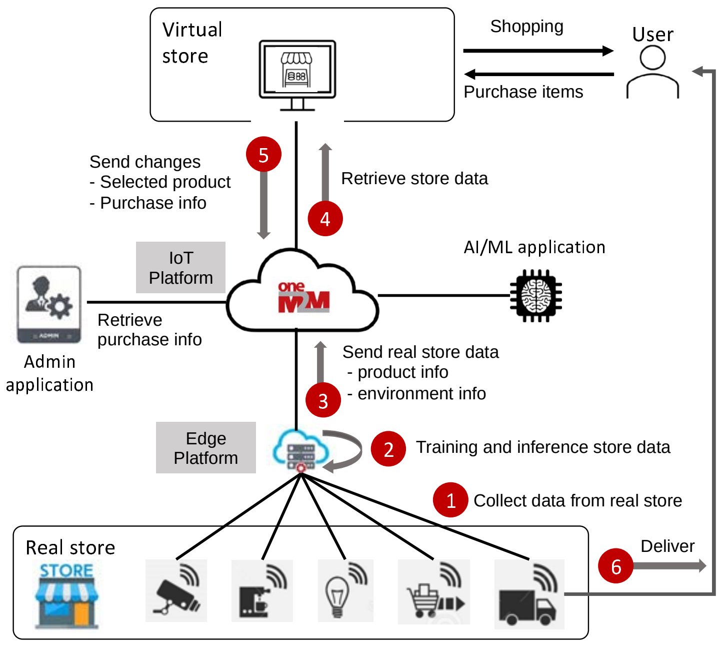

Figure 7.3.9-1: Conceptual diagram of metaverse virtual store

7.3.10 Potential Requirements

1) The oneM2M System will be able to synchronize between real and virtual world devices.

2) The oneM2M System will be able to enable Edge/Fog Nodes to run AI/ML models to retrieve information from the real world.

7.4 Use case #4 - Detection of patterns in video streams

7.4.1 Description

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.

For potential benefits for oneM2M, this use case introduces visual recognition functions (predefined classifier/custom classifier) on the Common Service Entity ready to be, configured trained and used by an Application Entity, subject to access control policies, to identify relevant scenes, objects, and situations within the city and receive the corresponding notifications.

Figure 7.4.1-1 provides a summary of the associated architecture.

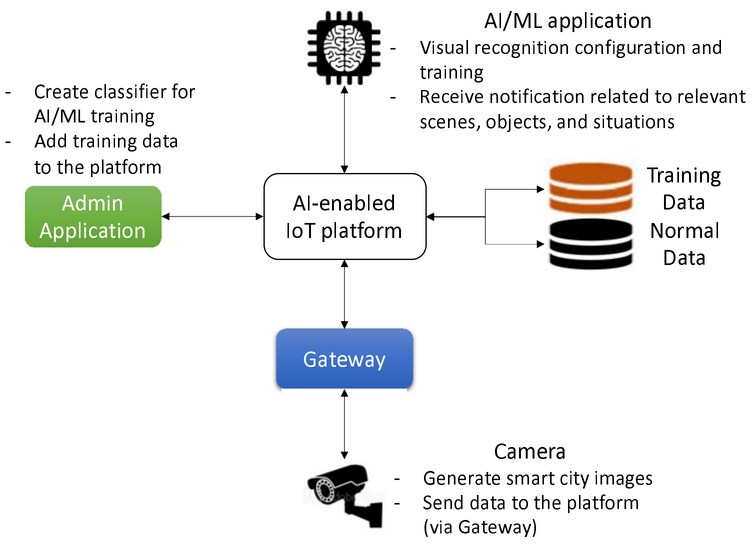
AI/ML application
- Visual recognition configuration and training
- Receive notification related to relevant scenes, objects, and situations
- Create classifier for AI/ML training
- Add training data to the platform
Admin Application
AI-enabled IoT platform
Training Data
Normal Data
Gateway
Camera
- Generate smart city images
- Send data to the platform (via Gateway)

Figure 7.4.1-1: High-level architecture

Figure 7.4.1-2 shows possible resources and attributes to enable pattern recognition service in oneM2M platform.

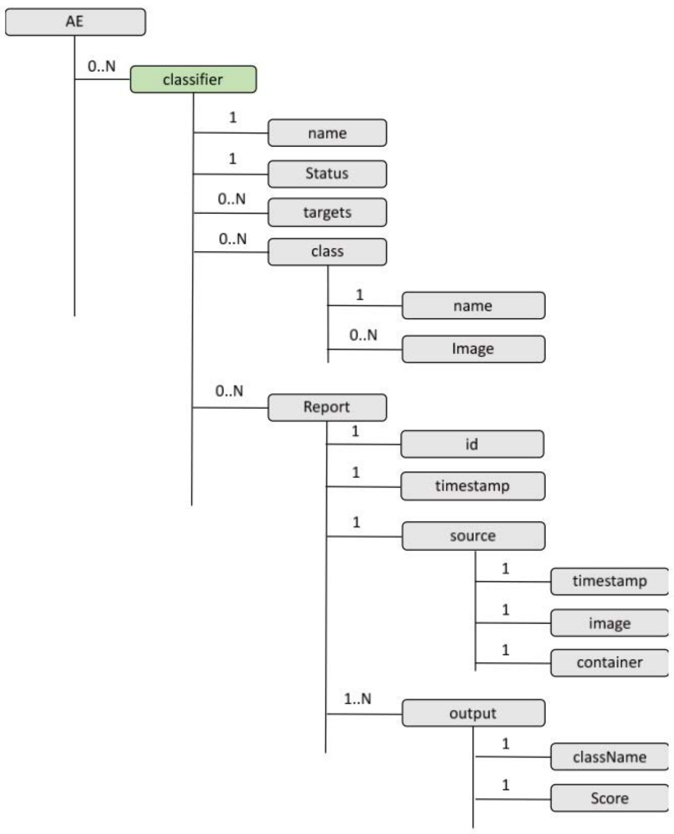

Figure 7.4.1-2: Possible resources and attributes to enable pattern recognition in oneM2M

7.4.2 Source

ETSI SmartM2M TRs for Artificial Intelligence and the oneM2M architecture [i.6] and AI for IoT: A Proof of Concept [i.7].

7.4.3 Actors

- Camera: a camera device that can provide video streaming functions to the IoT platform.
- Edge node: an IoT edge node perform machine learning model to detect a specific pattern specified in a classifier.
- AI-enabled IoT platform: an IoT platform performs detection of patterns based on developed AI model.
- Admin application: an application creates classifiers for AI/ML training and add training data.
- AI/ML application: an application that performs visual recognition configuration and training. Also the application receives notification related to relevant scenes, objects and situation.

7.4.4 Pre-conditions

- There exist a set of data for training.

7.4.5 Triggers

- Camera generates a video stream and sends it to the IoT platform.

7.4.6 Normal Flow

Figure 7.4.9-1 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.

- Step 1: All the relevant applications (Admin Application, Image Recognition Service, and Smart city Camera) are registered to the IoT platform.
- Step 2: The Admin Application creates a classifier A, for example, a car vandalism classifier, at the IoT platform.
- Step 3: The Image Recognition Service, which is an AI/ML capable application, train a model based on the given classifier. Then the service updates the classifier in the IoT platform for pattern detection.

NOTE: If there exist more classifiers to be added, steps 2 and 3 are repeated.

- Step 4: The camera device publishes screenshot image as a content instance inside the container images.
- Step 5: The Image Recognition Service performs pattern recognition for the newly published image and generates visual recognition report.
- Step 6: The Admin Application is notified by the IoT platform with the recognition report.

7.4.7 Alternative Flow

None.

7.4.8 Post-conditions

The AI-enabled IoT platform notifies detected scenes or events to the admin application.

7.4.9 High Level Illustration

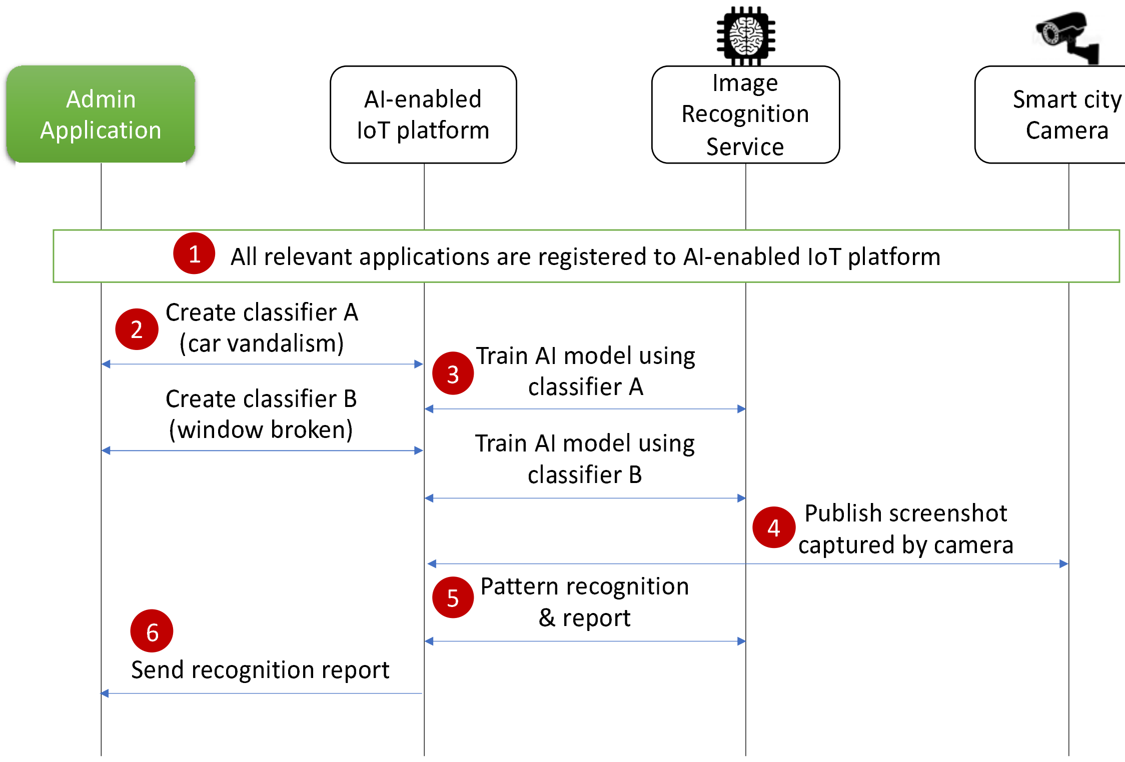
Admin Application
AI-enabled IoT platform
Image Recognition Service
Smart city Camera
1 All relevant applications are registered to AI-enabled IoT platform
2 Create classifier A (car vandalism)
Create classifier B (window broken)
3 Train AI model using classifier A
Train AI model using classifier B
4 Publish screenshot captured by camera
5 Pattern recognition & report
6 Send recognition report

Figure 7.4.9-1: Conceptual sequence of the detection of patterns in video stream

7.4.10 Potential Requirements

1) The oneM2M system will be able to support the creation and management of classifiers for AI/ML application as follows:

   - Predefined-classifier function comes with a predefined and pretrained classifier for Object detection, Object tracking, Semantic Segmentation, Instance Segmentation, etc. from data generated by IoT devices (e.g. smart city camera).

   - Customized classifier that can be generated by an application to support a specific detection function such as visual recognition.

7.5 Use case #5 - Autonomous Operations using Automated Machine Learning

7.5.1 Description

Autonomous Operations is the ability of a system to sense the state of its environment through analysing and learning the collected data (e.g. from IoT devices) to find operational problems changing resource demands and adapt the environment dynamically to resolve issues. Thanks to Artificial Intelligence (AI) & Machine Learning (ML) and IoT technologies, Autonomous Operations enable industry companies to intelligently automate various operations that were previously performed manually.

Autonomous driving is a major field of AI and ML. In order for an autonomous vehicle to drive itself, three functions are essential: recognition, judgment, and control. These functions are where AI/ML is required. In the field of driving environment recognition and autonomous operation using cameras, ML (e.g. Deep Learning) has become the most important technology. When ML is applied to the image input through the camera, static environmental information (car lane, traffic signal, product line signal, location of source product, etc.) and dynamic environmental factors (vehicle, pedestrian, other machines status) can all be detected and classified. For example, in the case of autonomous driving, since the surrounding environment and acquired data are constantly changing, it is important to continuously improve the quality of the model through new data after developing the model for the first time.

AutoML is a process that automates the time-consuming and repetitive task of developing machine learning models. In particular, AutoML is used to effectively develop high-quality models while minimizing developer intervention in the process of algorithm selection and parameter tuning in the data preprocessing process.

Various conditions are required to continuously collect new data, label it, and learn it to develop a better quality model in AutoML. For example, machine learning may be performed when more than a certain amount of labelled data is collected, or machine learning may be performed periodically (e.g. once a day) to generate a new model.

Therefore, in the case of the IoT platform, an automated learning function can be provided in fields such as autonomous driving by supporting the collection of training data, labelling data, and classifying conditions for performing AutoML.

In the use case below, autonomous machine learning for autonomous driving is introduced as an example, and IoT platform can provide the following functions:

- Manage collected dataset for training
- Support criteria when to perform ML and build a model

7.5.2 Source

None.

7.5.3 Actors

- Autonomous vehicle: a vehicle capable of autonomous driving and equipped with sensors and camera.
- AI-enabled IoT platform: An IoT platform stores data from the autonomous vehicle, classifies data for training, and perform ML to build a model under certain conditions.

7.5.4 Pre-conditions

- The autonomous vehicle is registered to IoT platform and continuously store its new measurements from embedded sensors and camera.
- The AI-enabled IoT platform holds a set of good quality training data for the autonomous vehicle.

7.5.5 Triggers

- If the IoT platform is configured to build a model at midnight every Sunday, the time triggers the process for automated ML.

7.5.6 Normal Flow

Figure 7.5.9-1 illustrates the high-level flows of the automated ML for autonomous driving use case, which consists of the following steps:

- Step 1: Initial training dataset is prepared from autonomous driving vehicle.
- Step 2: AI/ML application builds the initial model, Model #1.
- Step 3: Under configured AutoML condition (e.g. every week, the amount of dataset), AI/ML application build Model #2.

7.5.7 Alternative Flow

None.

7.5.8 Post-conditions

The AI-enabled IoT platform has dataset for continuous training of the delivery AI/ML model.

7.5.9 High Level Illustration

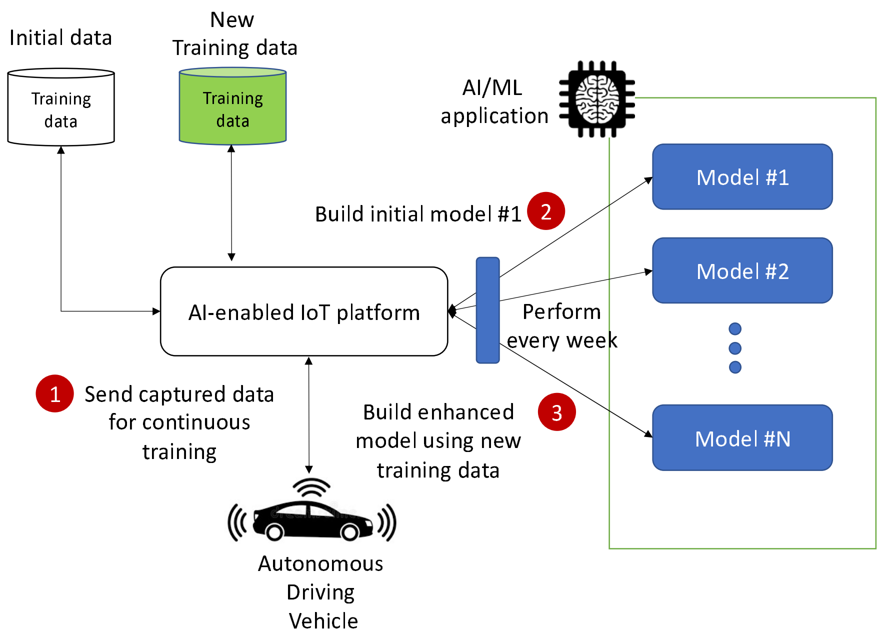

Figure 7.5.9-1: Conceptual diagram of automated ML for autonomous driving

7.5.10 Potential Requirements

1) The oneM2M System will be able to distinguish the dataset that will be trained and has already been trained.

2) The oneM2M System will be able to provide automated machine learning under certain conditions, e.g. building a model every week or when the number of datasets reaches 100.

7.6 Use case #6 - IoT Device Calibration using Machine Learning

7.6.1 Description

In the case of IoT sensors used in autonomous vehicles and smart factories, periodic inspection is required to verify that the required accuracy is continuously maintained. Many IoT sensors are difficult to calibrate and maintain regularly because their working environment is different.

Environmental factors typically influence air temperature measurement using low-cost temperature sensors, e.g. solar radiation, humidity, wind speed, and rainfall. Such environmental factors and wear of sensors are problematic for low-cost air temperature sensors, which lack a radiation shield or a forced aspiration system, exposing them to direct sunlight and condensation.

Drift is a natural phenomenon for sensors. It affects all sensors regardless of the manufacturer. It can be caused by physical changes in the sensor. The drifting of the sensor starts as soon as the sensor leaves the factory. When sensors do drift, then this is often a slow process. Drifting beyond the tolerance of the sensors can occur even before the next calibration.

In such cases, it is possible to check whether the sensor is operating normally by using the measurement values of several nearby high-accuracy reference sensors performing the same operation. For example, more data for ML training is generated by installing the same sensors redundantly in mines, farms, or machines. Periodically, it is possible to learn by using the values of the surrounding sensors, and by changing the calibration values so that the correct values are maintained continuously, it is possible to detect and prevent ageing or error conditions of the sensors.

The basic concept of this use case is to use Machine Learning in a situation where continuous IoT device calibration is required.

In order to support this use case, the IoT platform performs machine learning to generate a calibration value for an IoT device using data collected for a certain period from reference devices. The IoT platform then uses the output from Machine Learning to calibrate the target IoT device. (Optionally, the target device can download the output calibration value into its local memory and do calibration in the device.)

As the IoT device requires calibration regularly or when its measurement deviates from the standard value, the IoT platform can continuously perform Machine Learning for calibration.

In order to support the concept of IoT devices calibration using ML, additional information for maintaining calibration and new behaviours to IoT platforms are required.

7.6.2 Source

None.

7.6.3 Actors

- Low cost temperature sensor: a temperature sensor that requires periodic calibration to provide accurate measurement.
- Reference temperature sensor: a temperature sensor with high accuracy for generating reference measurement.
- IoT platform: an IoT platform stores data for calibration, and performs ML to build a model for calibration.

7.6.4 Pre-conditions

- The low-cost weather temperature sensor and the high-accuracy weather temperature sensors for reference are registered to the IoT platform.
- The IoT platform holds device calibration information and can perform ML for calibration.

7.6.5 Triggers

- If the IoT platform is configured to build a calibration model regularly, for example, on the first day of every month, the time triggers the process for ML for calibration.

7.6.6 Normal Flow

Figure 7.6.6-1 illustrates the high-level flows of the IoT device calibration using ML, which consists of the following steps:

- Step 1: All devices are registered to the IoT Platform. These devices send their measurement to the IoT Platform continuously.
- Step 2: Either the calibration time interval reaches, or the measurement of Sensor-A deviates from the standard range.

- Step 3: Then IoT Platform starts ML using the collected training data from the reference high-accuracy temperature sensors (in this case, Sensor-B, Sensor-C and Sensor-D). The IoT platform also manages various information, such as calibration interval, calibration log, calibration results and standard range, required to perform calibration ML.
- Step 4: The IoT platform performs ML using training values from the reference devices and stores ML results for calibration.
- Step 5: The IoT platform notifies the calibration results to Sensor-A.

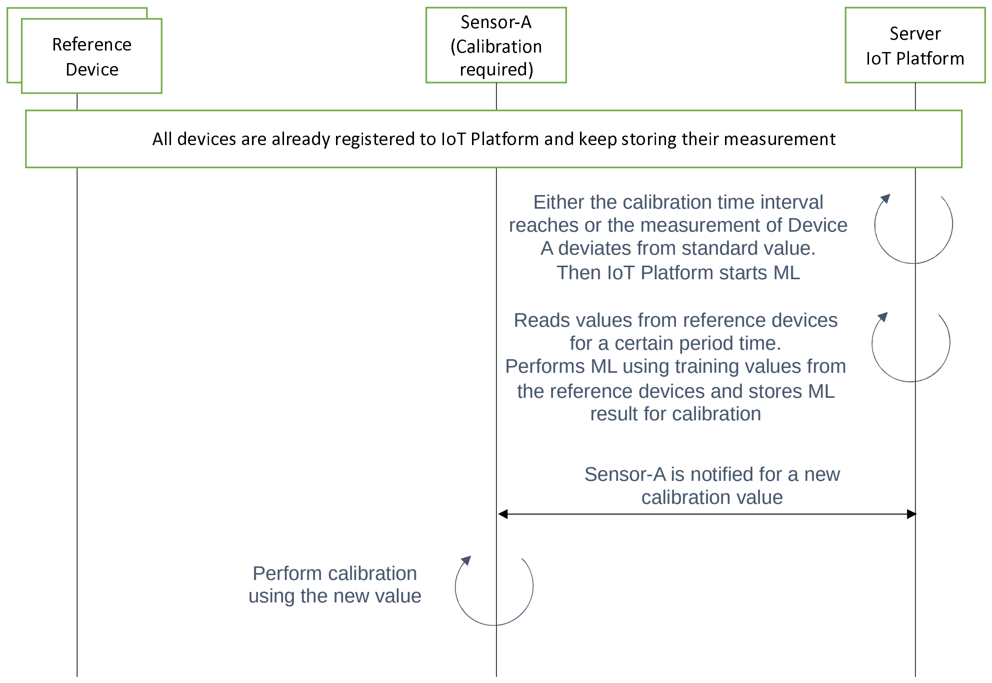
Reference Device
Sensor-A (Calibration required)
Server IoT Platform
All devices are already registered to IoT Platform and keep storing their measurement
Either the calibration time interval reaches or the measurement of Device A deviates from standard value. Then IoT Platform starts ML
Reads values from reference devices for a certain period time. Performs ML using training values from the reference devices and stores ML result for calibration
Sensor-A is notified for a new calibration value
Perform calibration using the new value

Figure 7.6.6-1: A flow for calibrating IoT devices using ML in the server IoT platform

7.6.7 Alternative Flow

None.

7.6.8 Post-conditions

- The low-cost weather sensor is calibrated based on the notified calibration results from the IoT platform after performing ML using a training dataset from reference high-accuracy temperature sensors.
- The sensors with severe drifting can also be substituted with new sensors.

7.6.9 High Level Illustration

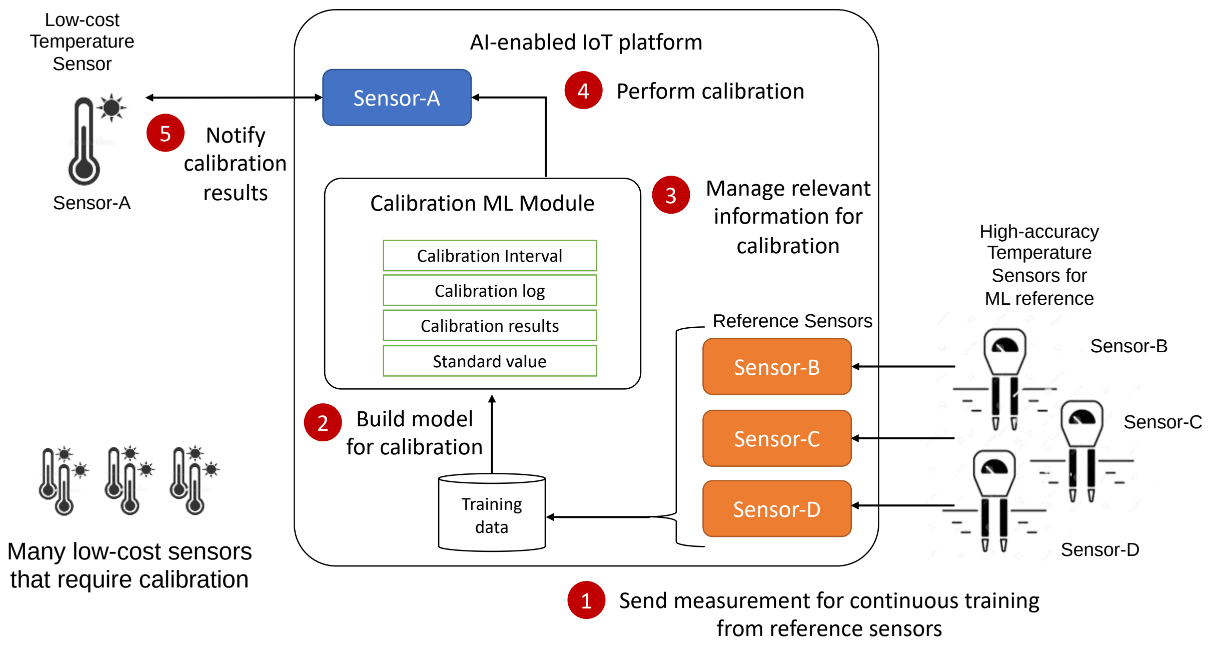
Low-cost Temperature Sensor
Sensor-A
5 Notify calibration results
Many low-cost sensors that require calibration
AI-enabled IoT platform
Sensor-A
4 Perform calibration
Calibration ML Module
Calibration Interval
Calibration log
Calibration results
Standard value
3 Manage relevant information for calibration
2 Build model for calibration
Training data
Reference Sensors
Sensor-B
Sensor-C
Sensor-D
High-accuracy Temperature Sensors for ML reference
Sensor-B
Sensor-C
Sensor-D
1 Send measurement for continuous training from reference sensors

Figure 7.6.9-1: Conceptual diagram of low-cost weather temperature sensor calibration using ML

7.6.10 Potential Requirements

1) The oneM2M System will be able to manage calibration information and training datasets for ML to eliminate or minimize measurement errors from IoT sensors.

2) The oneM2M System will be able to perform ML using training datasets from reference IoT devices and notify calibration results to a target sensor that requires calibration.

7.7 Use case #7 - Dataset creation for AI models

7.7.1 Description

The IoT platform which collects large volume of data from different data sources such as IoT sensors can be leveraged to provide training data and inference data for AI/ML models. The historical data from different sensors can be merged and processed to create AI/ML models. Once the model gets created, the newly stored data on the IoT platform can be merged and provided to the models to make inferences.

Traditionally, the IoT platform provides separate datasets from different data sources. For instance there is a separate data container per sensor. Data scientists get historical data for different sensors and other data sources to merge them by themselves. While dataset pre-processing for model training should be done by existing AI/ML tools, initial dataset creation before pre-processing can be done by IoT platforms since it has all data needed.

When prepare training datasets, there should be several policies needed. A common time (e.g. data creation time) is needed to join different sets of data as a key. Null values could be left empty or can be filled with by any means (e.g. mean or latest value). Preferred data format could be different from original data formats. JSON data instances from sensors could be converted into CSV for preference.

After building AI/ML models, input data for inference/prediction can also be supported by the IoT platform as well. Live periodical data (e.g. temperature measurement) or event-based (e.g. parking spots availability) data is kept provided from different sources. Anytime a source data is given, by merging with latest report from other sources, a new set of data is created so it can be used for running model prediction. If the model is executed periodically like a batch process, then it fetches the inference data regularly. Otherwise, the model can be fed with asynchronous input data over notifications.

7.7.2 Source

None.

7.7.3 Actors

- IoT sensors: provide measurement data to the IoT platform
- IoT platform: stores data from IoT sensors and provide datasets for AI/ML models
- Data scientist: fetches training datasets to build AI/ML models
- AI/ML application: receives input data for model prediction

7.7.4 Pre-conditions

- IoT sensors are registered to the IoT platform

7.7.5 Triggers

None.

7.7.6 Normal Flow

Figure 7.7.6-1 illustrates the high-level flows that shows use case of the AI/ML model support capabilities of IoT platforms:

- Step 1: IoT sensors provide sensor measurements and the IoT platform stores the data.
- Step 2: The data scientist requests to create a model training dataset which is the merged historical data from different sensors.
- Step 3: The IoT platform creates the training dataset with the policies (e.g. join key).
- Step 4: The created dataset is returned in the preferred format (e.g. CSV).
- Step 5: The data scientist build an AI/ML model with the training dataset and deploy it to the application.
- Step 6: The data scientist now requests for inference input data to run predictions on the AI/ML application. Like step 3, policies are given to merge different set of data from sensors. When the IoT platform successfully handles the request, it create a data container to store the inference input.
- Step 7: The data scientist subscribes to the inference data container to send notifications to the AI/ML application for the newly stored data from the sensors.
- Step 8: When the IoT platform stores the sensor data, which are set as part of policies in step 6, it creates the set of inference data and put into the inference data container.
- Step 9: Per subscription created in step 7, the IoT platform sends a notification which includes the newly created inference data to the AI/ML application. Then the application use the data to run prediction.

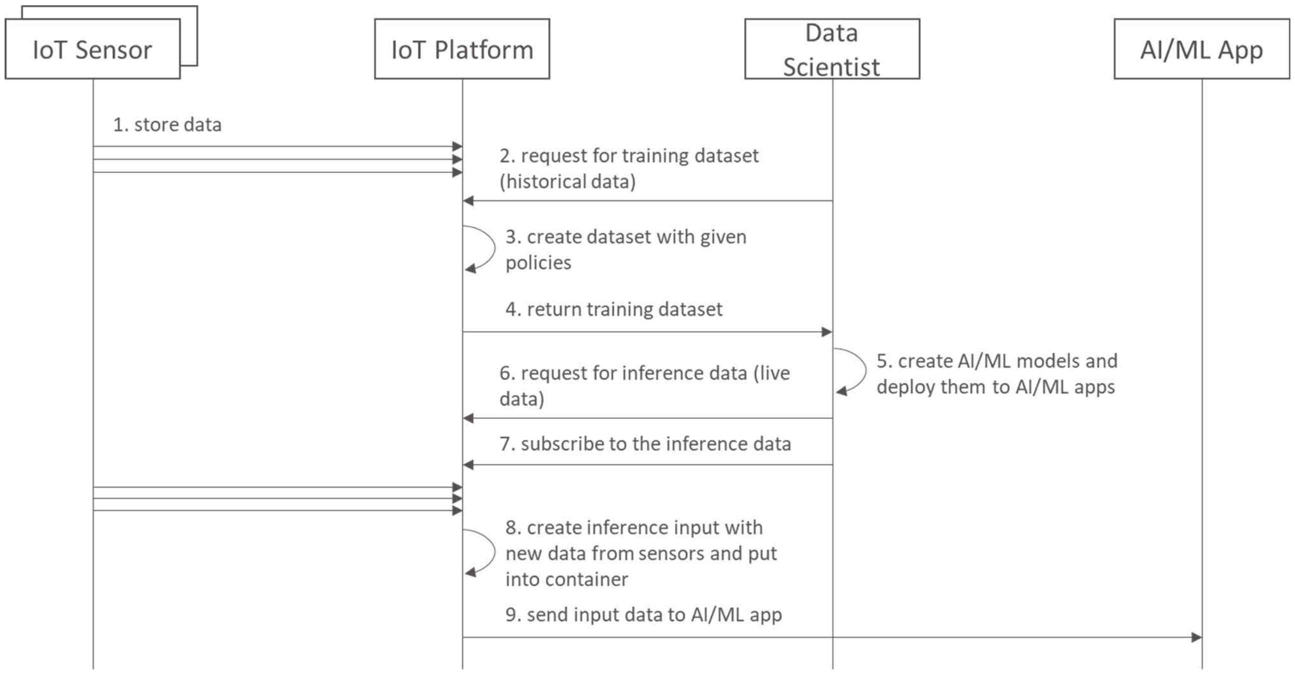
IoT Sensor
IoT Platform
Data Scientist
AI/ML App
1. store data
2. request for training dataset (historical data)
3. create dataset with given policies
4. return training dataset
5. create AI/ML models and deploy them to AI/ML apps
6. request for inference data (live data)
7. subscribe to the inference data
8. create inference input with new data from sensors and put into container
9. send input data to AI/ML app

Figure 7.7.6-1: A flow for AI/ML training dataset creation and inference input data serving

7.7.7 Alternative Flow

None.

7.7.8 Post-conditions

None.

7.7.9 High Level Illustration

Data scientists and their AI/ML applications can leverage the new AI supporting capabilities of the IoT platform. Traditionally the data scientist manually join data with external software tools after fetching data with APIs or database queries. However, with this capability, merging different data can be easily managed for AI/ML model training and prediction. It can be setup with an API call and even can be used with subscription/notification capability.

To create an AI/ML model, the IoT platform provides dataset creation for existing data on the platform, so it is historical data, for model training. For the prediction on the trained model, live data from the sensors can be merged in the similar way for training dataset and be stored on the platform. Then the inference input data can be periodically retrieved for batch prediction applications or can be notified for the event of new sensor data.

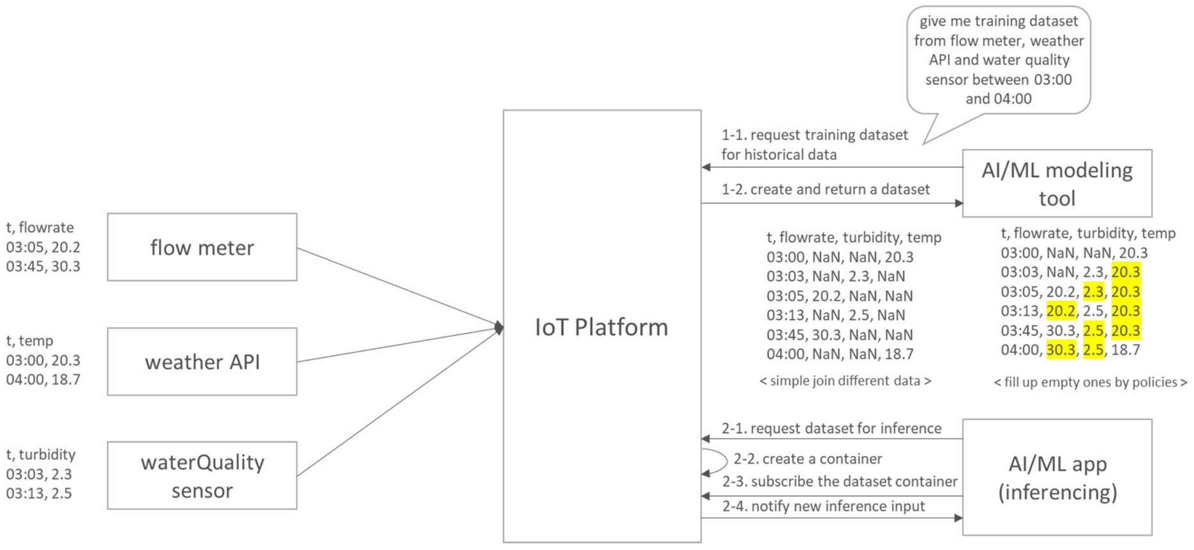

Figure 7.7.9-1: Dataset creation service to support AI/ML models

7.7.10 Potential Requirements

1) The oneM2M System will be able to create datasets using the historical data (e.g. IoT sensor) to train AI/ML models.

2) The oneM2M System will be able to create datasets using the current data (e.g. IoT sensor) to train AI/ML models or make prediction/inference with the trained models.

7.8 Use case #8 - AI model management

7.8.1 Description

A typical use case for on-device AIoT (AI + IoT) services would consist of three steps:

1) prepare training dataset with IoT sensor data

2) build AI models with dataset preprocessing, training and validation

3) deploy the models to an IoT device and perform inferencing

The first step has been described in clause 7.7 as AI dataset preparation use case. The next step is done by data scientists or AI modelers with their expertise (e.g. domain knowledge, algorithm/parameter optimization skills). The use case in this clause describes how the third step can be done with IoT platforms. After all three steps, the inference result can also be stored in the IoT platform so it can be served to AIoT applications.

To elaborate more on the third step in terms of AI model management, the AIoT platform provides AI model repository with model metadata and model deployment capabilities.

7.8.2 Source

None.

7.8.3 Actors

- IoT device: is on-device AI capability so it can run AI models that have been deployed remotely.
- IoT platform: provides AI model repository and model deployment capability.
- Data scientist: trains AI models and store them in the model repository on an AI-enabled IoT platform.

- AI service provider: manages AI models in the repository and deploy a model to an IoT device.

7.8.4 Pre-conditions

- The IoT device stores sensor measuring to the IoT platform.
- The data scientist creates AI models with the sensing data from the devices.
- The IoT device subscribes its model deployment resource, so it can get the deployment information.
- The IoT device can run AI models, while using its sensor readings as inference input data.

7.8.5 Triggers

None.

7.8.6 Normal Flow

Figure 7.8.6-1 illustrates the high-level flows that shows use case of the AI/ML model management and deployment capabilities:

- Step 1: The data scientist creates an AI model repository to store models.
- Step 2: The data scientist registers(stores) an AI model to the model repository.
- Step 3: The AI service provider checks the models in the repository and select a model for a field IoT device.
- Step 4: The AI service provider requests the IoT platform to deploy the selected model to the device.
- Step 5: The IoT platform send the notification for the model deployment. The notification either contains binary model or URI of the model which is stored outside the system (e.g. web storage).
- Step 6: The AI-enabled IoT device runs the AI model on local AI platform (e.g. tensorflow). Then it gets new sensor readings, it prepares the inference input data out of the sensor data and perform the inference.
- Step 7: The IoT device store the inference result to the IoT platform.

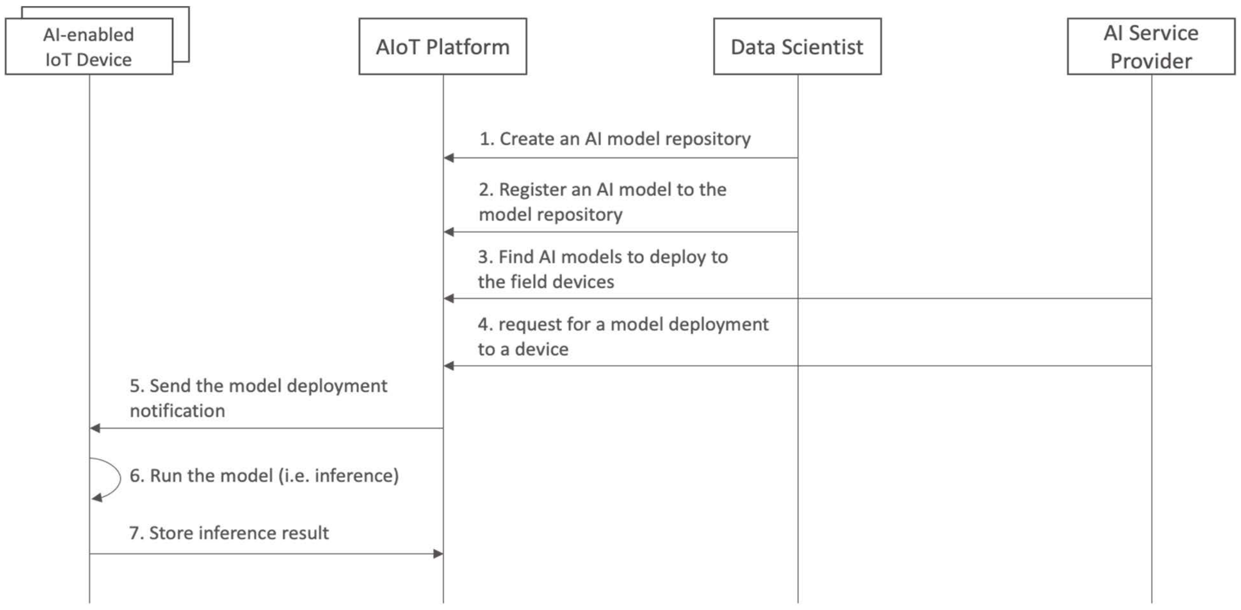
AI-enabled IoT Device
AIoT Platform
Data Scientist
AI Service Provider
1. Create an AI model repository
2. Register an AI model to the model repository
3. Find AI models to deploy to the field devices
4. request for a model deployment to a device
5. Send the model deployment notification
6. Run the model (i.e. inference)
7. Store inference result

Figure 7.8.6-1: A flow for AI/ML model management and deployment

7.8.7 Alternative Flow

In oneM2M standard point of view, the target of model deployment is basically applications while platforms provide AI model management and deployment capabilities. Therefore, the alternative flow to the main flow above is to deploy an AI model to AIoT applications (e.g. AI inference server on cloud computing environment).

7.8.8 Post-conditions

None.

7.8.9 High Level Illustration

In this use case scenario, there are two types of users (e.g. AI service provider and data scientist) and assumably they use the AI service portal which act as an IoT application. The portal provides UI/UX to those users, while interacting with the AIoT platform.

The data scientist, once creates an AI model, uploads the model to the portal so it is registered (stored) on the AIoT platform. Then the AI service provider checks the available models and deploy a model to IoT device(s).

Then the devices receives a new model or model information, it runs the model and perform inferencing with its own sensor measurements.

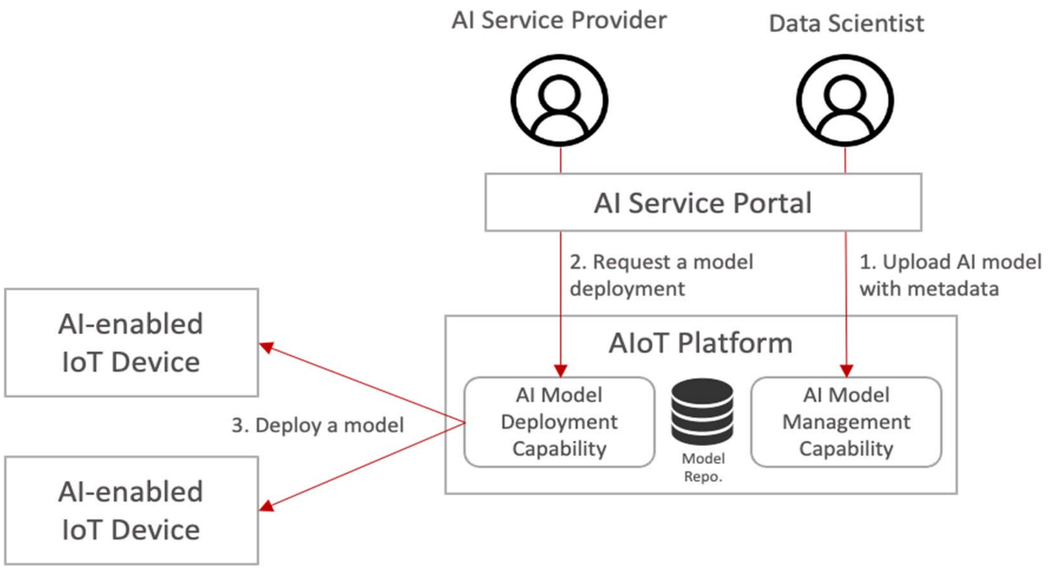

Figure 7.8.9-1: AI model management for AI-enabled IoT devices

7.8.10 Potential Requirements

1) The oneM2M System will be able to manage AI/ML models with model metadata.

2) The oneM2M System will be able to support an AI/ML model deployment to IoT devices (e.g. Edge/Fog nodes) and IoT applications.

8 Requirement Analysis of the Current oneM2M System to Support AI/ML

8.1 Overview

Table 8.1-1 presents a collection of potential requirements and their corresponding use cases specified in the previous clauses.

Table 8.1-1: Collection of potential requirements

Use case	Potential requirements
Use case #1. Data augmentation for autonomous driving	The oneM2M System will be able to handle data augmentation requests for AI/ML purposes.
	The oneM2M System will be able to generate augmented data resources from a given source data and data augmentation technique.
	The oneM2M System will be able to manage data for AI/ML purposes such as model training and augmentation of training dataset.
Use case #2. Last mile delivery	The oneM2M System will be able to manage structured and unstructured data for training, for example, preprocessing data, describing data and inferring meaning.
	The oneM2M System will be able to update trained AI/ML model according to continuous measuring data e.g. location, time series and historical data.
	The oneM2M System will be able to provide a classification function (e.g. split data into two parts, training and validating) in supervised Machine Learning.
Use case #3. Smart virtual store using metaverse	The oneM2M System will be able to synchronize between real and virtual world devices
	The oneM2M System will be able to enable Edge/Fog Nodes to run AI/ML models to retrieve information from the real world
Use case #4. Detection of patterns in video streams	The oneM2M system will be able to support the creation and management of classifiers for AI/ML application as follows: - Predefined-classifier function comes with a predefined and pretrained classifier for Object detection, Object tracking, Semantic Segmentation, Instance Segmentation, etc. from data generated by IoT devices (e.g. smart city camera). - Customized classifier that can be generated by an application to support a specific detection function such as visual recognition.
Use case #5. Autonomous operations using automated machine learning	The oneM2M System will be able to distinguish the dataset that will be trained and has already been trained.
	The oneM2M System will be able to provide automated machine learning under certain conditions, e.g. building a model every week or when the number of datasets reaches 100.
Use case #6. IoT device calibration using ML	The oneM2M System will be able to manage calibration information and training datasets for ML to eliminate or minimize measurement errors from IoT sensors.
	The oneM2M System will be able to perform ML using training datasets from reference IoT devices and notify calibration results to a target sensor that requires calibration.
Use case #7. Dataset creation for AI models	The oneM2M System will be able to create datasets using the historical data (e.g. IoT sensor) to train AI/ML models.
	The oneM2M System will be able to create datasets using the current data (e.g. IoT sensor) to train AI/ML models or make prediction/inference with the trained models.
Use case #8. AI model management	The oneM2M System will be able to manage AI/ML models with model metadata.
	The oneM2M System will be able to support an AI/ML model deployment to IoT devices (e.g. Edge/Fog nodes) and IoT applications.

The potential requirements can be further classified into four issues, i.e. (1) managing training data for AI/ML, (2) managing AI/ML input data, (3) creation of AI/ML model, and (4) managing AI/ML model (see Figure 8.1-1). Each key issue covers detailed items to be resolved as follows:

1) Training data management for AI/ML

- Data augmentation
- Training dataset
- Reuse training data

2) Input data management for AI/ML

   - Structured and unstructured data
   - Calibration data

3) Creation of AI/ML model

   - Trained model
   - Predefined and customized classifiers

4) Managing AI/ML model

   - Distributed training model
   - Continuous learning
   - AI/ML model on Edge/Fog

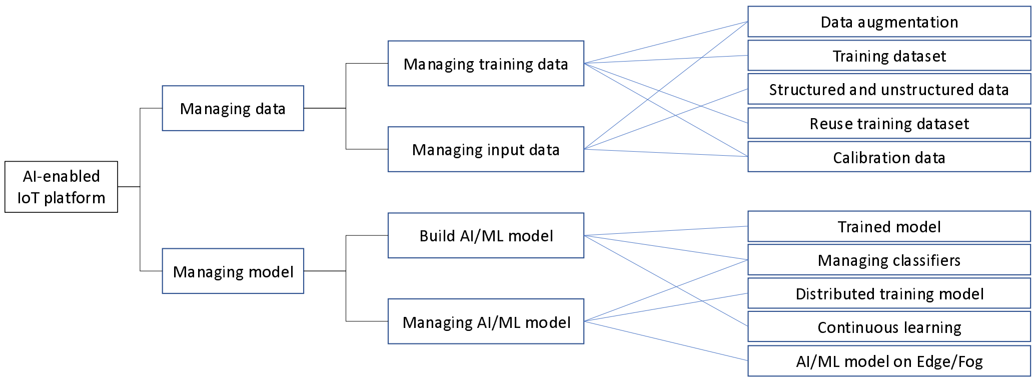

Figure 8.1-1: Classify potential requirements to identify key issues

8.2 Key Issue & Possible Concept 1 - Data augmentation

8.2.1 Key Issue

Many AI applications use data collected in IoT platforms. However, AI services often do not produce accurate results due to insufficient training data. Data augmentation in AI/ML, especially supervised machine learning, is a technique used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.

For example, an AI application trains its model based on images stored in various resources in an IoT platform. The AI application can enhance the existing dataset if there is a small dataset of images to build a better model. Typically, the AI application can take a small dataset of images and transform the objects to different sizes by zooming in or zooming out, flipping them vertically or horizontally or changing the brightness, whatever makes sense for the object.

Currently, the oneM2M platform does not provide any features supporting AI/ML applications to have increased data for AI/ML training. If the oneM2M platform support such data augmentation features, AI applications can easily build a model based on an augmented dataset.

8.2.2 Possible Concept

For convenience, this clause considers that there exists a small dataset of images in the oneM2M platform for AI/ML training. Then the oneM2M platform provides features to support the following data augmentation to its image resources:

- Flipping: flipping the image vertically or horizontally

- Rotation: rotates the image by a specified degree
- Cropping: objects appear in different positions in different proportions in the image
- Zoom in, Zoom out
- Changing brightness, contrast, colors, etc.

For each augmentation type, the AI application requires different configuration parameters. For example, the following information is required to perform data augmentation:

- Source resource URI that contains a target image
- Types of data augmentation (e.g. flipping, rotation, and cropping)
- Parameters for the selected data augmentation
- A destination resource URI to store a dataset of augmented images

For example, a data augmentation with the rotated image type can be performed by specifying a rotation degree and the total number of images to generate. Suppose there is a request from an AI/ML application to perform image rotation with input parameters, rotation 1 degree and generate 90 augmented images. In that case, the oneM2M platform generates 90 images by rotating the source image by 1 degree 90 times.

This can be done by introducing a new resource called the <`dataAugmentation`> to hold the information required to perform data augmentation. The <`dataAugmentation`> resource can have the following attributes (see Figure 8.2.2-1):

- Type: type of data augmentation (e.g. resize, crop, rotate)
- Source resource: a resource that contains the raw image
- Augmentation parameter: required parameters for the selected augmentation type
- Target resource: a resource or a set of resources to store generated images

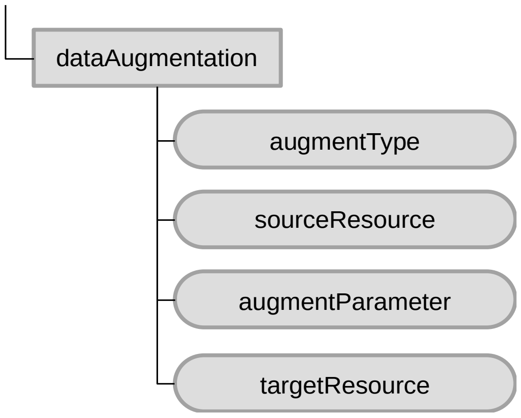

Figure 8.2.2-1: An example structure of [*dataAugmentation*] resource

Figure 8.2.2-2 shows a high-level procedure that an AI/ML application requests to perform data augmentation to enhance its training dataset.

- Step 1: AI application (oneM2M AE) sends request to the <`dataAugmentation`> resource.
- Step 2: CSE stores received input to the <`dataAugmentation`> resource.
- Step 3: Based on the given parameters, CSE tries to get the source image, apply the given data augmentation technique (e.g. resize), and generate target resources containing generated augment images.

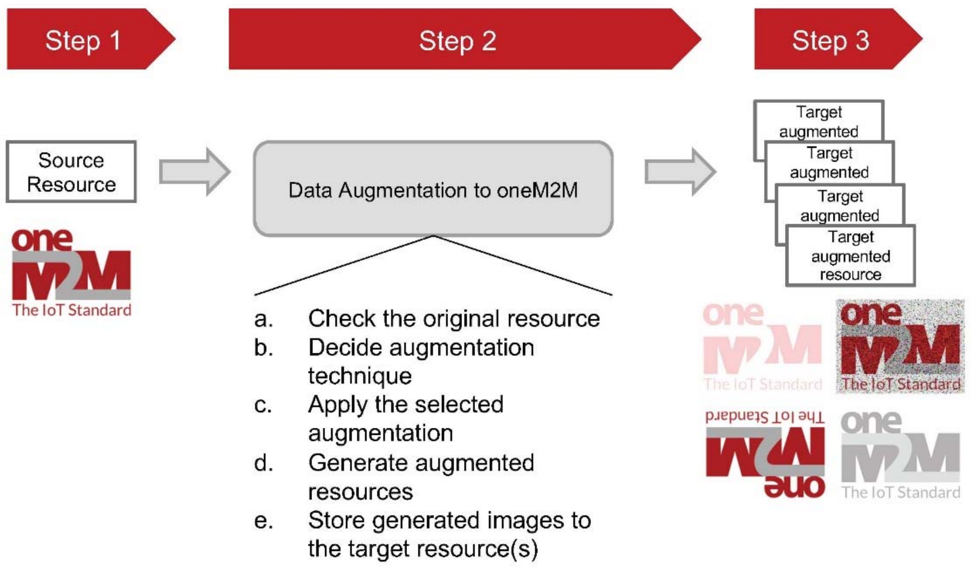
Step 1
Step 2
Step 3
Source Resource
Data Augmentation to oneM2M
a. Check the original resource
b. Decide augmentation technique
c. Apply the selected augmentation
d. Generate augmented resources
e. Store generated images to the target resource(s)
Target augmented
Target augmented
Target augmented
Target augmented resource

Figure 8.2.2-2: High-level procedure to perform data augmentation

8.3 Key Issue & Possible Concept 2 - Data labelling

8.3.1 Key Issue

Many Artificial Intelligence (AI) and Machine Learning (ML) applications use data collected in IoT platforms to train their model. IoT platform (including oneM2M) is a place holder to collect and manage various data (e.g. image, text, and sensory).

Data labelling is an essential step in a supervised machine learning task. Data labelling is the process of identifying raw data (e.g. images, text files, videos) and adding one or more meaningful and informative labels to provide a context for data. Data labelling is a task that requires much manual work. If an IoT platform provides a means to support providing data labelling, developers can save time and resources. For example, labels on data indicate whether data contains the temperature of a room or if an x-ray contains a tumour.

At the moment, there is no single standard format to annotate raw data. Below are several available annotation formats used by AI/ML developers:

- COCO: COCO has five annotation types: object detection, key-point detection, stuff segmentation, panoptic segmentation, and image captioning. The annotations are stored using JSON.
- Pascal VOC: Pascal VOC stores annotation in an XML file.
- YOLO: In YOLO labelling format, a .txt file with the same name is created for each image file in the same directory. Each .txt file contains the annotations for the corresponding image file (i.e. object class, object coordinates, height and width).

AI/ML developers can define a customized labelling format depending on which data needs annotation.

Here are several annotation types used in AI/ML:

- Bounding boxes: use rectangular boxes to define the target object's location.
- Polygonal segmentation: complex polygons are used to define the shape and location of the object in a much more precise way.
- Semantic segmentation: a pixel-wise annotation technique, where every pixel in the image is assigned to a class.
- 3D cuboids: similar to bounding boxes but with additional depth information about the object.

- Lines and splines: annotation is created by using lines and splines.
- Key-point and landmark: detect small objects and shape variations by creating dots across the image.
- Customization: various annotations can be used depending on the data type. For example, if the network packet is data, header field information (base address, offset, field description) can be used for labelling.

oneM2M currently provides semantics and ontology-related functions. These functions have the potential to be used for artificial intelligence and machine learning but are not directly utilized for training data. Suppose the oneM2M platform provides a new function of data labelling used for AI/ML using the semantics and ontology functions. In that case, the reuse of the learning data becomes possible, and AI application developers can create more accurate models.

8.3.2 Possible Concept

Suppose oneM2M systems support functions that allow developers to annotate labelling information to training datasets using ontologies. In that case, an AI/ML data labelling tool can create and manage resource(s) to annotate labelling information to training data (set).

This can be done by introducing a new resource called the <`dataLabel`> to hold the information required to perform data labelling. The <`dataLabel`> resource can have the following attributes (see Figure 8.3.2-1):

- `trainingData`: confirms that this data is for training
- `labelingType`: describes labelling type, e.g. square, polygon, line.
- `labelAnnotationFormat` : there exist many labelling format such as COCO, YOLO, Pascal VOC, custom
- `labelAnnotationContents`: contains actual annotation contents following the given format (e.g. XML or JSON)
- `refOntology`: reference ontology used in data label annotation

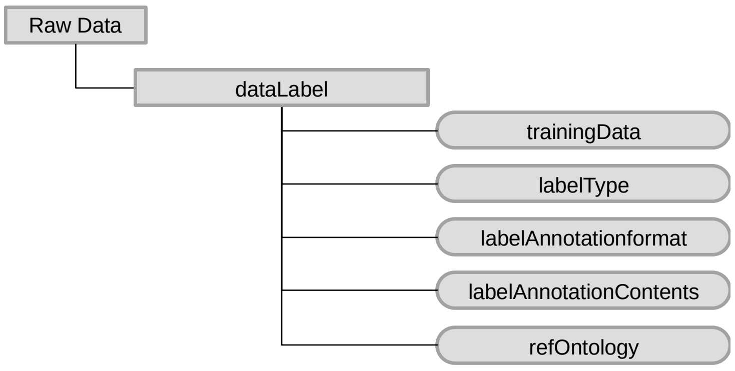

Figure 8.3.2-1: An example structure of [*dataLabel*] resource

For this purpose, it is assumed that an oneM2M platform holds dataset for AI/ML training to build a model (see Figure 8.3.2-2). First, a labelling annotation tool (oneM2M application) discover resources for training. Then the labelling tool creates a resource(s) for data labelling. The labelling application requests to update data labelling resources. Another AI/ML application then uses the training dataset with label information and builds a model for AI/ML service.

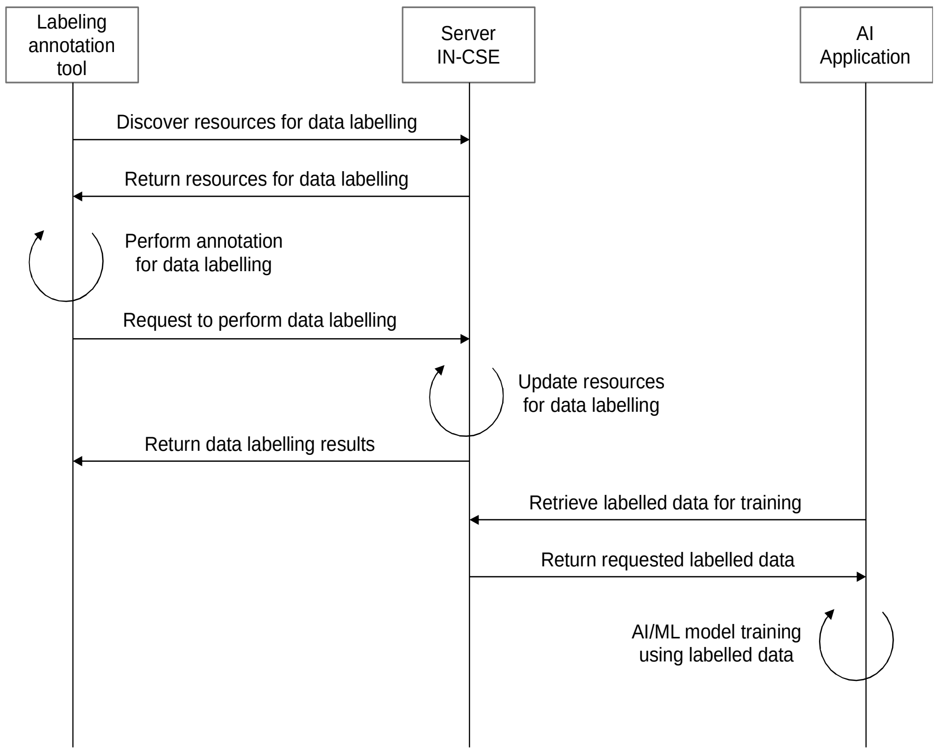
Labeling annotation tool
Server IN-CSE
AI Application
Discover resources for data labelling
Return resources for data labelling
Perform annotation for data labelling
Request to perform data labelling
Update resources for data labelling
Return data labelling results
Retrieve labelled data for training
Return requested labelled data
AI/ML model training using labelled data

Figure 8.3.2-2: An example procedure for data labelling

8.4 Key Issue & Possible Concept 3 - Sensor calibration

8.4.1 Key Issue

The basic concept of the use case introduced in clause 7.7 is to use Machine Learning in a situation where continuous IoT device calibration is required. In order to support this use case, the IoT platform is required to perform machine learning to generate a calibration value for an IoT device using data collected for a certain period from reference devices. The IoT platform then uses the output from Machine Learning to calibrate the target IoT device. (Optionally, the target device can download the output calibration value into its local memory and do calibration in the device.) As the IoT device requires calibration regularly or when its measurement deviates from the standard value, the IoT platform can continuously perform Machine Learning for calibration.

Currently, the oneM2M platform does not provide any features supporting AI/ML and calibrating deviated measurement data. If the oneM2M platform support such sensor calibration features, various IoT service players which require accurate measurement can easily provide high performance and quality service.

8.4.2 Possible Concept

Possible solution should support the following requirements (see clause 7.7):

1) The oneM2M System will be able to manage calibration information and training datasets for ML to eliminate or minimize measurement errors from IoT sensors.

2) The oneM2M System will be able to perform ML using training datasets from reference IoT devices and notify calibration results to a target sensor that requires calibration.

This can be done by introducing a set of attributes to a new resource (for example, <*devCalibration*>) or the <*flexContainer*> resource to hold the information required to perform continuous sensor calibration using AI/ML. The required attributes are as follows:

- *calibrationInterval*: defines intervals to perform machine learning for calibration

- *refCalDevices*: list of reference devices to perform ML for calibration. IoT platform uses the value of these referenced devices as training data for calibration
- *mlModel*: defines machine learning model for calibration. IoT platform aware which model to use to perform machine learning for calibration
- *calibrationLogs*: provides information when performed calibration previously
- *calibrationValue*: stores the result of machine learning for calibration. IoT device uses this value for calibration until next calibration
- *standardValue*: defines acceptable values for the measurements

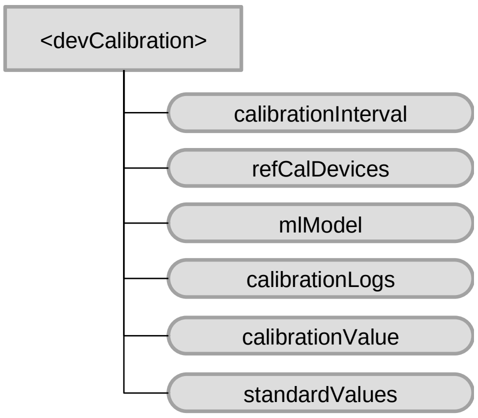

Figure 8.4.2-1: An example structure of [*devCalibration*] resource

With such attributes, the oneM2M IoT platform can support calibrating data measurements that drift over time.

In this case, an IoT platform can be considered to manage IoT devices that require data calibration. Those IoT devices are able to send their measurement to the IoT platform continuously based on the calibration value. When the calibration time interval reaches based on the *calibrationInterval* attribute, or the measurement deviates from the standard range stored in the *standardValues* attribute, the IoT platform performs AI/ML based on the model stored in the *mlModel* attribute. Then the IoT platform starts AI/ML using the collected training data from the referring high-accuracy temperature sensors based on the *refCalDevices* attribute. The IoT platform generates an updated calibration value and stores it in the *calibrationValue* attribute. IoT devices that require this updated calibration value get notifications about it and use it to generate calibrated measurements.

8.5 Key Issue & Possible Concept 4 - AI/ML model management

8.5.1 Key Issue

The artificial intelligence service basically requires data for learning, an algorithm to make an accurate model, and parameter information for performing the algorithm. In the existing IoT platform, many datasets that can be used for AI services are stored and managed, but the functions necessary to build and use AI models are not provided. Therefore, most AI services were developed and provided on a separate, independent platform. In order to provide common service functions required for AI services through the IoT platform, various information to build and manage AI/ML models should be stored in the IoT platform.

However, the oneM2M platform does not provide any features supporting AI/ML services. If the oneM2M platform supports AI/ML-related functions, then AI/ML developers can use the AI-enabled oneM2M platform to build their required services.

8.5.2 Possible Concept

Possible solution should support the following requirements (see clause 7.3):

1) The oneM2M System will be able to manage structured and unstructured data for training, for example, preprocessing data, describing data and inferring meaning.
2) The oneM2M System will be able to update trained AI/ML model according to continuous measuring data e.g. location, time series and historical data.
3) The oneM2M System will be able to provide a classification function (e.g. split data into two parts, training and validating) in supervised Machine Learning.

If oneM2M systems support such functions, an AI application can create a resource(s) to build a model for its prediction as follows:

- Assumption 1: an oneM2M platform holds all the data for training and has data for prediction
- Assumption 2: an oneM2M platform knows a list of ML algorithms to perform
- An AI/ML application creates a resource to build an AI/ML model to solve or support a specific domain application
- The AI/ML application triggers to build the model
- The AI/ML application downloads the trained model and performs prediction

This can be done by introducing a set of attributes to a new resource called <*mlManagement*>) to hold the information required to manage AI/ML functions in the oneM2M platform. The required attributes are as follows:

- *datasetTrain*: list of resources storing train data
- *datasetValidation*: list of resources storing validation data
- *datasetTest*: list of resources for testing a model
- *datasetRatio*: ratio of ML dataset (for example three tuple of percentage)
- *selectedModel*: an ML algorithm that represents the model to perform
- *modelParameters*: parameters used by the selected algorithm
- *trainedModel*: the result model (e.g. executable software) after training and validation
- *trigerBuildModel*: a triggering value to start building a model (assumption is to have proper values on *datasetTrain*, *datasetValication*, *datasetTest*, *dataSetRatio*, and *selectedModel*)

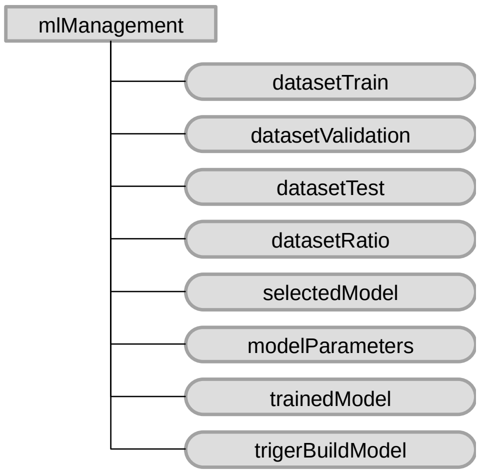

Figure 8.5.2-1: An example structure of [*mlManagement*] resource

Figure 8.5.2-1 shows a high-level procedure that enables AI/ML features using the <mlManagement> resource. This allows an AI application to create a resource that manages AI/ML model and perform AI/ML function as a common service via the oneM2M platform.

First, an AI/ML application (oneM2M application) creates a resource to perform and manage AI/ML function. The AI/ML application configures the <*mlManagement*> resource with proper information to build and manage an AI/ML model. The application also subscribes to the <*mlManagement*> resource to be notified when an AI/ML model is built or updated. The AI/ML application requests to perform training to build a model via setting the *triggerBuildModel* attribute. The oneM2M platform performs AI-enabled CSF to build a model based on the given attributes in the <mlManagement> resource. For example, the CSF uses training dataset specified in the *datasetTrain* attribute to train a model using an AI/ML algorithm specified in the *selectedModel* attribute. The trained model is then validated and tested via dataset stored resources in the *datasetValication* and *datasetTest* attributes, respectively. The AI-enabled CSF then builds a model that can be used by the AI/ML application. This update is notified to the Application for the AI/ML service.

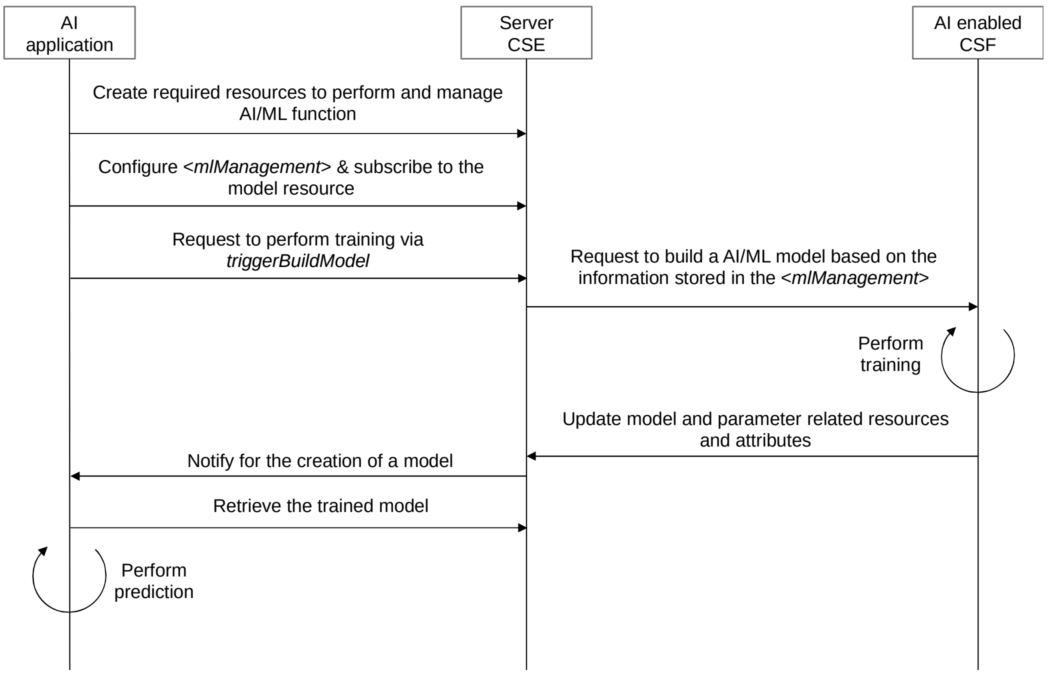
AI application
Server CSE
AI enabled CSF
Create required resources to perform and manage AI/ML function
Configure <mlManagement> & subscribe to the model resource
Request to perform training via triggerBuildModel
Request to build a AI/ML model based on the information stored in the <mlManagement>
Perform training
Update model and parameter related resources and attributes
Notify for the creation of a model
Retrieve the trained model
Perform prediction

Figure 8.5.2-2: An example procedure to enable AI/ML feature to oneM2M using the [*mlManagement*] resource

9 Conclusions

The present document analyses existing AI/ML technologies that can impact IoT service layer platforms, especially the oneM2M IoT System. Moreover, the present document introduces various potential AI/ML service use cases that employ data collected in the oneM2M system. As a result of such analysis, the present document identifies potential requirements and key features to enable AI/ML in the oneM2M system.

The followings are eight use cases identified in the present document:

- Use case #1. Data augmentation for autonomous driving
- Use case #2. Last mile delivery
- Use case #3. Smart virtual store using metaverse
- Use case #4. Detection of patterns in video streams
- Use case #5. Autonomous operations using automated machine learning
- Use case #6. IoT device calibration using ML

- Use case #7. Dataset creation for AI models
- Use case #8. AI model management

The oneM2M system currently does not support the identified AI/ML related requirements. Therefore, the high-level concepts discussed in the present document need to be investigated further. These investigations will facilitate further normative work, resulting in a oneM2M technical specification in Stage 2.

Annex A: Collection and Analysis of Use Cases from ETSI STF 601

A.1 Introduction to ETSI STF 601

ETSI STF 601 has published ETSI TR 103 778 [i.8] containing use cases where the IoT data and services require data usability. The scope of ETSI STF 601 focused on the data generated and processed by IoT devices and platforms and consumed by humans or machines (ML/AI) users. As discussed in a previous RDM meeting, this contribution brings the use cases into the present document to take advantage of the work done by ETSI STF 601 and for further exploration in this WG.

A.2 Collection of Use Cases from ETSI STF 601

The following use cases are based on several sources, including research, innovation road mapping, and pilot initiatives. The analysis that has already been completed by the Standards Development Organization could be another source of use cases. An example of such analysis may be found in the RDM-2021-0082R01-ML_AI_UseCasesFrom_STF601 (Use cases and applications) [i.9].

One approach to analysing potential service-layer requirements coming from the use of AI to IoT is use-case analysis. Because some AI modules apply to numerous use cases, this research also considers general-purpose capabilities to identify potential 'cross-cutting' requirements. Table A.2-1 presents a collection of use cases to be investigated.

Table A.2-1

S.No.	Category	Title
1	AI for Healthcare Systems	Adoption of an AI-based system for supporting patients
2		Electronic Health records
3	AI for Disaster and Emergency Management	Automatic direct emergency call from IoT device
4		Emergency services teams accessing pre-deployed IoT devices
5	AI for Industry Management	Monitoring of industrial manufacturing equipment
6		Monitoring of industrial manufacturing products
7	AI for Smartphone Security	Intelligent Software Rollouts
8	AI Driven Networks	Policy-based network slicing for IoT security
9	AI for Agriculture Management	Fertilization/ Irrigation/ Pest management service
10	AI for Intelligence in Vehicles	Vehicle Diagnostic & Maintenance Report
11		Smart Parking
12	AI for Intelligent Energy Systems	Energy optimization using AI
13	AI for Safety	Smart safety of workers at building construction site
14		Machine socialization
15	AI for Operational Efficiency in Stock Management	Retail inventory management
16		Vending Machines
17	AI for Labor Welfare	Crowd Safety and Security
18	AI for Automation in Fault Tolerance	Predictive maintenance and fault tolerance
19		Low-connection environments
20	AI for Smart Cities	Smart Lighting
NOTE 1: The term "Use Case" refers to a set of application-level features. NOTE 2: The "description" concentrates on specific characteristics of the use case. In particular: A use case is not meant to be adequately documented and to cover all of the aspects that may be connected with it. It is an illustration of a subset of specific components related to the impact analysis and the Proof-of-Concept. NOTE 3: The "important qualities" refer to the main AI-related features/technologies/that AI could bring to Use Case development.		

A.3 Use Case Analysis

A.3.1 Adoption of an AI-based system for supporting patients

Ref: Use Case A.1 in ETSI STF 601

Description: Diet and exercise are essential for living a long and healthy life. Indeed, tracking users' eating habits will aid in the prevention of many pathologies, chronic diseases, and cognitive decline. However, motivating people to establish and maintain better lifestyles is a difficult task. Diverting a user away from harmful foods, for example, needs supplying acceptable alternatives. These options can be based on logical food attributes or more abstract ones like the taste. This would be accomplished through a personal device interface that could collect data from the user, wearable sensors, and contextual data. Adapted representations, such as text, speech, video, or graphical alerts, are also used to deliver real-time feedback and notify the user with convincing and motivational messages.

To express all important information such as the number of calories and nutrients ingested in a specific meal, diet-oriented concepts and ontologies in the food domain are required. As a result, a complete description of each food, as well as its qualities, will be provided. Additionally, an intelligent platform would be deployed that would allow professionals to recommend monitoring norms and guidelines for a healthy lifestyle that users should follow to maintain their physical and emotional health. External factors, such as a person eating too much food or performing too little physical activity, would act as a trigger for the system.

AI/ML Impact: However, the user provides details about consumed nutrients, and the completed activity using a smartphone app and/or devices such as smartwatches. The system stores the data into a knowledge base and executes machine learning models for identifying possible undesired behaviours by trying to match the user's data with the instructions given in advance by the physician. The system keeps a record of the activities and modifies the policies used to generate the next feedback if the same unwanted event is recognized.

Analysis: However, if IoT technology is not developed properly, deployed carefully, and utilized sensibly, it might introduce new safety issues. Data integrity problems in patients' Electronic Health Records (EHRs) and other health IT systems as a consequence of erroneous or missing data are a critical issue in the healthcare industry that can have a significant impact on patients' (and users') health.

Data integrity concerns existed with paper medical records as well, but as EHRs grow more interoperable and hackable, inaccurate data is more readily available, easier to spread, and harder to erase. Data integrity issues include data from one patient appearing in another's record, missing or delayed data transfer, and clock synchronization difficulties between medical equipment and systems. These flaws can cause the AI engine to make incorrect inferences or classifications, resulting in inaccurate information being sent to clinicians and, even worse, putting patients' health at risk.

A.3.2 Electronic Health records

Ref: Use Case A.2 in ETSI STF 601

Description: Diagnostic and preventative medicine necessitates having access to a valid and accurate record of a patient's health for as long as possible. Thus, while health gadgets (heart rate monitors, blood pressure monitors, and so on) may be claimed to be vital, they are only critical if the measurements they take are recorded and as mentioned in the use case statement, can identify the context of the data with some accuracy. As stated expressly in Directive 2011/24/EU [i.10] on patients' rights in cross-border healthcare, health records are required to supersede international borders. A health record is a composite document and one of the challenges in defining a health record is establishing the border. Information is essential to build context in the domain of diagnostic medicine. For example, many diseases in their early stages share symptoms, and correctly attributing a symptom to a cause may be the difference between survival and death.

A health record does not have a set beginning and ending time. While a health record exists for an individual from birth, there are aspects of the individual's health that are directly tied to the parents (e.g. genetics) and to the period in the womb that will be linked to the individual's record. Records of health professionals, sites where medical interventions occur (e.g. hospital, clinic), prescriptions prescribed, and so on are all linked to the individual's record.

AI/ML Impact: The AI-based system operates in the background, continually constructing a patient context from the content of personal electronic health records and integrating it with external knowledge to assist physicians in their job (e.g. early diagnosis). The physician enters new information into a patient's electronic health record. The AI engine refreshes the patient's digital twin with new information and runs a check method to uncover probable circumstances that lead to the start of a disease. The AI engine recommends to the physician that more verifications be undertaken. The doctor saw that the sugar level in the blood is near to the attention level and prescribed a diet to the patient to prevent nutritional exacerbations.

Analysis: However, concerns with the format of such information or network outages, the information supplied by doctors is not correctly kept in the system. When IoT data is used, potential problems are associated with IoT devices that do not offer data at the fine-grained level necessary or do not supply data at all owing to various malfunctions. Furthermore, the sensors are susceptible to false reading difficulties, which might impair the overall data usage of the patient's PHR.

A.3.3 Automatic direct emergency call from IoT device (Public and Emergency Services Use Cases)

Ref: Use Case B.1 in ETSI STF 601

Description: When a smoke or temperature detector at a remote area (forest, remote facility, etc.) delivers an emergency alert in case of fire, this use case applies. The IoT gadget sends the emergency call (most likely to local authorities), which sends an alarm to the operator, who then dials 911. However, when the operator tries to check that a fire has ignited, he is unable to conclude that the notification was false. A secondary sensor raises an identical alert thirty minutes later. The operator can now confirm the alarm, but the fire has expanded to tremendous proportions, making it even more difficult to extinguish, and the facility is destroyed. This scenario is suitable for a massive building or a remote location. This use case does not directly relate to someone dialing 911 from their house on a smartphone.

AI/ML Impact: The emergency event is detected by the IoT sensor. The sensor generates and sends an emergency message to the IoT platform. The IoT platform receives the emergency message call, recognizes it via a machine learning model as an emergency message, and notifies the operator. The operator retrieves and analyses the communication before contacting the necessary public safety services and spreading the emergency message. Upon timely action, the fire department arrives early and can control the fire situation.

Analysis: Data in the regular flow may be compromised if the system is not properly designed (sensor locations are erroneously specified) or if system maintenance is insufficient to ensure that the sensors are in perfect working order and do not trigger false positives. In the case of a typical flow failure (failed completion), the operator was provided two sets of meaningless data, resulting in a disaster: False alerts as a result of the IoT platform and its sensors not being maintained. The initial sensor's position information was incorrect (probably due to the wrong configuration of the system).This is the same or even more important if the IoT platform is programmed to automatically validate the alarm. The consequences of failing this use case because the data became unusable at some point in time may be very important: forest, remote facility, building destroyed with the potential loss of human/animal lives.

A.3.4 Emergency services teams accessing pre-deployed IoT devices

Ref: Use Case B.2 in ETSI STF 601

Description: This use case refers to an emergency in which emergency services (e.g. firemen) require data from IoT devices placed in a smart building, such as smoke/heat detectors, security cameras, and devices in elevator cabins. These data might assist them in better concentrating their operations in the event of a fire, for example, by locating personnel who are still on the scene, areas where the fire is burning, and so on.

Unless there are control rooms (with employees managing the technology), emergency services typically do not have instant access to other IoT devices pre-deployed in a building, such as security cameras. In this situation, the IoT devices are owned by the building administration and management, and only the data they generate is shared with emergency services on demand. Normally, emergency services do not have access to these IoT devices.

AI/ML Impact: An emergency service decision-maker requests access to a building's safety system from the authenticating entity. Access is granted by the authenticating entity. The emergency service decision-maker, which is based on machine learning, may acquire IoT device data from the building's safety system via the IoT service platform.

Analysis: Sensor data and semantics are given by the building IoT platform Data for security authentication that may restrict emergency services from accessing and using building data. If the emergency response team's devices are not permitted access by the authenticating entity, the completion may fail. The equipment used by the emergency services team is incompatible with the data given by the building's safety system. The gadgets used by the emergency services team misread the data supplied by the building's safety system, reducing the overall efficiency of the rescue operation.

IoT data can be used if the individuals who need to access them have complete access to them. Lack of compatibility at the data level across systems (e.g. undefined semantics) may prevent this data from being used. The implications of failing this use case because the data are worthless, as explained above, may be severe: the building may be fully demolished, perhaps resulting in the loss of human life.

A.3.5 Monitoring of industrial manufacturing equipment

Ref: Use Case C.1 in ETSI STF 601

Description: A general purpose of Artificial intelligence in industry and manufacturing is the development or optimization of machinery used for production, manufacturing, and delivery. Additionally, the identification of faults in the manufacturing of a product is a common use of artificial intelligence in industry and manufacturing. A process that manufactures canned food, for example, may identify that a label was not properly placed on the can or that the can was damaged. This can happen at several phases, including "can" manufacture, labelling, packaging in cartons for shipping, loading cartons onto vehicles, stocking products in retail outlets, and delivery from delivery services.

AI/ML Impact: Machinery and equipment function in accordance with their intended purpose. The sensors monitor several aspects of the machinery or equipment's operation. The AI/ML system analyses sensor data to detect and recognize when the equipment is functioning outside of its usual parameters. When the measured parameters of the equipment exceed the "normal" range, a signal or indicator is created for system operators to act on to return the equipment to normal operating range.

Analysis: Data that may compromise data usability: data coming from the sensors if data are not generated normally (at the expected time intervals) or corrupted (fail to detect abnormal conditions), there is a risk that the machinery comprising the industrial production line could be damaged or the product that would be generated would be defective.

A.3.6 Monitoring of industrial manufacturing products

Ref: Use Case C.2 in ETSI STF 601

Description: The identification of faults in the manufacturing of a product is a typical use of artificial intelligence in industry and manufacturing. A process that manufactures canned food, for example, may identify that a label was not correctly placed on the can or that the can was damaged. This can happen at several phases, including "can" production, labelling, packing in cartons for shipment, loading cartons into vehicles, stocking items in retail outlets, and delivery from delivery services.

AI/ML Impact: Within a supply chain process, products are made or transferred. The sensors collect data on several features of the items. The AI/ML system examines sensor data to detect product features that are outside of typical parameters. When a product's attributes deviate from the "normal" range, a signal or indication is created for system operators to use to take necessary action.

Analysis: If sensor data is manipulated, it is possible that buyers would get faulty items, for example, a bag of chips is not fully closed.

A.3.7 Intelligent Software Rollouts

Ref: Use Case D.1 in ETSI STF 601

Description: Although an intelligent software rollout is automated, incorrect data might have effects on the system, and this use case examines such use and the effect. During the life of a key resource, such as a router, its firmware will be updated, not only to support new services or functions but also to correct existing flaws. A firmware rollout can take many months to design and implement in some circumstances. Operators can create distinct policies for different sorts of rollouts and resources by using the Experiential Networked Intelligence (ENI) System. One example is the creation of a hierarchy of parameters for phasing out the rollout, such as client class, geographical location, or time of day. Furthermore, taking dynamic onboarding and DevOps environments into account, the ENI System may be used to establish many sorts of regulations, such as the results of the testing should show a relationship between network function performance (throughput, jitter, and latency) and resource allocation (CPU, RAM, and I/O). Scheduled updates for the enterprise customers will be outside of office hours.

AI/ML Impact: The process starts from the network operator development environment, a new software version is instantiated. The program is subjected to a series of tests in the new environment that is set by pre-defined policies in the ENI System, the results of which will be used to produce a software profile. During the testing, the ENI System begins analysing the behaviour via the machine learning strategies of all the collected data from the devices, such as the software version on the PCs, and compares it to the profile of prior software versions. Because the test findings are consistent with earlier versions, the ENI System is ready to initiate the transition of the new version from development to production. The ENI System initiates the transition of the new version from development to production. Upon the completion of the procedure, the ENI System may signal relevant software components, such as OSS/BSS, that the software deployment was successful.

Analysis: Data that may compromise data usability: report of the updating process. User intervention may interrupt the automatic software update. Monitoring of data generated by sensors may produce inconsistency of results which may affect the retries of a failed upgrade. Also forces user intervention whilst update is taking place may give false feedback that installation has been completed which may not be the case. The consequence of the update not happening means that the software was not fully deployed properly, and this is not detected, this suggests inconsistencies in all systems and could mean unfruitful use of resources indirectly financial implications to the organization.

A.3.8 Policy-based network slicing for IoT security

Ref: Use Case D.2 in ETSI STF 601

Description: Smart cities are planned to be created in the near future employing a plethora of IoT devices, with a considerable proportion of them connected through 5G. These gadgets will be crucial in the implementation of numerous services (e.g. civil protection or other services provided by the municipality, where each service will have its target use and different device requirements). To facilitate this large device deployment, network slices will enable their aggregation either by functionality (e.g. security or city operations management support) or by other sorts of lower-level needs, such as low latency and high bandwidth. In this context, the prevention of Distributed Denial Of Service (DDOS) assaults is critical, as these devices are often intended to enable applications/services of social interest.

AI/ML Impact: Machine learning is used in the ENI System to recognize certain traffic patterns that indicate DDOS or other types of assaults. This is due to the rising sophistication of such assaults, which makes it more difficult to employ simpler algorithms (for example, pattern recognition) that focus on a set of predetermined information. A DDoS assault manifests itself as unusually sluggish network performance and/or the inability to access a certain group of online sites. When this occurs, the ENI System will be able to recognize and learn from the event utilizing AI approaches. If the new traffic pattern is detected as an attack based on previous experience, the ENI System will be able to initiate necessary reactions from the relevant management components.

Analysis: A potential error occurs when ENI data collected from websites or devices, for example, is corrupted in some way, resulting in the ENI system being unable to detect that an abnormal event has occurred (e.g. a website is no longer accessible, or a device's traffic patterns do not correspond to its expected behaviour) and carrying out necessary analysis to determine whether the system is under attack or not. Data generated from devices should be made available to the Network Administrator before analysis, so that, if necessary, the Network Administrator can intervene based on their analysis.

A.3.9 Fertilization/Irrigation/Pest management service

Ref: Use Case E.1 in ETSI STF 601

Description: This use case is about a technical solution that provides farmers with next-generation advice by providing a variety of innovative smart agricultural services. Such a technology solution is given as a low-cost service with no technological investment required by farmers, making it accessible to even small farmers. The system is made up of an IoT device infrastructure, a collection of cloud computing services, and a set of smart agricultural advice services. IoT devices function as telemetric autonomous terminals, collecting data from sensors installed in the field and recording atmospheric (air temperature, relative humidity, wind direction and velocity, rain, leaf wetness) and soil parameters (temperature, moisture) as well as controlling irrigation systems.

The cloud service combines IoT data with data from other sources (such as weather forecast services and satellite photos) and translates it into facts using advanced data analysis techniques. The inbuilt Decision Support System (DSS) converts facts into preliminary guidance, which farmers can access via applications and certified agricultural consultants hired by third parties. The advisers are examining and evaluating the material provided to provide final advice and help to the farmers. The advising and support services are given by a combination of software and certified advisors, which includes Fertilization and Irrigation guidance as well as Pest Management/Hazard warnings.

AI/ML Impact: The sensors collect environmental data and transfer it to the cloud service. The cloud service takes all of the sensor data and saves it to the internal knowledge graph. The decision support system which is based on machine learning evaluates incoming data in real-time and makes a decision that is recommended for the farmers. Farmers checks if the system has properly established the fertilizer and irrigation systems, or some pest management issues have been found.

Analysis: Data that may compromise data usability includes data provided by the satellite, data provided by weather stations and, device and IoT data located in the field. The decision support system is driven by the three types of data mentioned above. If one of the data sources fails to read the values or to provide them promptly, they would become not usable by the system with the risk of having ineffective management of the farming system.

In particular, the IoT devices located in the field are particularly exposed to adverse events (e.g. meteorological ones). This aspect can be mitigated by implementing a redundant network of sensors that can be used both as backup and as confirmation of the read values. This action would increase the overall data usability.

In particular, the IoT devices located in the field are particularly exposed to adverse events (e.g. meteorological ones). This aspect can be mitigated by implementing a redundant network of sensors that can be used both as backup and as confirmation of the read values. This action would increase the overall data usability.

A.3.10 Vehicle Diagnostic & Maintenance Report

Ref: Use Case F.1 in ETSI STF 601

Description: The Vehicle Service Centre needs to notify the car owner of the vehicle's status and to remind them to maintain the vehicle in a timely manner to avoid disaster. As a result, the Vehicle Service Centre will acquire and analyse data from the vehicle on a regular basis. Based on the analysis results, it will alert the vehicle owner, displaying what's wrong with the vehicle in simple language and visuals, as well as some maintenance recommendations.

The ASSIST-IoT project outlined additional detailed examples of this use case in deliverable D3.2 [i.11]: vehicle diagnostics and non-conformance cause identification, vehicle exterior condition inspection and documentation, and verification of fleet emissions in service.

AI/ML Impact: The car collects diagnostic data from internal sensors and delivers it to the service centre. The diagnostics data includes information from the Engine and Transmission Systems, the Stability Control System, the Air Bag System, the Emission System, the Antilock Brake System, and other systems. The diagnostic data is sent to the "Vehicle Detection M2M Application" by the vehicle service centre. This machine learning application collects and analyses the diagnostic data. The "Vehicle Detection M2M Application" determines that the brake pads require replacement. It searches the maintenance services supplied by the "M2M Application" and obtains the contact information for the company that may supply the components. Finally, the "Vehicle Detection M2M Application" transmits the Diagnostic & Maintenance Report, together with the proposed component providers, to the vehicle owner by email or alert message shown in the vehicle terminal. Based on the Diagnostic & Maintenance Report, the car owner will determine whether or not to maintain his or her vehicle.

Analysis: IoT Data that may compromise data usability includes M2M device data provided to Vehicle Service Centre; diagnostics data provided to Vehicle Detection M2M Application. For example, sensor measurements describing the vehicles' operation may be taken at very high sampling frequencies. Results of the maintenance evaluation which has to be statistically significant, e.g. at a 95 % confidence level. Diagnostic & Maintenance Report from Vehicle Resolution M2M application provided to Vehicle Owner: completeness of the vehicle-scanning report, user-friendliness of the defect-identification process, unambiguous identification of the documented vehicle. The Applications behind the Vehicle Service Centre need to be able to determine which parts need to be replaced. The data usability and easy understanding by humans of M2M device data are necessary (translation to natural language when relevant). In the case of AI, data presentation and integrity are important to ensure a valid AI decision.

The vehicle owner (and the service technician) needs to understand how urgent the maintenance is, what needs to be replaced and which part needs to be changed. For example, when receiving information about vehicle lights, it should know what action needs to be performed and on which side of the vehicle it should be done. If the data is misleading (e.g. only mentioning "defective front light" or indicating a hexadecimal code that leads to another part because the system was improperly set up), the technician may not be able to successfully perform the relevant maintenance. The mechanics (technician) needs also to be able to understand how to act on the IoT platform to check the validity of data delivered by sensors (e.g. to identify faulty sensors) and to reset the condition that led to maintenance when it has been fixed. If the report is ambiguous, the mechanics may also order the wrong part. The consequences of failing this use case because the data were unusable may be very important: safety is not fulfilled in the vehicle, and it may lead to an accident with the potential loss of human lives.

A.3.11 Smart Parking

Ref: Use Case F.2 in ETSI STF 601

Description: Smart parking assists in addressing one of the most difficult aspects of driving in cities: locating empty parking places and managing unlawful parking. Parking spots are widely dispersed and may be held by separate providers (e.g. mall parking provider and street parking provider), making it difficult to acquire information at a single location/time. Drivers may use smart parking services to search available parking places, pay parking rates, and even make reservations. Because regular parking services follow a first-come, first-served norm, making parking reservations would be available even dynamically for limited persons such as VIPs or the disabled.

In this use case, a user arrives at a mall but finds that the parking lot is filled. The mall parking provider converts the provisional reservation to a street parking reservation. The automobile driver is provided information on where to park on the street as well as a discount voucher to compensate for the higher parking charge on the street. This site is near a disabled-accessible location. To prohibit parking in designated places, law enforcement officials are also involved.

AI/ML Impact: Because the user already specified the mall as the destination, the navigator automatically communicates the user's location to the mall parking provider when the user is near the mall. The mall parking supplier advises the navigator that there are now no available parking spaces. Based on the car's position, which is near the mall, the mall parking provider uses the M2M service platform enabled by machine learning to enquire about the availability of alternative parking spots. Because there are available parking places on the street, the mall parking provider suggests that location. To begin parking, the user approaches the smartphone from a parking meter. The street parking provider, as well as the mall parking provider, are then alerted. As recompense, the mall parking provider provides a discount coupon for parking outside. To finish parking, the user taps the smartphone on the meter. The parking provider on the street charges a price. Through the M2M service platform, the bill with the discount coupon is delivered to the billing provider.

Analysis: Parking lot status, location of designated parking space in the street, parking area fee and discount coupon, parking area status (e.g. disabled-only) in the street and the mall, car plate information for example, if a vehicle driver is given confusing or incorrect information regarding available parking areas, he may park in a dynamically assigned disabled-only spot or an already reserved place and be punished. This might be caused by incorrect service configuration if the parking space numbers in the IoT system do not correspond to the actual parking place locations.

The use case also includes a sensor that notifies unlawful parking to a police station. The data given should be accurate and delivered to the cops in a human-readable manner. The implications of failing this use case due to useless or confusing data are moderate: a handicapped person may be unable to park and proceed to the mall, and the owner of another car may be overcharged. This is more about a poor degree of service quality.

A.3.12 Energy optimization using AI

Ref: Use Case G.1 in ETSI STF 601

Description: This use case elaborates on how NFV and AI can be combined to optimize usage of the energy in networks and to also show the consequence if there is no good data usability. The other equipment including switches, routers, storage equipment, and air conditioners takes the other 30 % of the total power consumption. The servers are deployed and running to meet the requirement of peak-hour service, which means the servers are normally at a high power-up state at full time even in non-peak hours. It is possible to move the services to some of the servers and turn the other servers to idle or underclocking state in non-peak hours, with the aim of optimizing the power usage at the DC.

AI/ML Impact: The ENI system uses an AI algorithm to build the relations between the network service and its required resources, and the relations between the power consumption and the environment settings including e.g. the location of the running servers, the setting of the cooling system, etc. The ENI system collects and stores information of the virtual networks, including CPU usage, storage usage, and network usage for each VNF, etc. as well as the power consumption information and environmental information.

Analysis: The data within the ENI system is received from software sensors, however, if the data is compromised because of a corrupted sensor interface, the data is no longer usable by the machine, or it could also mean that the machining process the wrong data to give false instruction indicating the power does not to be switched to save power. The data are from software sensors, but they still apply here when it comes to data usability.

A.3.13 Smart safety of workers at building construction site

Ref: Use Case H.4 in ETSI STF 601

Description: The main objective of this application is the prevention and near real-time detection of common OSH hazards such as stress, fatigue, overexposure to heat and UV radiation, slips, trips, falls from height, suspension trauma, immobility due to unconsciousness, collision (forceful impact) with heavy equipment, entrapment (unable to evacuate the worksite during an emergency) and improper use of PPE. The OSH manager combines, only a relevant subset of real-time data with information that is manually provided from the entire workforce via existing management and collaboration platforms to assess and report the overall risk status for the construction site. The physiological parameters of the construction workers are being monitored in real-time using wearable sensors to ensure that their health and safety are always protected while at the construction site. The construction workers' location within the construction site is monitored so that the first responder can be sent in case of an emergency.

When a construction worker requests navigation instructions from his current location to a worksite, he should follow approved walking paths through areas the worker is authorized to access. Within any building construction site, many people with various levels of training and experience, are occupied by several subcontracted companies, interact with each other, operate equipment, or interface with heavy machinery. Construction workers and the project's OSH manager are provided with relevant information about incidents and potential hazards.

AI/ML Impact: The IoT system monitors the weather conditions at the construction site, the exposure of the construction worker to UV radiation, the physiological parameters of the construction worker, and the motion pattern of the construction worker's body. The IoT system analyses all the collected data from all the construction workers who are present at the construction site and identifies no increased risk of exposure to OSH-related hazards for the construction worker.

The IoT system is aware of the construction worker's identity and pairs them with their smart PPE. The IoT system tracks the location of the construction worker. The IoT system has not detected any increased risk to the construction worker's health and safety, but they notify the OSH manager by raising an alarm through ASSIST-IoT.

Analysis: Lack of usability of the data or compromised data in this use case may result in limited safety for the site workers, which may have strong consequences in case of false negatives (unhealthy conditions are not detected or invalid navigation instructions that drive the worker to a hazardous area). Location and proximity data of workers on site, physiological parameter measurements, weather conditions measurements, personal identification information, training, and medical records, building information, users' thermal comfort preferences, alerts, and notifications. The BIM should be set up without any ambiguity or invalid information, as this may lead to invalid navigation and put the workers at risk. The user interfaces in the construction worker & device should be easily understandable, whatever the language and reading level of the worker. Data flow from all devices should be secured and respect privacy, as well as guaranteed data integrity.

A.3.14 Machine socialization

Ref: Use Case H.5 in ETSI STF 601

Description: If robot A has cleaned a room, it may inform the other robot that this room has been cleaned, so robot B can move to another room for a clean job. As in the hotel scenario, the robot owners may not tell the robots explicitly that there exists another robot with the same task. If the hotel has only one robot, it will clean rooms one by one.
A robot is designed to clean rooms in hotels. For example, after a machine scans a room, it will find out the clean status of that room (clean or dirty), when a robot is cleaning a room after it is cleaned, it will change the status of that room, the information will affect other robots' behaviour because for any other robots it is unnecessary to go to a room that is being cleaned or has been cleaned by another robot. Because the platform may help robots to discover each other, and the platform may initialize a powerful commander to optimize the job with multiple robots. Secondly, a robot should realize what kind of information will affect other robots' behaviour, and it should transmit messages to share this information with other co-operators. Thirdly, a robot should know the message interface of other robots.

AI/ML Impact: Robot A shares information with robot B and Robot B shares information with Robot A. Robots A and B discover each other (the discovery is performed by themselves or aided by the cloud robot service platform) The cloud robot service platform helps to optimize the task process and helps the robots to cooperate.

Analysis: Capabilities of a Robot: when a robot is placed into service, if it does not describe its capabilities correctly then the overall service requests cannot be handled optimally by the cloud robot service platform. cloud robot service platform commands: when a robot is commanded to perform a service, if the command is not recognized then the robot may not be efficiently utilized.

A.3.15 Retail inventory management

Ref: Use Case I.1 in ETSI STF 601

Description: The AI/ML algorithm processes data from the sensors that detect when items are removed from their retail position. When a certain number of products are removed a signal or indicator is generated for the system operators to act upon to take appropriate action.

AI/ML Impact: The presence of the following actors/entities as well as their associated roles is envisaged in the current use case: [Device] sensors that can identify the presence and absence of a product in a retail environment. [Human] A monitor, human or automatic, that can react based on the detected anomaly of a product inventory that needs replenishment. Data user/consumer: AI/ML Processing module, Monitor.

Analysis: Incorrect classifications from the AI/ML algorithm can lead to false "tasks or jobs" to replenish stock or failure to create a "replenish task/job" depending on the setting this could lead to expensive delays, in the case that a truck delivery needs to be rescheduled, or not enough trucks allocated to the delivery task.

A.3.16 Vending Machines

Ref: Use Case I.2 in ETSI STF 601

Description: As part of the use case, the provider can limit access to the availability of service for vending machines based on their geographic location. How many products are sold in specified areas during the specified time? The vending machine providers do NOT want the vending machine users to move the machine from the specified area to other locations (potentially for better sales), hence, the providers can control the geographic distribution of their vending machines and make decisions based on data statistics and analysis.

AI/ML Impact: After a vending machine successfully registers, it reports data information (for example, the product selling information and the stock information) periodically or for each product sale to the vending machine application platform through the M2M service platform. The vending machine application platform receives the data information report, records the information, and performs data analysis. If the current geographic location of the vending machine is in the permitted area, it allows for the data report. If the current geographical location of the vending machine is in the permitted area, it allows the vending machine to register.

Analysis: Data that may compromise data usability: Data from the M2M application, that will include the sensor data about the wrong data information. For example, some users may physically shake the machine and some products are available to users in which case it is not recorded as sales, but it is not available for use.

A.3.17 Crowd Safety and Security

Ref: Use Case J.1 in ETSI STF 601

Description: This use case is about a cloud-based IoT platform supporting a series of applications that can be used to monitor, record, and analyse the environment and consequently predict or identify situations that need attention in a large-scale event such as a concert. The MONICA services include crowd and capacity monitoring, detection of security, health, and safety incidents as well as the location of and communication between staff, visitors, and control centre. During the event, the security personnel can monitor the situation using a web-based interface - the MONICA COP which provides an operational picture of the environment in real-time, and which displays notifications in case of any unusual activities.

AI/ML Impact: The use case relies on sensors generated by wristbands, CCTV cameras, mobile phones, trackers, and glasses, reporting data to an IoT cloud-based system, which handles incidents, supported by applications that help ensure efficient communication and timely response.

Analysis: The applications are primarily based on data from CCTV cameras and crowd wristbands and these data can be compromised if, for example, the attendants of the event do not wear their wrist bands or readings are inaccurate. The consequence, if the data is not correct, means there could be many crowds out of control, and this could become unsafe in the context of large events like football games or concerts.

A.3.18 Predictive maintenance and fault tolerance

Ref: Use Case K.1 in ETSI STF 601

Description: This case is about how the maintenance companies and technicians can use the available information to set a predictive maintenance program for the lift, and how they can use the remote connection with the lift to fix the faults or problems. Furthermore, there are some faults extremely hard to discover and that require a long time to be fixed, so the capability for the maintenance technician to have direct and real-time support by the supplier technician could drastically reduce the out-of-service necessary to fix the fault. Predictive maintenance is the trend in the lift industry even if it has been applied in several industrial sectors for ages. The scope of predictive maintenance is to anticipate the event of a fault, evaluating the fault rate of the single components based on the number of runs of the lift.

AI/ML Impact: The remote central control unit AI component detects a data pattern that, with a probability higher than a specified threshold, can lead to a fault or an undesired event. The smart lift IoT ecosystem provides data to the remote central control unit installed within the Maintenance Company premises. The smart lift installation restarted to work and the alert within the remote central control unit disappeared.

Analysis: In this case for the maintenance technician, it is extremely hard to discover the fault the best solution is that the technician of the control cabinet supplier connects the lift from the remote position and analyses the faults by the history of the faults recorded and the capability of analysing the single input and output of the mainboard, he can very quickly identify which landing door causes the fault and understand why the fault appears. With predictive maintenance, the system sends a message to the maintenance company that the lifetime of a component (for example wheel of the doors, pushbutton, etc.) has expired the maintenance company can send a technician to substitute the component with a new one, so the time of substitution could be noticeably short and the possible out of service of the lift avoided.

A.3.19 Low-connection environments

Ref: Use Case K.2 in ETSI STF 601

Description: This use case is inspired by the Lift Predictive Maintenance one with the difference that often, data produced by lift installations can only be saved, retrieved, and exploited locally due to, for example, a network connection that is not always one or given the high bandwidth cost. From the data usability perspective, in this use case, the critical aspect is not related to the grant effective reading from IoT devices, but also to having an effective and efficient data transfer from each lift location and the central data collector. This aspect opens to a more complex scenario where the entire end-to-end pipeline is decentralized and, by assuming to have a central data collector processing all produced data, reasoning and learning operation can be performed in a federated way.

AI/ML Impact: The cloud data manager collects data from different installations deployed into low-connectivity environments and aggregates the data within a central data model. The decision support system connected to the cloud data manager detects a data pattern that, with a probability higher than a specified threshold, can lead to a fault or an undesired event. The lift IoT ecosystem provides data to the cloud data manager installed within the lift company premises. The lift IoT ecosystem provides data to the local data manager installed within the local environment. The lift company is alerted, and, in turn, it alerts the maintenance supplier about the undesired event together with the related explanation.

Analysis: Hence, any service and activities relying on the quality of the data provided by the IoT ecosystem to the main server can compromise the processing activities of the AI-based predictive system by making it ineffective for addressing the predictive maintenance task.

A.3.20 Smart Lighting

Ref: Use Case L.1 in ETSI STF 601

Description: In huge urban areas, the IoT stage carrying out this utilization case needs to continuously gather the information from many various sensors situated around the city, total them, and feed them to the AI cycle that takes the lighting choice.

AI/ML Impact: In large cities, the IoT platform implementing this use case will collect in real-time the data from thousands of different sensors located around the city, aggregate them, and feed them to the AI process that takes the lighting decision. In addition, this use case may help improve the inhabitant's safety (sensitive areas get higher illumination), improve the city maintenance operations, and reduce global energy consumption. Other use cases such as smart waste or environmental monitoring would lead to similar discussion in terms of data usability of the devices and the data they provide.

Analysis: Due to the large amount of data to be collected, intermediate platforms aggregating the data from a selected location or selected type of sensors may filter unusual events and facilitate the decision-making process, by reducing its complexity. Interoperability between the devices measuring the sensor data and the platforms aggregating and exploiting the data has a strong impact. They should receive only useful, complete (e.g. no missing data in the flow) and non-redundant data (except where needed). As in all AI use cases, the data presentation, format, and meaning need to be clearly defined at all levels of the processing chain. Lack of usability of the data or compromised data in this use case may have different consequences.

History

Document history		
V5.0.1	August 2023	Publication