**3GPP TSG-SA5 Meeting #162 *S5-253712Rev1* Gothenburg, Sweeden, 25 - 29 August 2025**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.3* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  |  | **CR** |  | **rev** | **-** | **Current version:** |  |  |
|  | | | | | | | | |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network | **X** | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | Input to DraftCR Rel-19 TS 28.105 AIML management - phase 2 rapporteur's clean up | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | NEC | | | | | | | | | |
| ***Source to TSG:*** | S5 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** |  | | | | |  | ***Date:*** | | | 2025-08-15 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | **F** |  | | | | | ***Release:*** | | |  |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) … Rel-17 (Release 17) Rel-18 (Release 18) Rel-19 (Release 19)  Rel-20 (Release 20)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | The work item on AIML management – Ph2 is planned to be conclused in the next meeting. Current content document in DraftCR need to be checked and corrected | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | This input to the Draft CR addresses a wide range of language issues by introducing corrections and improving clarity. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | The specification may remain misleading or unclear, potentially causing confusion and incorrect implementations. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 3.1,  4.X2, 4a.0, 4a.1,  6.2b,6.2b.1, 6.2b.2, 6.5  7.2a, 7.5.1 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **X** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **X** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **X** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

***1st change***

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 28.104: "Management and orchestration; Management Data Analytics".

[3] 3GPP TS 23.288: "Architecture enhancements for 5G System (5GS) to support network data analytics services".

[4] 3GPP TS 28.552: "Management and orchestration; 5G performance measurements".

[5] 3GPP TS 32.425: "Telecommunication management; Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN)".

[6] 3GPP TS 28.554: "Management and orchestration; 5G end to end Key Performance Indicators (KPI)".

[7] 3GPP TS 32.422: "Telecommunication management; Subscriber and equipment trace; Trace control and configuration management".

[8] Void

[9] 3GPP TS 28.405: "Telecommunication management; Quality of Experience (QoE) measurement collection; Control and configuration".

[10] Void

[11] 3GPP TS 28.532: "Management and orchestration; Generic management services".

[12] 3GPP TS 28.622: "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)".

[13] 3GPP TS 32.156: "Telecommunication management; Fixed Mobile Convergence (FMC) Model repertoire".

[14] 3GPP TS 32.160: "Management and orchestration; Management service template".

[15] 3GPP TS 28.533: "Management and orchestration; Architecture framework".

[16] 3GPP TS 38.300: "NR; NR and NG-RAN Overall description; Stage-2".

[17] 3GPP TS 38.401: "NG-RAN; Architecture description".

[18] 3GPP TS 28.541: " Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3".

[19] 3GPP TS 28.623: "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Solution Set (SS) definitions".

[20] 3GPP TS 29.520: "5G System; Network Data Analytics Services; Stage 3".

[X1] 3GPP TS 28.319: “Management and orchestration; Access Control for Management services”.

[X2] 3GPP TS 28.111: "Management and orchestration; Fault Management (FM)".

|  |
| --- |
| **Next change** |

## 3.1 Terms

For the purposes of the present document, the terms given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

**ML model:** a manageable representation of an ML model algorithm.

NOTE 1: An ML model algorithm is a mathematical algorithm through which running a set of input data can generate a set of inference output.

NOTE 2: ML model algorithm is proprietary and not in scope for standardization and therefore not treated in this specification.

NOTE 3: ML model may include metadata. Metadata may include e.g. information related to the trained model, and applicable runtime context.

**ML model training:** a process performed by an ML training function to take training data, run it through an ML model algorithm, derive the associated loss and adjust the parameterization of that ML model iteratively based on the computed loss and generate the trained ML model.

**ML model initial training:** a process of training an initial version of an ML model.

**ML model re-training:** a process of training a previously trained version of an ML model and generate a new version.

NOTE 4: A new version of a trained ML model supports the same type of inference as the previous version of the ML model, i.e., the data type of inference input and data type of inference output remain unchanged between the two versions of the ML model, but parameter values might be different for the re-trained model.

**ML model pre-specialized training**: the process of training an ML model on a dataset not specific to any type of inference. NOTEx1: The pre-specialised trained model supports an inference scope that may be potentially adapted to support a list of inference types, such as MDA types in MDA, analytics types in NWDAF, type of AI/ML supported functions in NG-RAN, or vendor-specific extensions.

**ML model Fine-tuning**: the process of training a pre-specialised trained ML model to narrow its inference scope to a new single inference type, generating a new ML model.

NOTE x2: The inference scope refers to a list of inference types that the ML model may be potentially adapted to support.

NOTE x3: The type of inference represents the specific type of ML inference supported by the model, such as MDA types in MDA, Analytics types in NWDAF, type of AI/ML supported functions in NG-RAN, or vendor-specific extensions.

**Distributed training:** a process of distributing the training workload across multiple ML training functions**.**

**Federated learning:**  a distributed machine learning approach where the ML model is trained collaboratively by multiple ML training functions. This includes multiple FL clients, which perform training on local data, and one FL server, which aggregates model outcomes from the clients iteratively without exchanging data samples.

**Horizontal federated learning:** a federated learning technique without exchanging/sharing local data set, wherein the local data set in different HFL clients for local model training have the same feature space for different samples.

**FL Client:** a training function that trains an ML model on local data and shares only the model updates with the FL server, preserving data privacy.

**FL Server:** a function that aggregates the ML model updates from FL Clients to produce a global ML model.

**Reinforcement learning**: a machine learning approach in which an RL agent interacts with an RL environment by observing states, taking actions and receiving rewards as feedback. The RL agent learns a decision making policy by maximizing rewards over time through trial and error.

**ML model joint training:** a process of training a group of ML models.

**ML training function**: a logical function with ML model training capabilities.

**ML knowledge**: the implicit information representing experience gained by the training of an ML model

NOTE x4: Examples of experience include statistics (e.g. a distribution) or summaries (e.g. tables) indicating the ML model’s recommended output for a given set of input data.

**ML model testing:** a process of evaluating the performance of an ML model using testing data different from data used for model training and validation.

**ML model joint testing**: a process of evaluating the performance of a group of ML models using testing data different from data used for model training and validation.

**ML testing function**: a logical function with ML model testing capabilities.

**AI/ML inference**: a process of running a set of input data through a trained ML model to produce set of output data, such as predictions.

NOTE 5: The inference represents the process to realize the AI capabilities by utilizing a trained ML model and other AI enablers if needed, hence the AI/ML prefix is used when referring to inference as compared to training and testing.

**AI/ML inference function**: a logical function that employs trained ML model(s) to conduct inference.

**AI/ML inference emulation**: running the inference process to evaluate the performance of an ML model in an emulation environment before deploying it into the target environment.

**ML model deployment:** a process of making a trained ML model available for use in the target environment.

**ML model loading**: a process of making a trained ML model available to an inference function.

**AI/ML activation**: a process of enabling the inference capability of an AI/ML inference function.

**AI/ML deactivation**: a process of disabling the inference capability of an AI/ML inference function.

|  |
| --- |
| **Next change** |

## 3.2 Symbols

Void.

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and TS 28.533 [15]. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1] and TS 28.533 [15].

AI Artificial Intelligence

ML Machine Learning

FL Federated Learning

HFL Horizontal Federated Learning

VFL Vertical Federated Learning

***Next change***

4 Concepts and overview

4.1 Overview

The AI/ML techniques and relevant applications are being increasingly adopted by the wider industries and proved to be successful. These are now being applied to telecommunication industry including mobile networks.

Although AI/ML techniques in general are quite mature nowadays, some of the relevant aspects of the technology are still evolving while new complementary techniques are frequently emerging.

The AI/ML techniques can be generally characterized from different perspectives including the followings:

- **Learning methods**

The learning methods include supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning. Each learning method fits one or more specific category of inference (e.g. prediction), and requires specific type of training data. A brief comparison of these learning methods is provided in table 4.1-1.

**Table 4.1-1: Comparison of Learning methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Supervised learning** | **Semi-supervised learning** | **Unsupervised learning** | **Reinforcement learning** |
| **Category of inference** | Regression (numeric), classification | Regression (numeric), classification | Association, Clustering | Reward-based behaviour |
| **Type of training data** | Labelled data (Note) | Labelled data (Note), and unlabelled data | Unlabelled data | Not pre-defined |
| NOTE: The labelled data refers to a set of training and testing data that have been assigned with one or more labels in order to add context and meaning. | | | | |

**- Learning complexity:**

- As per the learning complexity, there are Machine Learning (i.e. basic learning) and Deep Learning.

**- Learning architecture**

- Based on the topology and location where the learning tasks take place, the AI/ML can be categorized to centralized learning, distributed learning and federated learning.

**- Learning continuity**

- From learning continuity perspective, the AI/ML can be offline learning or continual learning.

Artificial Intelligence/Machine Learning (AI/ML) capabilities are used in various domains in 5GS, including management and orchestration (e.g. MDA, see 3GPP TS 28.104 [2]) and 5G networks (e.g. NWDAF, see 3GPP TS 23.288 [3]).

The AI/ML inference function in the 5GS uses the ML model for inference.

Each AI/ML technique, depending on the adopted specific characteristics as mentioned above, may be suitable for supporting certain type/category of use case(s) in 5GS.

To enable and facilitate the AI/ML capabilities with the suitable AI/ML techniques in 5GS, the ML model and AI/ML inference function need to be managed.

The present document specifies the generic AI/ML management related capabilities and services without specifically taking any of the above-mentioned learning methods into consideration. The AI/ML management capabilities which include the followings:

- ML model training.

- ML model testing.

- AI/ML inference emulation.

- ML model deployment.

- AI/ML inference.

4.X1 Management of AI/ML capabilities for RAN

For the ML model training and AI/ML inference functions are both located in gNB, and the ML training function can be located in the management system and AI/ML inference function is located in the gNB. The NG-RAN AI/ML based feature defined in TS 38.300 [16] can be supported.

4.X2 Management of AI/ML capabilities for 5GC

For the ML model training and AI/ML inference functions are both located in 5GC. The NWDAF feature defined in TS 23.288 [3] can be supported.

4.X3 Management of AI/ML capabilities for MDA

For MDA, the ML training function can be located inside or outside the MDAF. The AI/ML inference function is in the MDAF. The MDA capabilities defined in TS 28.104 [2] can be supported.

|  |
| --- |
| **Next Change** |

4a AI/ML management functionality and service framework

4a.0 ML model lifecycle

AI/ML techniques are widely used in 5GS (including 5GC, NG-RAN, and management system), the generic AI/ML operational workflow shown in Figure 4a.0-1, highlights the main steps of an ML model lifecycle.

****

**Figure 4a.0-1: ML model lifecycle**

The ML model lifecycle includes training, testing, emulation, deployment, and inference. These steps are briefly described below:

**- ML model training:** training, including initial training, re-training, pre-specialised training and fine-tuning. Training could be for a single ML model or a group of ML models. It may also include validation of the ML model(s) to evaluate the performance when the ML model(s) performs on the validation data. If the validation result does not meet the expectation (e.g., the variance is not acceptable), the MnS Producer may decide to re-train the ML model(s).

**- ML model testing:** testing of a validated ML model to evaluate the performance of the trained ML model when it performs on testing data. If the testing result meets the expectations, the ML model may proceed to the next step If the testing result does not meet the expectations, the ML model needs to be re-trained.

**- AI/ML inference emulation:** running an ML model for inference in an emulation environment. The purpose is to evaluate the inference performance of the ML model in the emulation environment prior to applying it to the target network or system. If the emulation result does not meet the expectation (e.g., inference performance does not meet the target, or the ML model negatively impacts the performance of other existing functionalities), the ML model needs to be re-trained.

NOTE: The AI/ML inference emulation is considered optional and can be skipped in the ML model lifecycle.

**- ML model deployment:** ML model deployment includes the ML model loading process (a.k.a. a sequence of atomic actions) to make a trained ML model available for use at the target AI/ML inference function.

ML model deployment may not be needed in some cases, for example when the training function and inference function are co-located.

**- AI/ML inference:** performing inference using trained ML model(s) by the AI/ML inference function. The AI/ML inference may also trigger model re-training or update based on e.g., performance monitoring and evaluation.

NOTE 1: Depending on system implementation and AI/ML functionality arrangements, both AI/ML inference emulation and ML deployment steps may be skipped.

NOTE 2: The sequence of ML model lifecycle in Figure 4a.0-1 is the typical process in an E2E perspective. The actual sequence will depend on certain training techniques as described in their use cases, e.g., ML reinforcement learning, Distributed training or Federated learning, etc.

**Next change**

## 4a.1 Functionality and service framework for ML model training

An ML training Function playing the role of ML training MnS producer, may consume various data for ML model training purpose.

As illustrated in Figure 4a.1-1 the ML model training capability is provided via ML training MnS in the context of SBMA to the authorized MnS consumer(s) by ML training MnS producer.



Figure 4a.1-1: Functional overview and service framework for ML model training

The internal business logic of ML model training leverages the current and historical relevant data, including those listed below to monitor the networks and/or services where relevant to the ML model, prepare the data, trigger and conduct the training:

- Performance Measurements (PM) as per 3GPP TS 28.552 [4], 3GPP TS 32.425 [5] and Key Performance Indicators (KPIs) as per 3GPP TS 28.554 [6].

- Trace/MDT/RLF/RCEF/RRC data, as per 3GPP TS 32.422 [7].

- QoE and service experience data as per 3GPP TS 28.405 [9].

- Analytics data offered by NWDAF as per 3GPP TS 23.288 [3].

- Alarm information and notifications as per 3GPP TS 28.111 [X2].

- CM information and notifications.

- MDA reports from MDA MnS producers as per 3GPP TS 28.104 [2].

- Management data from non-3GPP systems.

- Other data that can be used for training.

|  |
| --- |
| **Next change** |

4a.2 AI/ML functionalities management scenarios (relation with managed AI/ML features)

The ML training function and/or AI/ML inference function can be located in the RAN domain MnS consumer (e.g. cross-domain management system) or the domain-specific management system (i.e. a management function for RAN or CN), or Network Function.

For MDA, the ML training function can be located inside or outside the MDAF. The AI/ML inference function is in the MDAF.

For NWDAF, the ML training function can be located in the MTLF of the NWDAF or the management system, and the AI/ML inference function is in the AnLF.

For RAN, the ML training function and AI/ML inference function can both be located in the gNB, or the ML training function can be located in the management system and AI/ML inference function is located in the gNB.

For LMF-based AI/ML Positioning, the ML training function can be located in the LMF or CN-domain management function, and the AI/ML inference function can be located in the LMF.

Therefore, there might exist several location scenarios for ML training function and AI/ML inference function.

**Scenario 1:**

The ML training function and AI/ML inference function are both located in the 3GPP management system (e.g. RAN domain management function). For instance, for RAN domain-specific MDA, the ML training function and AI/ML inference functions for MDA can be located in the RAN domain-specific MDAF. As depicted in figure 4a.2-1.

****

**Figure 4a.2-1: Management for RAN domain specific MDAF**

Similarly, for CN domain-specific MDA the ML training function and AI/ML inference function can be located in CN domain-specific MDAF.

**Scenario 2:**

For RAN AI/ML capabilities the ML training function is located in the 3GPP RAN domain-specific management function while the AI/ML inference function is located in gNB. See figure 4a.2-2.

**A diagram of a computer

Description automatically generated**

**Figure 4a.2-2: Management where the ML model training is located in RAN domain management function and AI/ML inference is located in gNB**

**Scenario 3:**

The ML training function and AI/ML inference function are both located in the gNB. See figure 4a.2-3.

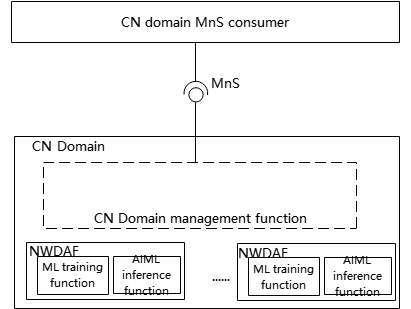
**A diagram of a function

Description automatically generated**

**Figure 4a.2-3: Management where the ML model training and AI/ML inference are both located in gNB**

**Scenario 4:**

For NWDAF, the ML training function and AI/ML inference function are both located in the NWDAF. See figure 4a.2-4.



**Figure 4a.2-4: Management where the ML model training and AI/ML inference are both located in CN**

**Next change**

6 AI/ML management use cases and requirements

## 6.2b ML model training

### 6.2b.1 Description

Before an ML model is deployed to conduct inference, the ML model algorithm associated with the ML model needs to be trained. The ML model training can be an initial training or the re-training of an already trained ML model.

The ML model is trained by the ML training MnS producer, and the training can be triggered by request(s) from one or more ML training MnS consumer(s), or initiated by the ML training MnS producer (e.g., as a result of model performance evaluation).

The procedures in [X1] for authentication and authorization are applicable for all ML model training use cases.

**Next change**

6.2b.2 Use cases

6.2b.2.X1 ML-Knowledge-based Transfer Learning

#### 6.2b.2.X1.1 Description

It is known that existing ML capability can be leveraged in producing new or improving other ML capability. Specifically, transfer learning allows knowledge contained in one or more ML models to be transferred to another ML model. Transfer learning relies on task and domain similarity to determine whether parts of a deployed ML model can be reused in another domain / task with some modifications.

In multi-vendor environments, aspects of transfer learning need to be supported in network management systems. However, as ML models are not expected to be multi-vendor objects, i.e. an ML model cannot be transferred directly from one function to another, the knowledge contained within the model, referred to as ML knowledge, should be transferred instead of the ML model itself.

ML knowledge represents the information representing experience gained by the training of an ML model (e.g. experience that indicates the recommended outputs for a given set of input data). This information can be of various forms such as statistics (e.g. a distribution) or summaries (e.g. tables).

For example, knowledge contained in an ML model deployed to perform mobility optimization during the day can be leveraged to produce a new ML model to perform mobility optimization at night. As illustrated in Figure 6.2b.2.X1.1-1, the network or its management system needs to have the required management services to support ML Knowledge-based Transfer Learning (MLKTL). ML Knowledge-based Transfer Learning refers to the capability of enabling and managing the transfer learning between any two ML models or training functions.



Figure 6.2b.2.X1.1-1: ML Knowledge-based Transfer Learning (MLKTL) flow between the source MLKTL  
(which is the ML training function with the pre-trained ML model), the peer MLKTL  
(which is the ML training function that shall train a new ML model) and the MLKTL MnS consumer  
(which may be the operator or another management function that triggers or controls MLKTL)

NOTE 1: The services between the source and peer for the transfer knowledge and triggering of learning is implementation specific and is not in scope of the current specification.

NOTE 2: Transfer learning (including knowledge-based transfer learning) can be sensitive from a data exposure and privacy aspect. Knowledge-based transfer learning will not be triggered for non-authorized MnS consumers, but the authorization of the MnS consumers is not in scope of the current document.

#### 6.2b.2.X1.2 Use cases

##### 6.2b.2.X1.2.1 Discovering sharable Knowledge

For transfer learning, it is expected that the source ML Knowledge-based Transfer Learning (MLKTL) Functionshares its knowledge with the target ML training function, either through a single instance of knowledge transfer or through an interactive transfer learning process. The ML knowledge here represents any experiences or information gathered by the ML model within the source MLKTL Functionthrough training, inference, updates, or testing. This knowledge can be in the form of data statistics, features of the underlying ML model, or the output of the ML model. The 3GPP management systems should provide mechanisms for an MnS consumer to discover this potentially shareable knowledge as well as the means for the MLKTL provider to share that knowledge with the MnS consumer.

##### 6.2b.2.X1.2.2 Knowledge sharing and transfer learning

Transfer learning may be triggered by an MnS consumer, either to fulfil its own learning needs or delegate the process to another ML training function. The model containing the knowledge may be an independent managed entity or it could be an attribute of a managed ML model or ML training function. In the latter case, MLKTL does not involve sharing the ML model itself or parts of it but would focus on enabling the sharing of the knowledge contained within the ML model or ML-enabled function.

The 3GPP management system should provide mechanisms and services needed to realize the ML Knowledge-based transfer learning process. Specifically, the 3GPP management system should enable an MnS consumer to request and receive sharable knowledge as well as means for the MnS producer of MLKTL to share the knowledge with the MnS consumer or a designated target ML training function. Similarly, the 3GPP management system should support MnS consumers in managing and controlling the MLKTL process, including handling requests associated with knowledge transfer learning between two ML models directly or via a shared knowledge repository.

The two use cases should address the four scenarios illustrated below.

It should be noted that, the use cases and requirements here focus on the required management capabilities the implementation of the knowledge transfer learning processes are implementation details that are out of the scope of the present document.

**Scenario 1:** Interactions for ML-Knowledge-based Transfer Learning (MLKTL) to support training at the ML knowledge Transfer MnS consumer as a peer - ML knowledge-based Transfer Learning MnS consumer obtains the ML knowledge which it then uses for training the new ML model based on knowledge received from the MLKTL source Function.

**Scenario 2:** Interactions for ML-Knowledge-based Transfer Learning (MLKTL) to support training at the peer ML Knowledge-based Transfer Learning Functiontriggered by the MLKTL Source - the ML Knowledge-based Transfer Learning MnS consumer acting as the MLKTL Source (the source of the ML knowledge) triggers the training at the ML knowledge-based Transfer Function by providing the ML knowledge to be used for the training, the ML Knowledge-based Transfer Learning MnS consumer then undertakes the training.

**Scenario 3**: Interactions for ML-Knowledge-based Transfer Learning (MLKTL) to support training at the Peer ML knowledge-based Transfer Learning Function who is different from the ML knowledge-based Transfer Learning MnS consumer - the ML knowledge-based Transfer Learning MnS consumer triggers training at the MLKTL peer Function. The MLKTL MnS consumer then obtains the ML knowledge from the MLKTL source Functionand then uses the knowledge for training the new ML model based on knowledge received from the MLKTL source Function.

**Scenario 4: I**nteractions for ML-Knowledge-based Transfer Learning (MLKTL) to support training at the Source ML knowledge Transfer Function- the ML knowledge-based Transfer Learning MnS consumer triggers training at the MLKTL source Function. The MLKTL MnS consumer then takes its available ML knowledge-based and uses the knowledge for training the new ML model based.

**Next change**

#### 6.2b.2.X2 ML model training for multiple contexts

Although the ML model may provide an AI/ML inference service for multiple scenarios, there are similarities in the contexts where ML models operate and perform AI/ML inferences. For e.g., two ML model instances for the same inference type in urban or rural areas would have significant overlap in their contexts with respect to their type of learning, performance characteristic, task solving type, clustering technique, training and inference time. The context similarity can be leveraged in forming a cluster of ML models, where ML model instances in the cluster are either trained from the same previously trained ML model or from an ML model previously trained for another similar context as the baseline. The training of an ML model for multiple contexts allows for efficiency by cluster training rather than individually training each one of them. ML training needs to support the capability to form a cluster of ML models as per clustering criteria and train them from the same baseline ML model or from an ML model previously trained for another similar context as the baseline. As input to the training, the clustering criteria needed to distinguish the ML model instances i.e. which ML models can form the cluster and trained together having similarities in context, learning paradigm, evaluation performance metrics, task type, training and inference time etc., may be provided by the MnS consumer.

In the case of degradation of ML models, updating of ML models is expected to be triggered. For ML models created by training in clusters, the retraining of a degraded ML model could be triggered to start from another member of the cluster, i.e. start from an ML model with another context to create a new ML model with the desired context.

|  |
| --- |
| **Next change** |

6.2b.2.X3 ML Pre-specialised training

ML model pre-specialised training refers to the process of training an ML model using a dataset that is not specific to any single type of inference. This means that the ML model training is not intended to support only one type of inference but rather leverages commonalities among multiple use cases. ML pre-specialised training can be appliedto AI/ML-based use cases specified in [2], and [3]. For example, an ML model could be pre- specialised trained using dataset from SLS analysis capability group covering type of inference including ServiceExperienceAnalysis, NetworkSliceThroughputAnalysis, NetworkSliceTrafficAnalysis, NetworkSliceLoadAnalysis and E2ElatencyAnalysis (see TS 28.104 []).

A pre-specialised trained model is not designed to conduct inference for a specific inference type, but this does not preclude the possibility for a pre-specialised model to conduct inference once it achieved performance requirement for a specific inference type.

A pre-specialised trained model can be fine-tuned to narrow down its inference scope, evolving into a new ML model with a single inference type.

The consumer may want to know which ML model could be pre-specialised trained, so the MnS producer shall identify whether an ML model could be pre-specialised trained.

6.2b.2.X4 ML Fine-tuning of pre-specialised trained model

Fine-tuning of a pre-specialised trained ML model narrows down the inference scope to support a single inference type. This means that a fine-tuned ML model can only support a single inference type for conducting inference.

Fine-tuning differs from re-training, where an ML model is re-trained on the same dataset and inference type for which it was previously trained.

The key distinction between re-training and fine-tuning is that re-training results in a new version of the same ML model for the same inference type, while fine-tuning produces a new ML model with an adapted inference scope or a single inference type.

Consumer may request fine-tuning of a pre-specialised trained ML model, and MLT MnS producer may evaluate whether the pre-specialised ML model can be fine-tuned according to the training requirements.

Authentication procedures need to be established with appropriate tunnelling and authorisation mechanisms prior to an ML fine-tuning training request.

**Next change**

#### 6.2b.2.X5 Management of distributed training

Distributed training is a model training approach that involves distributing the training workload across multiple training functions to accelerate the training process and/or reduce the required computational resources.

In 5GS, the ML training function may be located within the management system or in the NF (e.g. gNB or NWDAF),. Each training node has different computing resources and storage capacity based on physical infrastructure such as CPU/GPU/DPU, memory, storage, and network bandwidth. In order to obtain load balance between nodes and maximize the efficiency of resource utilization, splitting up the training may be necessary and involving multiple training functions according to the actual situation of nodes may be needed. Thus, aspects of distributed training need to be supported in the management systems.

Distributed training refers to the approach of distributed computation to scale out a training job, either to accelerate the process or to handle workloads that cannot fit into a single machine.

Management of distributed Training can be used for AI/ML-based use cases specified in [2] and [3]. In 5GS, distributed training can apply across various deployment scenarios for the ML training function. These functions may be located within the 3GPP management system, domain-specific management functions (e.g., RAN or CN), or directly in Network Functions such as the gNB or NWDAF. The location of these functions depends on the specific scenario defined in clause 4a.2.

When receiving an ML training request, the MLT MnS producer may evaluate whether distributed training is needed according to the training requirements provided by the ML training consumer, and it is up to the MLT MnS producer to determine, based on some information (e.g target inference location) provided by the consumer, appropriate training function(s) which may need to participate in the ML model training. The training requirement may further include (not limited to) expected model performance. Collaboration, mutual agreement and authentication procedures are needed to be established between distributed ML training functions before sharing any information between these functions.

The actions of ML model distributed training may involve for example, splitting the training of an ML model across many ML training functions, each responsible for computing a portion of the ML model’s operations. Since the training data may be sparse, MLT MnS consumer may provide indication that the training data should not be split while splitting the training among multiple training functions.

NOTE 1: How to split the ML model and synchronize the parameters in different training function depends on the distributed algorithm which are proprietary and not in scope for standardization.

NOTE 2: The data exchange between different training functions should be in the security tunnel with appropriate authentication and authorization mechanisms.

|  |
| --- |
| **Next of Change** |

#### 6.2b.2.X6 Management of Federated learning

##### 6.2b.2.X6.1 Description

Federated learning (FL) is a distributed machine learning approach that allows multiple FL clients to collaboratively train an ML model on local datasets contained in each FL Client without explicitly exchanging data samples.

FL is supported by a group of FL clients and FL server wherein FL client keeps the data localized and private and trains the ML model directly on the local nodes (client) where the data is obtained or stored.

Federated learning can be categorized into two main types: Horizontal federated learning (HFL) and Vertical federated learning (VFL), based on the nature of the data distribution and the way the model training is orchestrated among participants. For HFL, the process typically includes FL Client discovery and selection, local ML model training and updates by the FL Clients, ML model updates aggregation, and global ML model distribution by the FL Server. Management of Federated learning can be used for AI/ML-based use cases specified in [2] and [3].

****

**Figure 6.2b.2.X6.1-1: ML model distribution and aggregation for HFL**

NOTE: A prior agreement as well as authentication procedures needs to be established between FL Server and FL clients before sharing any information between these functions.

##### 6.2b.2.X6.2 Use cases

###### 6.2b.2.X6.2.1 Management of different roles in Federated learning

For FL, an ML model is collaboratively trained by a group of FL clients (e.g. MTLF in NWDAFs) including one acting as FL server and the others acting as FL clients. Federated Learning training allows multiple FL clients to collaboratively train an ML model on local datasets.

For managing the FL, the ML training MnS consumer needs to know the FL clients and FL server involved in the FL, so that the consumer understands the impact of each one of them and can manage them correspondingly.

When receiving an ML training request, the ML training MnS Producer should evaluate whether FL process needs to be started according to the training requirements (e.g. minimum number of FL Clients, minimum number of total iterations, minimum number of data samples and available time of the FL Clients, fault tolerance, energy source and carbon emission information) provided by the ML training consumer. Based on the received requirements, the ML training MnS Producer with the role of FL server may select (including adding and removing) appropriate FL Clients.

To evaluate the performance of FL, the consumer can query the performance of the final global ML model running on the local training data set of participating FL clients. For instance, if an FL server cannot produce a global ML model with satisfied performance for the FL clients, the consumer may interact with the MnS ML training producer to optimize the FL for future training, e.g. updating the criteria for selecting FL clients.

In addition, the consumer needs to get the information about the contribution of each FL client to the FL process, for instance, the number of iterations in which the FL client participated in the FL, the number of data samples the FL client used and the training duration performed by the FL Client.

|  |
| --- |
| **Next change** |

#### 6.2b.2.X7 Management of Reinforcement Learning

##### 6.2b.2.X7.1 Description

In RL, the agent uses a trial-and-error approach to develop a policy that maximises its cumulative reward over time.

RL can be used for supporting various AI/ML functions (including NG-RAN AI/ML supported functions, e.g., Energy Saving (ES), Mobility Load Balancing (MLB) and Mobility Robustness Optimization (MRO), etc.) when the ML training function is located in OAM. For these scenarios, RL needs to be managed.

An RL agent functionally consists of two main parts: a training component and an inference component.

The training component learns to generate or update the RL model from the outcomes of its actions taken by the inference component in an RL environment (e.g., a live 5G subnetwork or a simulated environment). RL can occur in one of two modes for:

- **RL in online mode**: in online mode, the RL model is trained and applied in real time through direct interaction between the RL agent components and the RL environment. In this approach, the RL agent’s training component continuously learns by receiving rewards, and observing state transitions resulting from its actions, while the RL agent’s inference component uses the trained model in real time to determine actions in the RL environment.



**Figure 6.2b.2.X7.1-1: RL in online mode**

For RL operating in online mode within a target network, direct interaction with the network occurs in real-time. As a result, both ML model testing and AI/ML emulation steps may not be executed as a distinct step, and if the RL agent training components and RL agent inference component are not co-located, model deployment may occur automatically between them without an external deployment trigger.

NOTE x1: RL in online mode is not supported in current release specifications.

- **RL in offline mode**: In offline mode, instead of direct interaction, the training process relies on a pre-collected dataset from a data collection entity (e.g., the MnS producer for collecting the performance measurements, Trace events, MDT/UE level measurements, alarm information, and/or configuration parameters updates, etc.). This dataset consists of state transition, action, and reward information collected over a period of time from the RL environment, allowing the RL agent training component to train the model without real-time interaction with the environment.



**Figure 6.2b.2.X7.1-2: RL in offline mode**

For RL in offline mode, the RL model may undergo ML model testing and AI/ML emulation steps before being deployed in the target network.

For RL in online mode, the RL model cannot be tested as much as RL in offline mode. If online RL agent training is discontinued in the target network, the RL agent inference component continues to operate, and the rewards and state may not be sent to the RL agent training component from the RL environment.

The RL environment will be impacted by actions of the agent. Agent’s action depends on input data, which comes from the RL environment. The RL agent needs the data samples for training regardless of the RL environment they come from.

NOTE x2: Rewards, states, and actions are not subject to standardization. While measurements and KPIs may be defined, the mapping of such data to states and rewards is implementation-specific.

NOTE x3: Figures 6.2b.2.X7.1-1 and 6.2b.2.X7.1-2 conceptually and logically illustrate how the RL process works in both online and offline modes, without restricting the implementation of RL agent training and inference components.

##### 6.2b.2.X7.2 Use cases

###### 6.2b.2.X7.2.1 Enabling Reinforcement Learning

Once the RL model is trained and deployed (if training and inference occur on separate entities), the inference component adopts the RL model and makes decisions based on state transitions in the RL environment.

To enable, facilitate, and manage RL for 5GS, the 3GPP management system needs to support the following RL components in performing their respective functions:

- **RL agent training component:** collects the data for training, trains the RL model and reports the training result of the training component. The data for training includes state transitions, rewards from the RL environment for online RL, or the pre-processed data set for a period of time from the data collection entity for offline RL, as well as the actions taken by the RL agent inference component. The trained RL model would be used by the inference component to determine actions for observed state conditions. Additionally, the RL agent training component needs to indicate the RL environment(s) (e.g., a live subnetwork or a simulation subnetwork) for which an RL model has been trained.

- **RL agent inference component:** activates the trained RL model (RL policy), receives state transitions from the RL environment, and determine actions based on the RL model. The RL agent inference component needs to be configured with the information about the RL environment(s) where a trained RL model can be used.

- **RL environment:** executes the actions determined by the RL agent inference component and reports rewards to the RL agent training component for online RL or to the data collection entity for offline RL.

###### 6.2b.2.X7.2.2 Exploration in Reinforcement Learning

Reinforcement Learning (RL) has the ability to learn and adapt itself to dynamic environments and thus finds the near optimal solution for a problem. However, the potential negative impact to the mobile network caused by RL is still the main drawback. In particular, during the exploration step performing trials and learning from errors may have an impact on the operational network and may result in unsafe operations causing network performance degradations. Therefore, the exploration step in RL needs to be under a controlled configuration range that the RL agent is allowed to act upon, so that the RL actions do not violate system performance requirements. If the RL agent behaves in an unexpected manner, there needs to be a set of fall-back actions in place, e.g. to switch from RL-based solution to non-RL-based solution, to fall back to last discrete time step, and to terminate the RL process.

When RL is supported, a consumer may want to provide a scope (e.g. geographical area, time window) that can aid the MnS producer to select/create the environment when performing RL. The environment may include information of the entities the RL agent may impact when performing RL, i.e., the allowed scope for entities to be impacted by RL actions. If the MnS producer supports multiple types of environments, the consumer may want to state their preference for environment type for RL during training, i.e., simulated environment or live network. When the live network is preferred by the MnS consumer, the consumer can provide network performance requirements (e.g. lower bound threshold, acceptable range, maximum performance deterioration Rate, etc.) of performing ML training of RL, to make the MnS producer adapt the training configurations to meet the network performance requirements. Furthermore, the consumer can provide its concerned performance metrics to guide the MnS producer to set reward/state.

NOTE: Support for both environment types can be considered optional in the RL training.

#### 6.2b.2.X8 Training data statistics

During ML training, it is important to ensure that training data is as uniform and representative as possible, and outliers are handled appropriately during the data pre-processing to train robust ML models that do not require frequent re-training.

Non-uniform distribution of training data can significantly degrade the performance of trained **ML models.** The ML model may learn and reflect this non-uniformity, rather than the true underlying patterns in the data. This could lead to inaccurate predictions when the ML model is deployed for inference. Furthermore, an ML model trained on non-uniform data may not generalize well to new unseen data, as the training data might not accurately represent the full range of possible inputs the ML model may encounter during inference. This can limit the usefulness of the ML model.

Similarly, outliers in training data can significantly degrade the performance of trained ML model. ML models learn to make predictions based on the patterns they identify in the training data. Outliers can skew these patterns and lead to an ML model that is biased towards these extreme values, rather than accurately reflecting the majority of the data.

Therefore, it could be useful to ensure that training data is as uniform and representative as possible, and outliers are handled appropriately during the data pre-processing to train robust ML models that do not require frequent re-training.

***Next change***

6.2b.3 Requirements for ML model training

**Table 6.2b.3-1**

| **Requirement label** | **Description** | **Related use case(s)** |
| --- | --- | --- |
| **REQ-ML\_TRAIN-FUN-01** | The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to request ML model training. | ML model training requested by consumer (clause 6.2b.2.1) |
| **REQ- ML\_TRAIN-FUN-02** | The ML training MnS producer shall have a capability allowing the authorized ML training MnS consumer to specify the data sources containing the candidate training data for ML model training. | ML model training requested by consumer (clause 6.2b.2.1) |
| **REQ- ML\_TRAIN-FUN-03** | The ML training MnS producer shall have a capability allowing the authorized ML training MnS consumer to specify the AI/ML inference name of the ML model to be trained. | ML model training requested by consumer (clause 6.2b.2.1) |
| **REQ- ML\_TRAIN-FUN-04** | The ML training MnS producer shall have a capability to provide the training result to the ML training MnS consumer. | ML model training requested by consumer (clause 6.2b.2.1), ML model training initiated by MnS producer (clause 6.2b.2.2) |
| **REQ- ML\_TRAIN-FUN-05** | The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to configure the thresholds of the performance measurements and/or KPIs to trigger the re-training of an ML model. (See Note) | ML model training initiated by MnS producer (clause 6.2b.2.2) |
| **REQ- ML\_TRAIN-FUN-06** | The ML training MnS producer shall have a capability to provide the version number of the ML model when it is generated by ML model re-training to the authorized ML training MnS consumer. | ML model training requested by consumer (clause 6.2b.2.1), ML model training initiated by MnS producer (clause 6.2b.2.2) |
| **REQ- ML\_TRAIN-FUN-07** | The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to manage the training process, including starting, suspending, or resuming the training process, and configuring the ML context for ML model training. | ML model training requested by consumer (clause 6.2b.2.1), ML model training initiated by MnS producer (clause 6.2b.2.2), ML model joint training (clause 6.2b.2.6) |
| **REQ- ML\_TRAIN-FUN-08** | The ML training MnS producer should have a capability to provide the grouping of ML models to an authorized ML training MnS consumer to enable coordinated inference. | ML model joint training (clause 6.2b.1.2.6) |
| **REQ- ML\_TRAIN-FUN-09** | The ML training MnS producer should have a capability to allow an authorized ML training MnS consumer to request joint training of a group of ML models. | ML model joint training (clause 6.2b.2.6) |
| **REQ- ML\_TRAIN-FUN-10** | The ML training MnS producer should have a capability to jointly train a group of ML models and provide the training results to an authorized consumer. | ML model joint training (clause 6.2b.2.6) |
| **REQ-ML\_SELECT-01** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer to discover the properties of available ML models including the contexts under which each of the models were trained. | ML model and ML model selection (clause 6.2b.2.3) |
| **REQ-ML\_SELECT-02** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer to select an ML model to be used for inference. | ML models and ML model selection (clause 6.2b.2.3) |
| **REQ-ML\_SELECT-03** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer to request for information and be informed about the available alternative ML models of differing complexity and performance. | ML model and ML model selection (clause 6.2b.2.3) |
| **REQ-ML\_SELECT-04** | The 3GPP management system shall have a capability to provide a selected ML model to the authorized ML training MnS consumer. | ML model and ML model selection (clause 6.2b.2.3) |
| **REQ-ML\_TRAIN- MGT-01** | The ML training MnS producer shall have a capability allowing an authorized consumer to manage and configure one or more requests for the specific ML model training, e.g. to modify the request or to delete the request. | ML model training requested by consumer (clause 6.2b.2.1), Managing ML model Training Processes (clause 6.2b.2.4) |
| **REQ-ML\_TRAIN- MGT-02** | The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to manage and configure one or more training processes, e.g. to start, suspend or restart the training. | ML model training requested by consumer (clause 6.2b.2.1),  Managing ML model training processes (clause 6.2b.2.4) |
| **REQ-ML\_TRAIN- MGT-03** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer (e.g. the function/model different from the function that generated a request for ML model training) to request for a report on the outcomes of a specific training instance. | Managing ML model training processes (clause 6.2b.2.4) |
| **REQ-ML\_TRAIN- MGT-04** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer to define the reporting characteristics related to a specific training request or training instance. | Managing ML model training processes (clause 6.2b.2.4) |
| **REQ-ML\_TRAIN- MGT-05** | 3GPP management system shall have a capability to enable the ML training function to report to any authorized ML training MnS consumer about specific ML model training process and/or report about the outcomes of any such ML model training process. | Managing ML model training processes (clause 6.2b.2.4) |
| **REQ-ML\_ERROR-01** | The 3GPP management system shall enable an authorized consumer of data services (e.g. an ML training function) to request from an MnS producer of data services a Value Quality Score of the data, which is the numerical value that represents the dependability/quality of a given observation and measurement type. | Handling errors in data and ML decisions (clause 6.2b.2.5) |
| **REQ-ML\_ERROR-02** | The 3GPP management system shall enable an authorized consumer of AI/ML decisions (e.g. a controller) to request ML decision confidence score which is the numerical value that represents the dependability/quality of a given decision generated by an AI/ML inference function. | Handling errors in data and ML decisions (clause 6.2b.2.5) |
| **REQ-ML\_ERROR-03** | The 3GPP management system shall have a capability to enable an authorized consumer to provide to the ML training MnS producer, a training data quality score, which is the numerical value that represents the dependability/quality of a given observation and measurement type. | Handling errors in data and ML decisions (clause 6.2b.2.5) |
| **REQ-ML\_ERROR-04** | The 3GPP management system shall enable an MnS producer of ML decisions (e.g. an AI/ML inference function) to provide to an authorized consumer of ML decisions (e.g. a controller) an AI/ML decision confidence score which is the numerical value that represents the dependability/quality of a given decision generated by the AI/ML inference function. | Handling errors in data and ML decisions (clause 6.2b.2.5) |
| **REQ-ML\_VLD-01** | The ML training MnS producer should have a capability to validate the ML models during the ML model training process and report the performance of the ML models on both the training data and validation data to the authorized consumer. | ML model validation performance reporting (clause 6.2b.2.7) |
| **REQ-ML\_VLD-02** | The ML training MnS producer should have a capability to report the ratio (in terms of quantity of data samples) of the training data and validation data used during the ML model training and validation process. | ML model validation performance reporting (clause 6.2b.2.7) |
| **REQ-TRAIN\_EFF-01** | The 3GPP management system should have the capability to allow an authorized consumer to configure an ML training function to report the effectiveness of data used for model training. | Training data effectiveness reporting (clause 6.2b.2.8) |
| **REQ-ML\_TRAIN\_PM-1** | The ML training MnS producer should have a capability to allow an authorized consumer to get the capabilities about what kind of ML models the ML training function is able to train. | Performance indicator selection for ML model training (clause 6.2b.2.9.2) |
| **REQ-ML\_TRAIN\_PM-2** | The ML training MnS producer should have a capability to allow an authorized consumer to query what performance indicators are supported by the ML model training for each ML model. | ML model performance indicators query and selection for ML model training (clause 6.2b.2.9.3) |
| **REQ-ML\_TRAIN\_PM-3** | The ML training MnS producer should have a capability to allow an authorized consumer to select the performance indicators from those supported by the ML training function for reporting the training performance for each ML model. | ML model performance indicators query and selection for ML model training (clause 6.2b.2.9.3) |
| **REQ-ML\_TRAIN\_PM-4** | The ML training MnS producer should have a capability to allow an authorized consumer to provide the performance requirements for the ML model training using the selected the performance indicators from those supported by the ML training function. | MnS consumer policy-based selection of ML model performance indicators for ML model training (clause 6.2b.2.9.4) |
| **REQ-MLKTL-1** | The 3GPP management systemshould have a capability to enable an authorized MnS consumer to discover or request all or part of the available shareable knowledge at a given MLKTL MnS producer. | ML-Knowledge-based Transfer Learning (clause 6.2b.2.X1.2.1) |
| **REQ-MLKTL-2** | The 3GPP management systemshould have a capability for an MLKTL MnS producer to provide to an authorized MnS consumer all or part of its available shareable knowledge | ML-Knowledge-based Transfer Learning (clause 6.2b.2.X1.2.1) |
| **REQ-MLKTL-3** | The 3GPP management systemshould have a capability enabling an authorized MnS consumer to request a MLKTL MnS producer to initiate and execute a transfer learning instance to a specified ML model or ML-enabled function | ML-Knowledge-based Transfer Learning (clause 6.2b.2.X1.2.2) |
| **REQ-MLKTL-4** | The 3GPP management systemshould have a capability to enable an authorized MnS consumer to manage or control the knowledge request or the knowledge process or transfer learning process, e.g. to suspend, re-activate or cancel the ML Knowledge Request; or to adjust the description of the desired knowledge  NOTE: An example MnS consumers include an operator or the function that generated the request for available Knowledge | ML-Knowledge-based Transfer Learning (clause 6.2b.2.X1.2.2) |
| **REQ-MLKTL-5** | The 3GPP management systemshould have a capability to enable an ML model or ML training function to register available knowledge to a shared knowledge repository, e.g. through a ML Knowledge Registration process | ML-Knowledge-based Transfer Learning (clause 6.2b.2.X1.2.2) |
| **REQ-MLKTL-6** | The 3GPP management systemshould have a capability enabling an authorized MnS consumer to request the Knowledge Repository to provide some or all the knowledge available for sharing based on specific criteria | ML-Knowledge-based Transfer Learning (clause 6.2b.2.X1.2.2) |
| **REQ-ML\_TRAIN\_CLUSTER-01** | The ML training MnS producer should have a capability for an authorized MnS consumer to request training of a cluster of ML models as per clustering criteria associated to a set of multiple contexts from a previously trained ML model. | ML model training for multiple contexts (clause 6.2b.2.X2) |
| **REQ-ML\_TRAIN-PRE-01** | The ML training MnS producer should have a capability allowing an authorized ML training MnS consumer to request pre-specialized training of a ML model. | ML Pre-specialised training (clause 6.2b.2.X3) |
| **REQ- ML\_TRAIN-FT-x1** | The ML training MnS producer should have a capability to enable an authorised consumer to request the fine-tuning of a pre-specialised trained ML model. | ML Pre-specialised training (clause 6.2b.2.X3), ML fine-Tuning (clause 6.2b.2.X4) |
| **REQ- ML\_TRAIN-FT-x2** | The ML training MnS producer should have a capability allowing the consumer to specify the training type of an ML model training request such as pre-specialised training, and fine-tuning. | ML Pre-specialised training (clause 6.2b.2.X3), ML fine-Tuning (clause 6.2b.2.X4) |
| **REQ-ML\_DIST-TRNG-01** | The ML training MnS producer should have a capability allowing and authorized consumer to provide distributed training requirements to the MnS Producer. | Management of distributed training (clause 6.2b.2.X5) |
| **REQ-ML\_TRAIN\_FL-1** | The ML training MnS producer should have a capability allowing an authorized consumer to discover the FL roles (FL server or FL client) in Federated learning. | Management of different roles in Federated learning (Clause 6.2b.2.X6.2.1) |
| **REQ-ML\_TRAIN\_FL-2** | The ML training MnS producer should have a capability allowing an authorized consumer to provide FL training requirements to the MnS Producer. | Management of different roles in Federated learning (Clause 6.2b.2.X6.2.1) |
| **REQ-ML\_TRAIN\_FL-3** | The ML training MnS producer should have a capability allowing an authorized consumer to provide requirements for selecting (including adding and removing) FL clients in Federated Learning to the MnS Producer. | Management of different roles in Federated learning (Clause 6.2b.2.X6.2.1) |
| **REQ-ML\_TRAIN\_FL-4** | The ML training MnS producer should have a capability allowing an authorized consumer to get the performance of the global ML model on each participating FL Client. | Management of different roles in Federated learning (Clause 6.2b.2.X6.2.1) |
| **REQ-ML\_TRAIN\_FL-5** | The ML training MnS producer should have a capability to report the information about the contribution of each FL client to the FL process to MnS consumer. | Management of different roles in Federated learning (Clause 6.2b.2.X6.2.1) |
| **REQ-ML\_TRAIN\_FL-6** | The ML training MnS producer should have a capability to report the candidate FL Clients for the FL process. | Management of different roles in Federated learning (Clause 6.2b.2.X6.2.1) |
| **REQ-RL\_TRAIN\_01** | The ML training MnS producer should have a capability allowing an authorized MnS consumer to query if RL training is supported. | Enabling Reinforcement learning (6.2b.2.X7.2.1) |
| **REQ-RL\_TRAIN\_02** | The ML training MnS producer should have a capability to report RL types (i.e., online RL, offline RL) to an authorized consumer. | Enabling Reinforcement learning (6.2b.2.X7.2.1) |
| **REQ-RL\_TRAIN\_03** | The ML training MnS producer should have a capability to allow an authorized consumer to get the type and scope of the RL environment for which an RL model has been trained. | Exploration in Reinforcement learning (6.2b.2.X7.2.2) |
| **REQ-RL\_TRAIN\_04** | The ML training MnS producer should have a capability to allow an authorized consumer to select the type of the RL environment for which an RL model is to be trained. | Exploration in Reinforcement learning (6.2b.2.X7.2.2) |
| **REQ-RL\_TRAIN\_05** | The ML training MnS producer should have a capability to allow an authorized consumer to provide the scope of the RL environment for which an RL model is to be trained. | Exploration in Reinforcement learning (6.2b.2.X7.2.2) |
| **REQ-RL\_TRAIN\_06** | The ML training MnS producer should have a capability allowing an authorized MnS consumer to provide network performance requirements of performing RL training. | Exploration in Reinforcement learning (6.2b.2.X7.2.2) |
| **REQ-RL\_TRAIN\_07** | The ML training MnS producer should have a capability to allow an authorized MnS consumer to specify the configuration range that the RL agent is allowed to explore. | Exploration in Reinforcement learning (6.2b.2.X7.2.2) |
| **REQ-RL\_TRAIN\_08** | The ML training MnS producer should have a capability to allow an authorized consumer to provide the allowed scope for the entities to be impacted by the RL actions. | Exploration in Reinforcement learning (6.2b.2.X7  .2) |
| **REQ-ML\_TRAIN\_DST-01** | The 3GPP management system should enable an authorized consumer to provide information on the training dataset distribution. | Training data statistics (clause 6.2b.2.X8) |
| **REQ-ML\_TRAIN\_DST-02** | The 3GPP management system should enable an authorized consumer to provide information on the usage of outliers in the training dataset. | Training data statistics (clause 6.2b.2.X8) |
| NOTE: The performance measurements and KPIs are specific to each type (i.e., the inference type that the ML model supports) of ML model. | | |

***Next change***

6.3 AI/ML inference emulation

### 6.3.1 Description

Before the ML model is applied in the production network, the MnS inference consumer may want to receive results of inference in one or more environments that emulate (to different extents) the expected inference characteristics, in a process that may be termed as Inference emulation. The Inference emulation phase enables this.

### 6.3.2 Use cases

#### 6.3.2.1 AI/ML inference emulation

After the ML model is validated and tested during development, the MnS consumer may wish to receive information from an inference emulation process that indicates if the ML model or the associated ML inference function is working correctly under certain runtime context.

The management system should have the capabilities enabling an MnS consumer:

- request an inference emulation function to provide emulation reports; and

- to receive the results from running inference through an AI/ML inference emulation environment available at the emulation MnS producer.

#### 6.3.2.X1 ML inference emulation environment selection

Although an ML model may be well-trained, its performance in the production network can be difficult to predict and guarantee because the training environment and production network are not identical. If a trained or tested ML model is directly applied to the production network, it may negatively impact the production network.

ML emulation involves applying the ML model in an emulation environment to verify whether its performance meets the expected inference characteristics. Considering the diversity of ML inference scenarios, one or more emulation environments are provided, each differing in terms of emulation scope, emulation performance, and other factors.

The management system should have the capability to enable an MnS consumer to select the appropriate emulation environment and provide the necessary configuration properties related to that environment. The configuration information may include defining the scope of the emulation, the emulation time, and other relevant parameters.

## 6.3.3 Requirements for Managing AI/ML inference emulation

**Table 6.3.3-1**

| **Requirement label** | **Description** | **Related use case(s)** |
| --- | --- | --- |
| **REQ-AI/ML\_EMUL-1:** | The MnS producer for AI/ML inference emulation should have a capability enabling an authorized MnS consumer to receive reporting about the ML inference emulation. | AI/ML Inference emulation (clause 6.3.2.1) |
| **REQ-AI/ML\_EMUL-2:** | The MnS producer for AI/ML inference emulation should have a capability enabling an authorized MnS consumer to request an inference emulation function to provide inference emulation reports on an ML model or inference Function. | AI/ML Inference emulation (clause 6.3.2.1) |
| **REQ-EMUL\_SEL-1** | The MnS producer for AI/ML inference emulation should have a capability enabling an authorized MnS consumer to select the emulation environment. | ML inference emulation environment selection (clause 6.3.2.X1) |

**Next change**

6.4 ML model deployment

6.4.1 ML model loading

6.4.1.1 Description

ML model loading refers to the process of making an ML model available for use in the inference function. After a trained ML model meets the performance criteria per the ML model testing and optionally ML emulation, the ML model could be loaded into the target inference function(s) in the system. The way for loading the ML model is not in scope of the present document.

6.4.1.2 Use cases

6.4.1.2.1 Consumer requested ML model loading

After a trained ML model or the coordination group of ML models are tested and optionally emulated, if the performance of the ML model or the coordination group of ML models meet the MnS consumer’s requirements, the MnS consumer may request to load the one or more ML models to one or more target inference function(s) where the ML models will be used for conducting inference. Once the ML models loading request is accepted, the MnS consumer (e.g., operator) needs to know the progress of the loading and needs to be able to control (e.g., cancel, suspend, resume) the loading process. For a completed ML model loading, the ML model instance loaded to each target inference function needs to be manageable individually, for instance, to be activated/deactivated individually or concurrently.

6.4.1.2.2 Control of producer-initiated ML model loading

To enable more autonomous AI/ML operations, the MnS producer is allowed to load the ML model or the coordination group of ML models without the consumer’s specific request.

In this case, the consumer needs to be able to set the policy for the ML loading, to make sure that ML models loaded by the MnS producer meet the performance target. The policy could be, for example, the threshold of the testing performance of the ML models, the threshold of the inference performance of the existing ML model, the time schedule allowed for ML model loading, etc.

ML models are typically trained and tested to meet specific requirements for inference, addressing a specific use case or task. The network conditions may change regularly, for example, the gNB providing coverage for a specific location is scheduled to accommodate different load levels and/or patterns of services at different times of the day, or on different days in a week. One or more ML models may be loaded per the policy to adapt to a specific load/traffic pattern.

6.4.1.2.3 ML model registration

After multiple iterations, there could be a large number of ML models with different versions, deployment environments, performance levels, and functionalities. ML model registration refers to the process of recording, tracking, controlling those trained ML models enabling future retrieval, reproducibility, sharing and loading in the target inference functions across different environments. For example, the inference MnS consumer could recall the most applicable version dealing with a sudden changed deployment environment of the target inference function by tracking the registration information.

The ML training MnS producer should register the ML model along with its loading information, e.g., ML model metadata and relevant information (e.g., description, version, version date, target inference function, deployment environment, etc.).

6.4.1.3 Requirements for ML model loading

**Table 6.4.1.3-1**

| **Requirement label** | **Description** | **Related use case(s)** |
| --- | --- | --- |
| **REQ- ML\_LOAD-FUN-01** | The MnS producer for ML model loading shall have a capability allowing an authorized consumer to request to trigger loading of one or more ML model(s). | Consumer requested ML model loading (clause 6.4.1.2.1) |
| **REQ- ML\_LOAD-FUN-02** | The MnS producer for ML model loading shall have a capability allowing an authorized consumer to provide a policy for the MnS producer to trigger loading of one or more ML model(s). | Producer-initiated ML model loading (clause 6.4.1.2.2) |
| **REQ- ML\_LOAD-FUN-03** | The MnS producer for ML model loading shall be able to inform an authorized consumer about the progress of ML model loading. | Consumer requested ML model loading (clause 6.4.1.2.1) and Producer-initiated ML model loading (clause 6.4.1.2.2) |
| **REQ- ML\_LOAD-FUN-04** | The MnS producer for ML model loading shall have a capability allowing an authorized consumer to control the process of ML model loading. | Consumer requested ML model loading (clause 6.4.1.2.1) and Producer-initiated ML model loading (clause 6.4.1.2.2) |
| **REQ- ML\_REG-01** | The ML training MnS producer should have a capability to register an ML model to record the relevant information that may be used for loading. | ML model registration (Clause 6.4.1.2.3) |
| **REQ- ML\_REG-02** | The ML training MnS producer should have a capability to allow an authorized consumer (e.g., an AI/ML inference function) to acquire the registration information of ML models. | ML model registration (Clause 6.4.1.2.3) |

**Next change**

6.5 AI/ML inference

### 6.5.X1 Managing ML models in use in a live network

#### 6.5.X1.1 Description

These use cases deal with managing ML models that are in-use or deployed in the live network. These use cases will apply to all supported AI/ML inference capabilities.

#### 6.5.X1.2 Use cases

##### 6.5.X1.2.1 Handling of underperforming ML trained models in live networks

Actions may need to be taken by a network operator once trained ML models has been identified that is contributing towards non-optimal running of the network. These actions may involve for example, without service interruptions, reverting to running of the network without ML based optimizations (i.e deavtivating the AIMLInferenceFunction) or replacing current ML model with an earlier model one that was performing better (i.e deploying a new MLmodel).

##### 6.5.X1.2.2 Performance monitoring of Network Functions with ML trained models in live networks

Several trained ML models maybe in use in an operator network with each one of them influencing network performance. The Network Functions with these ML models need to be monitored to ensure network is running optimally with these models in use. KPIs for evaluating runtime performance of Network Functions using ML models should be provided for this purpose as part of the model. These KPIs correspond to specific network functions of 5GC AI/ML supported functions and NG-RAN AI/ML supported functions and would need to be collectable by the network operator.

#### 6.5.X1.3 Requirements for Managing ML models in use in a live network

Table 6.5.X1.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-AI/ML\_INF-LIVES-01** | The 3GPP management system should have a capability enabling an authorized consumer to identify ML model(s) that caused performance measurement and/or KPI degradation. | Handling of underperforming ML trained models in live networks (6.5.X1.2.1) |
| **REQ-AI/ML\_INF-LIVES-02** | The 3GPP management system should have a capability to recommend remedial actions to address performance measurement and/or KPI degradation caused by ML model(s). | Handling of underperforming ML trained models in live networks (6.5.X1.2.1) |
| **REQ-AI/ML\_INF-LIVES-03** | The 3GPP management system should be able to collect network related performance data pertaining to Network Functions actively utilizing ML models. | Performance monitoring of Network Functions with ML trained models in live networks (6.5.X1.2.2) |

**Next change**

6.5.X2 AI/ML inference explainability

6.5.X2.1 Description

Explainable ML refers to a process that enables the consumers (e.g. operator) to understand and trust the outputs provided by ML models. In essence, AI/ML inference explainable is about making the decision-making of ML comprehensible to its consumers. The aim is to explain individual outputs provided by an ML model, i.e., it focuses on explaining why a specific output was generated by the ML model for a particular input data sample.

6.5.X2.2 Use cases

##### 6.5.X2.2.1 Management of explanation in AI/ML inference

Once the ML model is trained and deployed for inference, the generated explanations by the AI/ML inference MnS producer may need to be reported to the AI/ML inference MnS consumer. These explanations explain why this ML model has produced a certain inference result. These explanations can be used by MnS consumers to understand to making more informed decisions for network operationregarding the deactivation of AI/ML inference or fallback to a previous version of the ML model. E.g. the explanation of ML model inference results typically involves the following aspects to understand why the model produces a specific inference result, such as the critical features in the training or inference data.

#### 6.5.X2.3 Requirements for AI/ML inference explainability managment

Table 6.5.X2.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-ML-INF-EXP-01** | The 3GPP management system should have a capability to report the explanations for AI/ML inference to an authorized consumer. | Management of explanation in AI/ML inference (clause 6.5.X2.2.1) |

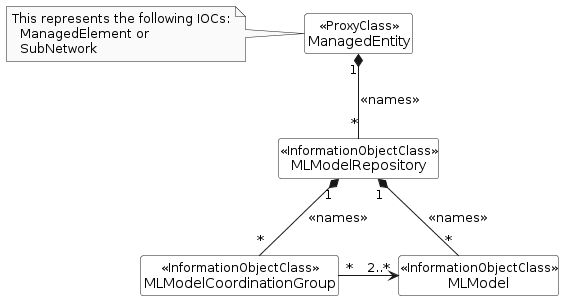
***Next change***

# 7 Information model definitions for AI/ML management

7.2a Common information model definitions for AI/ML management

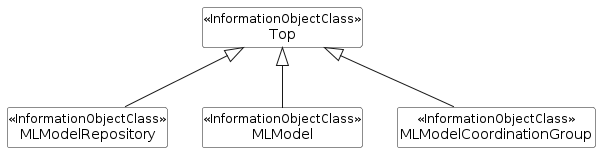
7.2a.1 Class diagram

7.2a.1.1 Relationships



**Figure 7.2a.1.1-1: Relations for common information models for AI/ML management**

7.2a.1.2 Inheritance



**Figure 7.2a.1.2-1: Inheritance Hierarchy for common information models for AI/ML management**

7.2a.2 Class definitions

7.2a.2.1 MLModel

7.2a.2.1.1 Definition

This IOC represents the ML model. ML model algorithm or ML model are not subject to standardization. It is name-contained by MLModelRepository.

This MLModel MOI can be created by the system (MnS producer) or pre-installed. The MnS consumer can request the system to delete the MLModel MOI.

The MLModel contains 3 types of contexts - TrainingContext, ExpectedRunTimeContext and RunTimeContext which represent status and conditions of the MLModel. These contexts are of mLContext <<dataType>>, see clauses 7.4.3 and 7.5.1 for details.

It also contains a reference named retrainingEventsMonitorRef which is a pointer to ThresholdMonitor MOI. This indicates the list of performance measurements and the corresponding thresholds that are monitored and used to identify the need for re-training by the MnS Producer. After the MLModel MOI has been instantiated, the MnS Consumer can request MnS producer to instantiate a ThresholdMonitor MOI and update the reference in the MLModel MOI that can be used by the MnS producer to decide on the re-training of the MLModel. The MnS producer can be ML training MnS producer or AI/ML Inference MnS Producer.

The ML model includes information about its applicable type of training, which includes pre-specialised training, fine-tuning, or re-training.

For a pre-specialised trained ML model, the MLModel MOI also include information about its applicable inference scope, which corresponds to a list of inference types which the model can be adapted (fine-tuned) to support.

7.2a.2.1.2 Attributes

The MLModel IOC includes attributes inherited from Top IOC (defined in TS 28.622 [12]) and the following attributes:

**Table 7.2a.2.1.2-1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | | **isWritable** | | **isInvariant** | **isNotifyable** |
| mLModelId | M | T | | F | | F | T |
| aIMLInferenceName | M | T | | F | | F | T |
| mLModelVersion | M | T | | F | | F | T |
| expectedRunTimeContext | M | T | | T | | F | T |
| trainingContext | CM | T | | F | | F | T |
| runTimeContext | O | T | | F | | F | T |
| supportedPerformanceIndicators | O | T | | F | | F | T |
| mLCapabilitiesInfoList | M | T | | F | | F | T |
| mLTrainingType | M | T | | F | | F | T |
| inferenceScope | CM | T | | F | | F | T |
| **Attribute related to role** |  |  | |  | |  |  |
| retrainingEventsMonitorRef | O | T | | T | | F | T |
| aIMLInferenceReportRefList | O | | T | | F | F | T |
| usedByFunctionRefList | O | | T | | F | F | T |

7.2a.2.1.3 Attribute constraints

**Table 7.2a.1.3-1**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| trainingContext | Condition: The trainingContext represents the status and conditions related to training and should be added when training is completed. |
| inferenceScope | Condition: When MLModel MOI represents the ML model which was trained by a pre-specialised training. |

7.2a.2.1.4 Notifications

The common notifications defined in clause 7.6 are valid for this IOC, without exceptions or additions.

## 7.3a Information model definitions for AI/ML operational phases

### 7.3a.1 Information model definitions for ML model training

#### 7.3a.1.2 Class definitions

##### 7.3a.1.2.1 MLTrainingFunction

###### 7.3a.1.2.1.1 Definition

The IOC MLTrainingFunction represents the model that undertakes ML model training. The MOI of MLTrainingFunction is also the container of the MLTrainingRequest, MLTrainingReport, MLTrainingProcess and ThresholdMonitor MOI(s).

This MLTrainingFunction instance is created by the system (MnS producer) or pre-installed, it can only be deleted by the system.

The ThresholdMonitor contains the list of performance measurements and the corresponding thresholds that are monitored and used to identify the need for ML model re-training by the MnS Producer.

TheML training function represented by MLTrainingFunction MOI supports training of one or more MLModel(s).

The MLTrainingFunction includes information about its applicable type of training, which includes pre-specialised training, fine-tuning, or re-training.

The MLTrainingFunction MOI have a supportedLearningTechnology attribute to indicate the supported learning technology including Reinforcement learning, Federated learning and distributed training. This attribute can enable the ML training MnS producer allowing ML training MnS consumer to query if RL/FL/DL is supported.

An MLTrainingFunction instance may contain a set of ML knowledge instances associated with a set of ML models that have been trained. An MnS consumer can find available ML knowledge by reading the mLKnowledge attribute on the ML MLTrainingFunction. Relatedly, the MnS consumer can find the characteristics of a specific ML knowledge instance by reading the related mLKnowledge. The request for training using MLknowledge is not to be combined with training using collected data – the training function should not provide ML knowledge along side the raw data used for creating the ML knowledge.

###### 7.3a.1.2.1.2 Attributes

The MLTrainingFunction IOC includes attributes inherited from ManagedFunction IOC (defined in TS 28.622 [12]) and the following attributes:

Table 7.3a.1.2.1.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| supportedLearningTechnology | M | T | F | F | T |
| mLKnowledge | O | T | F | F | T |
| mLTrainingType | M | T | F | F | T |
| **Attribute related to role** |  |  |  |  |  |
| mLModelRepositoryRef | M | T | F | F | T |

###### 7.3a.1.2.1.3 Attribute constraints

None.

###### 7.3a.1.2.1.4 Notifications

The common notifications defined in clause 7.6 are valid for this IOC, without exceptions or additions.

7.3a.1.2.2 MLTrainingRequest

7.3a.1.2.2.1 Definition

The IOC MLTrainingRequest represents the ML model training request that is trigered by the ML training MnS consumer.

To trigger the ML model training process, ML training MnS consumer needs create MLTrainingRequest object instances on the ML training MnS producer. The MLTrainingRequest MOI is contained under one MLTrainingFunction MOI.

The MLTrainingRequest MOI may represent the request for initial ML model training or re-training. For ML model re-training, the MLTrainingRequest is associated to one MLModel for re-training a single ML model, or associated to one MLModelCoordinationGroup.

The MLTrainingRequest includes information about a ML training type to define the type of training requested by the MnS consumer. The training type can be one of the following: (1) initial training, where the MnS consumer requests to train an ML model of which the instance does not exist yet, (2) pre-specialised training, where the ML model is trained on a dataset that is not specific to any particular type of inference, (3) re-training, where the ML model is re-trained on the same type of dataset on which it was previously trained to support the same type of inference, and (4) fine-tuning, where the ML model is trained to adapt it to support a new single type of inference.

The aIMLInferenceName means the inference type will be used for conducting inference.

The MLTrainingRequest has a source to identify where it is coming from, which is represented with trainingRequestSource attribute. This attribute may be used by an ML training MnS producer to prioritize the training resources for different sources.

Each MLTrainingRequest indicates the expectedRunTimeContext that describes the specific conditions for which the MLModel should be trained.

In case the request is accepted, the ML training MnS producer decides when to start the ML model training based on MnS consumer requirements. Once the MnS producer decides to start the training based on the request, the ML training MnS producer instantiates one or more MLTrainingProcess MOI(s) that are responsible to perform the followings:

- collects (more) data for training, if the training data are not available or the data are available but not sufficient for the training;

- prepares and selects the required training data, with consideration of the MnS consumer’s request provided candidate training data if any. The ML training MnS producer may examine the MnS consumer's provided candidate training data and select none, some or all of them for training. In addition, the ML training MnS producer may select some other training data that are available in order to meet the MnS consumer’s requirements for the ML model training.

- trains the MLModel using the selected and prepared training data.

The MLTrainingRequest may have a requestStatus field to represent the status of the specific MLTrainingRequest:

- The attribute values are "NOT\_STARTED", " IN\_PROGRESS", "SUSPENDED", "FINISHED", and "CANCELLED".

- When value turns to " IN\_PROGRESS", the ML training MnS producer instantiates one or more MLTrainingProcess MOI(s) representing the training process(es) being performed per the request and notifies the MLT MnS consumer(s) who subscribed to the notification.

When all of the training process associated to this request are completed, the value turns to "FINISHED".

The ML training MnS prodcuer shall delete the corresponding MLTrainingRequest instance in case of the status value turns to "FINISHED" or "CANCELLED". The MnS producer may notify the status of the request to MnS consumer after deleting MLTrainingRequest instance.

For the MLTrainingRequest used to trigger the ML model training of RL, the MLTrainingRequest MOI has an rLRequirement attribute to indicate the requirements of the RL.

The MLTrainingRequest can be used to trigger ML-knowledge-based transfer learning. The source ML knowledge should be indicated using the mLKnowledgeName, where the source does not want to reveal the source MLModel. The request for training using ML knowledge is not to be combined with training using collected data – the request cannot be for both mLKnowledgeName and candidateTrainingDataSource.

For the MLTrainingRequest to include clustering criteria, indicating which ML models with multiple contexts belonging to the same MnS producer can form the cluster and trained together, the MLTrainingRequest MOI is enhanced with attribute clusteringInfo containing information that provides the clustering criteria for the ML models to be trained together.

7.3a.1.2.2.2 Attributes

The MLTrainingRequest IOC includes attributes inherited from Top IOC (defined in TS 28.622 [12]) and the following attributes:

**Table 7.3a.1.2.2.1-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| aIMLInferenceName | CM | T | F | F | T |
| candidateTrainingDataSource | O | T | T | F | T |
| trainingDataQualityScore | O | T | T | F | T |
| trainingRequestSource | M | T | T | F | T |
| requestStatus | M | T | F | F | T |
| expectedRuntimeContext | M | T | T | F | T |
| performanceRequirements | M | T | T | F | T |
| rLRequirement | CM | T | T | F | T |
| cancelRequest | O | T | T | F | T |
| suspendRequest | O | T | T | F | T |
| trainingDataStatisticalProperties | O | T | T | F | T |
| distributedTrainingExpectation | O | T | T | F | T |
| mLKnowledgeName | CM | T | T | F | T |
| mLTrainingType | M | T | T | F | T |
| expectedInferenceScope | CM | T | T | F | T |
| clusteringInfo | O | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| mLModelRef | CM | T | F | F | T |
| mLModelCoordinationGroupRef | CM | T | F | F | T |

7.3a.1.2.2.3 Attribute constraints

**Table 7.3a.1.2.2.3-1**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| aIMLInferenceName | Condition: MLTrainingRequest MOI represents the request for ML model initial training. |
| mLModelRef | Condition: MLTrainingRequest MOI represents the request for ML model re-training. |
| mLModelCoordinationGroupRef | Condition: ML model joint training is supported. |
| mLKnowledgeName | Condition: Knowledge is indicated only if candidateTrainingDataSource is not indicated |
| rLRequirement | Condition: MLTrainingRequest MOI represents the request for Reinforcement learning |
| expectedInferenceScope | Condition: The MLTrainingRequest is for an ML model pre-specialised training. |

7.3a.1.2.2.4 Notifications

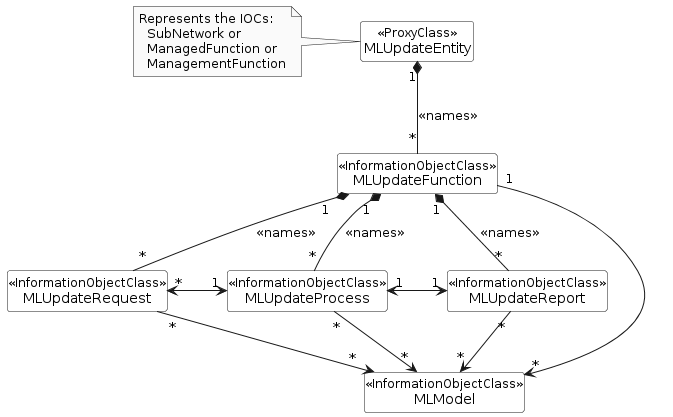
The common notifications defined in clause 7.6 are valid for this IOC, without exceptions or additions.

***Next change***

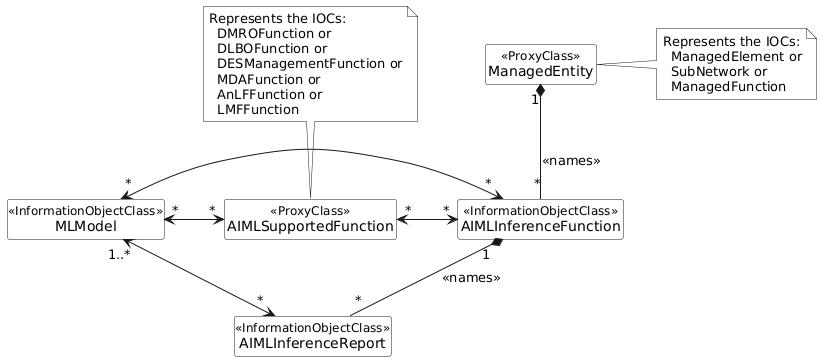
7.3a.4 Information model definitions for AI/ML inference

7.3a.4.1 Class diagram

7.3a.4.1.1 Relationships



**Figure 7.3a.4.1.1-1: NRM fragment for ML update**



NOTE 1: The ManagedEntity and AIMLSupportedFunction shall not represent the same MOI.

NOTE 2: For AnLFFunction, DMROFunction, DLBOFunction, and DESManagementFunction see [18] and for MDAFunction see [2].

**Figure 7.3a.4.1.1-2: NRM fragment for AI/ML inference function**

***Start of the next Change***

7.3a.4.2.5 AIMLInferenceFunction

7.3a.4.2.5.1 Definition

This IOC represents the common properties of the AI/ML inference function.

This AIMLInferenceFunction instance can be created by the system or pre-installed.

The AIMLInferenceFunction MOI may be associated with one or more MOIs that represent the functions/functionalities (Note) provided by the subject AIMLInferenceFunction MOI.

The AIMLInferenceFunction MOI can be only created by the MnS producer but not MnS consumer.

The MOI of AIMLInferenceFunction or the MOI of the IOC inheriting from the AIMLInferenceFunction IOC contains one or more MOI(s) of MLModel.

NOTE: The IOCs representing the functions/functionalities (Note) that use the AI/ML inference function include MDAFunction, AnLFFunction, DMROFunction, DLBOFunction, LMFFunction, and DESManagementFunction.

The AIMLInferenceFunction MOI may be contained by either a SubNetwork MOI, a ManagedElement MOI, or an MOI of ManagedFunction’s subclass, and it is allowed for an MnS producer to support multiple AIMLInferenceFunction MOIs contained in different superordinated MOIs among SubNetwork, ManagedElement and the ManagedFunction’s subclass.

The generation of inference outputs is based on the configuration of inference, e.g., to start a stated time, or to be executed at all times. The observations of the inference function and information on derived Outputs is registered in the inference report.

7.3a.4.2.6 AIMLInferenceReport

7.3a.4.2.6.1 Definition

This IOC represents a report from a AI/ML Inference.

An AIMLInferenceFunction may generate one or more AIMLInferenceReport(s).

Each AIMLInferenceReport provides information about inference outputs from one or more MLModel.

The AIMLInferenceReport also provides historical inference outputs for a series of time stamps.

The AIMLInferenceReport instance can be created by the MnS producer when creating an AIMLInferenceFunction instance.

The potentialImpactInfo includes impacted managed objects and network performance metrics, which can be utilized to manage the network performance to prevent network performance degradation.

7.3a.4.2.6.2 Attributes

The AIMLInferenceReport includes inherited attributes from Top IOC (defined in TS 28.622 [12]) and the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| inferenceOutputs | M | T | F | F | T |
| potentialImpactInfo | O | T | F | F | T |
| **Attributes related to role** |  |  |  |  |  |
| mLModelRef | M | T | F | F | T |

7.3a.4.2.6.3 Attribute constraints

None.

7.3a.4.2.6.4 Notifications

The common notifications defined in clause 7.6 are valid for this IOC, without exceptions or additions.

***Next change***

### 7.3a.3 Information model definitions for ML model deployment

#### 7.3a.3.1 Class diagram

##### 7.3a.3.1.1 Relationships

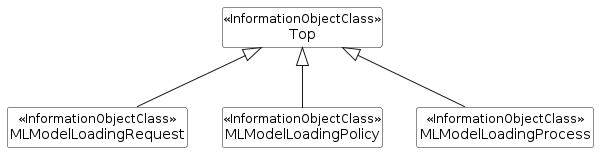
This clause depicts the set of classes (e.g. IOCs) that encapsulates the information relevant to ML model loading. For the UML semantics, see TS 32.156 [13].

A diagram of a model

AI-generated content may be incorrect.

**Figure 7.3a.3.1.1-1: NRM fragment for ML model loading**

7.3a.3.1.2 Inheritance



**Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML model loading related NRMs**

***Next change***

## 7.4 Data type definitions

### 7.4.X1 DataStatisticalProperties <<dataType>>

#### 7.4.X1.1 Definition

This data type specifies the data statistical properties that the training MnS producer should consider when preparing the training data for training an ML model.

#### 7.4.X1.2 Attributes

**Table 7.4.X1.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| uniformlyDistributedTrainingData | O | T | T | F | T |
| trainingDataWithOrWithoutOutliers | O | T | T | F | T |

#### 7.4.X1.3 Attribute constraints

None.

#### 7.4.X1.4 Notifications

The notifications specified for the IOC using this <<dataType>> for its attribute(s), shall be applicable.

### 7.4.X2 DistributedTrainingExpectation <<dataType>>

#### 7.4.X2.1 Definition

This data type represents the distributed training expectation from ML training MnS consumer.

The attribute dataSplitIndication provides MnS consumers the ability to provide its preferences on splitting the training data. If the data is to be split, the data split mechanism is up to the MnS producer.

The attribute suggestedTrainingNodeList provides the ability for an MnS consumer to provide suggestions on nodes involved in distributed training.

#### 7.4.X2.2 Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
| expectedTrainingTime | O | T | T | F | T |
| dataSplitIndication | O | T | T | F | T |
| suggestedTrainingNodeList | O | T | T | F | T |

#### 7.4.X2.3 Attribute constraints

None

#### 7.4.X2.4 Notifications

The common notifications defined in clause 7.6 are valid for this IOC, without exceptions or additions.

**Next Change**

7.4.X3 PotentialImpactInfo <<dataType>>

7.4.X3.1 Definition

This datatype define the potential network impacts due to the inference output results.

7.4.X3.2 Attributes

The PotentialImpactInfo includes the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| impactedScope | M | T | F | F | T |
| impactedPM | M | T | F | F | T |

7.4.X3.3 Attribute constraints

None.

7.4.X3.4 Notifications

The notifications specified for the IOC using this <<datatype>> for its attribute(s), shall be applicable.

7.4.X4 ImpactedPM <<dataType>>

7.4.X4.1 Definition

This datatype define the potential performance data that may be affected in a non-optimal way due to the recommendations/configurations provided as part of inference output result.

7.4.X4.2 Attributes

The ImpactedPM includes the following attributes:

**Table 7.4.X4.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| pMIdentifier | M | T | F | F | T |

7.4.X4.3 Attribute constraints

None.

#### 7.4.X4.4 Notifications

The notifications specified for the IOC using this <<datatype>> for its attribute(s), shall be applicable.

### 7.4.X5 MLKnowldge <<dataType>>

#### 7.4.X5.1 Definition

The MLKnowldge represents the properties of the ML knowledge, i.e.,information on the experience gained by training of an ML model.

The MLKnowledge is identified by a specific aIMLInferenceName.

The MLKnowledge is contained in a a pair of linked lists PredictorArray and ResponseArray. The nature of the data inside the lists is left to implementation, only the entities that have a prior agreement can exchange and use the lists. To identify available ML Knowldge, the MnS consumer can execute a getMOIattributes operation on the MLKnowldge. The knowledge is implementation specific

#### 7.4.X5.2 Attributes

**Table 7.4.X5.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| mLKnowledgeName | M | T | F | F | F |
| KnowledgeType | M | T | F | F | F |
| PredictorResponseArray | O | T | F | F | T |

#### 7.4.X5.3 Attribute constraints

None

#### 7.4.X5.4 Notifications

The notifications specified for the IOC using this <<dataType>> for its attribute(s), shall be applicable.

|  |
| --- |
| **Next change** |

7.4.X6 EnvironmentScope <<choice>>

7.4.X6.1 Definition

This <<choice>> represents the scope of environment. This represents the information that can be used to select/determine the environment for the Reinforcement learning.

managedEntitiesScope indicates the environment by the DNs of the managed entities.

areaScope indicates the target geographical location of the environment. When defined, the network node(s) serving the specified location forms the RL environment.

timeWindow indicates the timeframe information at which the model is to be trained. It may define a time duration in a day or a time schedule information, i.e. when it is expected for the model to be trained or when it is expected for the trained model to perform its inference.

7.4.X6.2 Attributes

**Table 7.4.X6.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **S** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| Choice\_1.1 managedEntitiesScope | CM | T | T | F | T |
| Choice\_1.2 areaScope | CM | T | T | F | T |
| Choice 2 timeWindow | CM | T | T | F | T |

7.4.X6.3 Attribute constraints

**Table 7.4.X6.3-1**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| Choice\_1.1 managedEntitiesScope | Condition: the MnS producer supports to identify the scope by managed entities. |
| Choice\_1.2 areaScope | Condition: the MnS producer supports to identify the scope by area scope. |
| Choice\_2 timeWindow | Condition: the MnS producer supports to identify the scope by time window. |

7.4.X6.4 Notifications

The notifications specified for the IOC using this <<dataType>> for its attribute(s), shall be applicable.

|  |
| --- |
| **Next change** |

7.4.X7 SupportedLearningTechnology <<dataType>>

7.4.X7.1 Definition

This dataType represents the supported learning technologies of the ML training function for ML model training.

The SupportedLearningTechnology contains the following attributes:

learningTechnologyName indicates learning technologies including the name of Reinforcement learning, Federated learning and distributed training which can be supported by the ML training function. supportedInferenceNameList indicates the type of inference function that the learning technologies can be applied.

7.4.X7.2 Attributes

The SupportedLearningTechnology includes the following attributes:

**Table 7.4.X7.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| learningTechnologyName | M | T | F | F | T |
| supportedEnvironment | CM | T | F | F | T |
| supportedInferenceNameList | O | T | F | F | T |

7.4.X7.3 Attribute constraints

**Table 7.4.X7.3-1**

|  |  |
| --- | --- |
| **Name** | **Definition** |
| supportedEnvironment | Consdition: This attribute shall be supported, when the ML training MnS producer supports Reinforcement learning. |

7.4.X7.4 Notifications

The notifications specified for the IOC using this <<datatype>> for its attribute(s), shall be applicable.

7.4.X8 RLRequirement <<dataType>>

7.4.X8.1 Definition

This dataType represents the ML model training requirement for the RL.

rLEnvironment Type indicates required RL environment type, indicating real-network and simulation network environment where the ML model should be trained.

rLEnvironmentScope indicates RL environment scope, which may be a RL geographical area, network node(s), and time window. The scope does not only include the entities directly involved in RL process, but also includes other entities impacted by RL agent actions. ML training MnS consumer can provide the specific environment scope enabling the producer to select/determine/create the RL environment.

rLPerformanceRequirements indicates the attribute of the network performance requirements for performing online ML training, which indicates the tolerable network performance degradation. When the network performance is within the range, the RL training process can be continued. Otherwise, fall back actions can be determined by the producer. This information can be used to decide the appropriate rewards and state settings. ML training MnS consumer can use this attribute to provide performance requirements of the real operational network during RL training.

These RL related attributes may be used by an ML training MnS producer to perform the ML model training with the training technology of RL.

7.4.X8.2 Attributes

The RLRequirement includes the following attributes:

**Table 7.4.X8.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| rLEnvironmentType | O | T | T | F | T |
| rLEnvironmentScope | O | T | T | F | T |
| rLImpactedScope | O | T | T | F | T |
| rLPerformanceRequirements | O | T | T | F | T |

7.4.X8.3 Attribute constraints

None

7.4.X8.4 Notifications

The notifications specified for the IOC using this <<datatype>> for its attribute(s), shall be applicable.

***Start of second change***

7.4.X9 ClusteringCriteria <<dataType>>

7.4.X9.1 Definition

This dataType represents a set of requirements or criteria related to ML models in order to decide which all ML models can be grouped or clustered together for training.

performanceMetric indicates the criteria based on the performance metric for which the ML model is mainly evaluated. For example, the models, which intend to achieve same performance characteristic, can be clustered together for training. It indicates the performance metric used to evaluate the performance of an ML model.

taskType indicates grouping criteria based on the task the ML model is trained for. This can be aIMLInferenceName or capabilityName defined in 3GPP TS 28.105. That is, which can indicate the type of inference or the capability that the ML model supports like the values of the MDA type (3GPP TS 28.104), Analytics ID(s) of NWDAF (3GPP TS 23.288), types of inference for NG-RAN (TS 38.300 and TS 38.401), and vendor's specific extensions

allowedClusterTrainingTime indicates the combined time limit within which the training of ML models ‘cluster’ shall be completed. A cluster of ML models takes more time to train together as compared to time taken for training an individual ML model. The criteria allows grouping only those ML models for training whose training time does not exceed the set combined time limit.preferredModelDiversity indicates the preferred model diversity types that can be considered for models clustering for training together.

NOTE: It is left upto producer to decide how to consider these criteria for clustering ML models in case of multiple criteria.

7.4.X9.2 Attributes

The ClusteringCriteria includes the following attributes:

**Table 7.4.X9.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| performanceMetric | O | T | T | F | T |
| taskType | M | T | T | F | T |
| allowedClusterTrainingTime | M | T | T | F | T |
| preferredModelDiversity | O | T | T | F | T |

7.4.X9.3 Attribute constraints

None.7.4.X9.4 Notifications

The notifications specified for the IOC using this <<datatype>> for its attribute(s), shall be applicable.

|  |
| --- |
| **Next change** |

7.4.9 InferenceOutput <<dataType>>

7.4.9.1 Definition

This dataType represents the properties of the content of an inference output.

The inference output contains a time stamp which indicates the time at which the inference output is generated. The inference output may include inference explanation information of the ML model during Inference.

7.4.9.2 Attributes

The InferenceOutput includes the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Support Qualifier** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| inferenceOutputId | M | T | F | F | T |
| aIMLInferenceName | M | T | F | F | T |
| inferenceOutputTime | M | T | F | F | T |
| inferencePerformance | O | T | F | F | T |
| inferenceExplanationInfo | O | T | F | F | T |
| outputResult | M | T | F | F | T |

NOTE: The relation between the Output and Outputs of other instances like MDA is not addressed in the present document

7.4.9.3 Attribute constraints

None.

7.4.9.4 Notifications

The notifications specified for the IOC using this <<datatype>> for its attribute(s), shall be applicable.

***Next change***

## 7.5 Attribute definitions

### 7.5.1 Attribute properties

| **Attribute Name** | **Documentation and Allowed Values** | **Properties** |
| --- | --- | --- |
| mLModelId | It identifies the ML model.  It is unique in each MnS producer.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| candidateTrainingDataSource | It provides the address(es) of the candidate training data source provided by MnS consumer. The detailed training data format is vendor specific.  allowedValues: N/A. | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| aIMLInferenceName | It indicates the type of inference that the ML model supports.  allowedValues: see clause 7.4.10 | type: AIMLInferenceName  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTrainingRequest.aIMLInferenceName | It indicates the type of inference that the ML model conducting inference.  allowedValues: see clause 7.4.10 | type: AIMLInferenceName  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| mDAType | It indicates the type of inference that the ML model for MDA supports.  The detailed definition and corresponding allowed values for mDAType see TS 28.104 [2]. | type: MDAType (TS 28.104 [2])  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| nwdafAnalyticsType | It indicates the type of inference that the ML model for NWDAF supports.  The detailed definition and corresponding allowed values for nwdafAnalyticsID see NwdafEvent in TS 29.520 [20]. | type: NwdafEvent (TS 29.520 [20])  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| ngRanInferenceType | It indicates the type of inference that the ML model for NG-RAN supports.  The detailed definition and corresponding allowed values for ngRanInferenceType see clause 7.4a.1 | type: NgRanInferenceType  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| vSExtensionType | It indicates the type of inference that is vendor's specific extension.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| usedConsumerTrainingData | It provides the address(es) where lists of the MnS consumer-provided training data are located, which have been used for the ML model training. It may include the information about the effectiveness of training data, which indicates the MnS consumer-provided training data is useful or not.  allowedValues: N/A. | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| trainingRequestRef | It is the DN(s) of the related MLTrainingRequest MOI(s). | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| trainingProcessRef | It is the DN(s) of the related MLTrainingProcess MOI(s) that produced the MLTrainingReport. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| trainingReportRef | It is the DN of the MLTrainingReport MOI that represents the reports of the ML model training. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| lastTrainingRef | It is the DN of the MLTrainingReport MOI that represents the reports for the last training of the ML model(s). | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| modelConfidenceIndication | It indicates the average confidence value (in unit of percentage) that the ML model would perform for inference on the data with the same distribution as training data.  Essentially, this is a measure of degree of the convergence of the trained ML model.  allowedValues: { 0..100 }. | type: Integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| trainingRequestSource | It identifies the entity that requested to instantiate the MLTrainingRequest MOI.  This attribute is the DN of a managed entity, otherwise, it is a String. | type: <<Choice>>  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTrainingRequest.requestStatus | It describes the status of a particular ML model training request.  allowedValues: NOT\_STARTED, IN\_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLTrainingProcessId | It identifies the training process.  It is unique in each instantiated process in the MnS producer.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| priority | It indicates the priority of the training process.  The priority may be used by the ML model training to schedule the training processes. Lower value indicates a higher priority.  allowedValues: { 0..65535 }. | type: Integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: 0  isNullable: False |
| terminationConditions | It indicates the conditions to be considered by the ML training MnS producer to terminate a specific training process.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| progressStatus | It indicates the status of the process.  allowedValues: N/A. | type: ProcessMonitor  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLUpdateProcess.cancelProcess | It allows the ML update MnS consumer to cancel the ML update process.  Setting this attribute to "TRUE" cancels the ML update process. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLUpdateProcess.suspendProcess | It allows the ML update MnS consumer to suspend the ML update process.  Setting this attribute to "TRUE" suspends the ML update process. The process can be resumed by setting this attribute to “FALSE” when it is suspended. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| mLModelVersion | It indicates the version number of the ML model.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| performanceRequirements | It indicates the expected performance for a trained ML model when performing on the training data.  allowedValues: N/A. | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| modelPerformanceTraining | It indicates the performance score of the ML model when performing on the training data.  allowedValues: N/A. | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| MLTrainingProcess.progressStatus.progressStateInfo | It provides the following specialization for the “progressStateInfo“ attribute of the “ProcessMonitor“ data type for the “MLTrainingProcess.progressStatus“.  When the ML model training is in progress, and the " mLTrainingProcess.progressStatus.status " is equal to "RUNNING", it provides the more detailed progress information.  allowedValues for " mLTrainingProcess.progressStatus.status " = "RUNNING":  - “COLLECTING\_DATA”  - “PREPARING\_TRAINING\_DATA”  - “TRAINING” + DN of the MLModel being trained  The allowed values for " mLTrainingProcess.progressStatus.status " = "CANCELLING" are vendor specific.  The allowed values for " mLTrainingProcess.progressStatus.status " = "NOT\_STARTED" are vendor specific. | type: String  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| inferenceOutputName | It indicates the name of an inference output of an ML model.  allowedValues: the name of the MDA output IEs (see 3GPP TS 28.104 [2]), name of analytics output IEs of NWDAF (see TS 23.288 [3]), RAN inference output IE name(s), and vendor's specific extensions. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| performanceMetric | It indicates the performance metric used to evaluate the performance of an ML model, e.g. "accuracy", "precision", "F1 score", etc.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| performanceScore | It indicates the performance score (in unit of percentage) of an ML model when performing inference on a specific data set (Note).  The performance metrics may be different for different kinds of ML models depending on the nature of the model. For instance, for numeric prediction, the metric may be accuracy; for classification, the metric may be a combination of precision and recall, like the "F1 score".  allowedValues: { 0..100 }. | type: Real  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTrainingRequest.cancelRequest | It allows the ML training MnS consumer to cancel the ML model training request.  Setting this attribute to "TRUE" cancels the ML model training request. The request can be resumed by setting this attribute to "FALSE" when it is suspended. Cancellation is possible when the requestStatus is the "NOT\_STARTED", " IN\_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTrainingRequest.suspendRequest | It allows the ML training MnS consumer to suspend the ML model training request.  Setting this attribute to "TRUE" suspends the ML model training process. Suspension is possible when the requestStatus is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTrainingProcess.cancelProcess | It allows the ML training MnS consumer to cancel the ML model training process.  Setting this attribute to “TRUE“ cancels the ML model training process. Cancellation is possible when the “mLTrainingProcess.progressStatus.status“ is not the “FINISHED“ state. Setting the attribute to “FALSE“ has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTrainingProcess.suspendProcess | It allows the ML training MnS consumer to suspend the ML model training process.  Setting this attribute to "TRUE" suspends the ML model training process. The process can be resumed by setting this attribute to “FALSE” when it is suspended. Suspension is possible when the " mLTrainingProcess.progressStatus.status" is not the "FINISHED", "CANCELLING" or "CANCELLED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| inferenceEntityRef | It describes the target entities that will use the ML model for inference. | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| dataProviderRef | It describes the entities that have provided or should provide data needed by the ML model e.g. for training or inference | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| areNewTrainingDataUsed | It indicates whether new training data are used for the ML model training.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| trainingDataQualityScore | It indicates numerical value that represents the dependability/quality of a given observation and measurement type. The lowest value indicates the lowest level of dependability of the data, i.e. that the data is not usable at all.  allowedValues: { 0..100 }. | type: Real  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| decisionConfidenceScore | It is the numerical value that represents the dependability/quality of a given decision generated by the AI/ML inference function. The lowest value indicates the lowest level of dependability of the decisions, i.e. that the data is not usable at all.  allowedValues: { 0..100 }. | type: Real  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| expectedRuntimeContext | This describes the context where an MLModel is expected to be applied.  allowedValues: N/A | type: MLContext  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| trainingContext | This specifies the context under which the MLModel has been trained.  allowedValues: N/A | type: MLContext  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| runTimeContext | This specifies the context where the MLmodel or model is being applied.  allowedValues: N/A | type: MLContext  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTrainingRequest.mLModelRef | It identifies the DN of the MLModel requested to be trained. | type: DN  multiplicity: 0..1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| MLTrainingReport.mLModelGeneratedRef | It identifies the DN of the MLModel generated by the ML model training. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLModelRepositoryRef | It identifies the DN of the MLModelRepository. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLRepositoryId | It indicates the unique ID of the ML repository. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| modelPerformanceValidation | It indicates the performance score of the ML model when performing on the validation data.  allowedValues: N/A | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| dataRatioTrainingAndValidation | It indicates the ratio (in terms of quantity of data samples) of the training data and validation data used during the training and validation process. It is represented by the percentage of the validation data samples in the total training data set (including both training data samples and validation data samples). The value is an integer reflecting the rounded number of percent \* 100.    allowedValues: { 0 .. 100 }. | type: Integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTestingRequest.requestStatus | It describes the status of a particular ML testing request.  allowedValues: NOT\_STARTED, IN\_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTestingRequest.cancelRequest | It allows the ML testing MnS consumer to cancel the ML testing request.  Setting this attribute to "TRUE" cancels the ML testing request. Cancellation is possible when the requestStatus is the "NOT\_STARTED", " IN\_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTestingRequest.suspendRequest | It allows the ML testing MnS consumer to suspend the ML testing request.  Setting this attribute to "TRUE" suspends the ML testing request. The request can be resumed by setting this attribute to “FALSE” when it is suspended. Suspension is possible when the requestStatus is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTestingRequest.mLModelRef | It identifies the DN of the MLModel requested to be tested. | type: DN  Multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| modelPerformanceTesting | It indicates the performance score of the ML model when performing on the testing data.  allowedValues: N/A. | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLTestingResult | It provides the address where the testing result is provided.  The detailed testing result format is vendor specific.  allowedValues: N/A. | type: String  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| testingRequestRef | It identifies the DN of the MLTestingRequest MOI. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| supportedPerformanceIndicators | This parameter lists specific PerformanceIndicator(s) of an ML model.  allowedValues: N/A. | type: SupportedPerfIndicator  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| performanceIndicatorName | It indicates the identifier of the specific performance indicator.  allowedValues: N/A | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| isSupportedForTraining | It indicates whether the specific performance indicator is supported a performance metric of ML model training for the ML model.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| isSupportedForTesting | It indicates whether the specific performance indicator is supported a performance metric of ML model testing for the ML model.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| mLUpdateProcessRef | It is the DN of the mLUpdateProcess MOI that represents the process of updating an ML model. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLUpdateRequestRefList | It is the list of DN of the MLUpdateRequest MOI that represents an  ML update request. | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLUpdateReportRef | It is the DN of the MLUpdateReport MOI that represents an ML update report. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLUpdateReportingPeriod | It specifies the time duration upon which the MnS consumer expects the ML update is reported. | type: TimeWindow  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| availMLCapabilityReport | It represents the available ML capabilities.  allowedValues: N/A. | type: AvailMLCapabilityReport multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| updatedMLCapability | It represents the updated ML capabilities.  allowedValues: N/A. | type: AvailMLCapabilityReport multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| availMLCapabilityReportID | It identifies the available ML capability report.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| newCapabilityVersionId | It indicates the specific version of AI/ML capabilities to be applied for the update. It is typically the one indicated by the MLCapabilityVersionID in a newCapabilityVersion | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mlCapabilityVersionId | It indicates the version of ML capabilities that is available for the update. | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| performanceGainThreshold | It defines the minimum performance gain as a percentage that shall be achieved with the capability update, i.e., the difference in the performances between the existing capabilities and the new capabilities should be at least performanceGainThreshold otherwise the new capabilities should not be applied.  Allowed value: float between 0.0 and 100.0 | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| expectedPerformanceGains | It indicates the expected performance gain if/when the AI/ML capabilities of the respective network function are updated with/to the specific set of newly available AI/ML capabilities. | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| updateTimeDeadline | It indicates the maximum as stated in the MLUpdate request that should be taken to complete the update | type: TimeWindow  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLUpdateReport.mLModelRefList | It indicates the DN of MLModel instances that can be updated. | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| MLUpdateRequest.requestStatus | It describes the status of a particular ML update request.  allowedValues: NOT\_STARTED, IN\_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLUpdateRequest.cancelRequest | It allows the MnS consumer to cancel the ML update request.  Setting this attribute to "TRUE" cancels the ML update request. Cancellation is possible when the requestStatus is the "NOT\_STARTED", " IN\_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLUpdateRequest.suspendRequest | It allows the MnS consumer to suspend the ML update request.  Setting this attribute to "TRUE" suspends the ML update request. The request can be resumed by setting this attribute to “FALSE” when it is suspended. Suspension is possible when the requestStatus is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| memberMLModelRefList | It identifies the list of member ML models within an ML model coordination group. | type: DN  multiplicity: 2..\*  isOrdered: True  isUnique: True  defaultValue: None  isNullable: False |
| MLTrainingRequest.mLModelCoordinationGroupRef | It identifies the DN of the MLModelCoordinationGroup requested to be trained. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTrainingReport.mLModelCoordinationGroupGeneratedRef | It identifies the DN of the MLModelCoordinationGroup generated by ML model joint training. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTestingRequest.mLModelCoordinationGroupRef | It identifies the DN of the MLModelCoordinationGroup requested to be tested. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| retrainingEventsMonitorRef | It indicates the DN of the ThresholdMonitor MOI that indicates the performance measurements and its corresponding thresholds to be used by MnS producer to initiate the re-training of the MLModel. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLModelLoadingRequest.requestStatus | It describes the status of a particular ML model loading request.  allowedValues: NOT\_STARTED, IN\_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLModelLoadingRequest.cancelRequest | It allows the MnS consumer to cancel the ML model loading request.  Setting this attribute to "TRUE" cancels the ML model loading. Cancellation is possible when the requestStatus is the "NOT\_STARTED", " IN\_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLModelLoadingRequest.suspendRequest | It allows the MnS consumer to suspend the ML model loading request.  Setting this attribute to "TRUE" suspends the ML model loading request. The request can be resumed by setting this attribute to “FALSE” when it is suspended. Suspension is possible when the requestStatus is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| mLModelToLoadRef | It identifies the DN of a trained MLModel requested to be loaded to the target inference function(s). | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| policyForLoading | It provides the policy for controlling ML model loading triggered by the MnS producer.  This policy contains two thresholds in the thresholdList attribute. The first threshold is related to the ML model to be loaded, and the second threshold is related to the existing ML model being used for inference. | type: AIMLManagementPolicy  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| thresholdList | It provides the list of threshold. | type: ThresholdInfo  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| MLModelLoadingProcess.progressStatus.progressStateInfo | It provides the following specialization for the "progressStateInfo" attribute of the "ProcessMonitor" data type for the "MLModelLoadingProcess.progressStatus".  When the ML model loading is in progress, and the " MLModelLoadingProcess.progressStatus.status " is equal to "RUNNING", it provides the more detailed progress information.  allowedValues for " MLModelLoadingProcess.progressStatus.status " = "RUNNING":  The allowed values for " MLModelLoadingProcess.progressStatus.status " = "CANCELLING" are vendor specific.  The allowed values for " MLModelLoadingProcess.progressStatus.status " = "NOT\_STARTED" are vendor specific. | type: String  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLModelLoadingProcess.cancelProcess | It allows the MnS consumer to cancel the ML model loading process.  Setting this attribute to "TRUE" cancels the process. Cancellation is possible when the "MLModelLoadingProcess.progressStatus.status" is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLModelLoadingProcess.suspendProcess | It allows the MnS consumer to suspend the ML model loading process.  Setting this attribute to "TRUE" suspends the process. The process can be resumed by setting this attribute to "FALSE" when it is suspended. Suspension is possible when the "MLModelLoadingProcess.progressStatus.status" is not the "FINISHED", "CANCELLING" or "CANCELLED" state. Setting the attribute to "FALSE" has no observable result.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| mLModelLoadingRequestRef | It identifies the DN of the associated MLModelLoadingRequest. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLModelLoadingPolicyRef | It identifies the DN of the associated MLModelLoadingPolicyRef. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| loadedMLModelRef | It identifies the DN of the MLModel that has been loaded to the inference function. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| activationStatus | It describes the activation status.  allowedValues: ACTIVATED, DEACTIVATED. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| AIMLManagementPolicy.managedActivationScope | It provides a list of sub scopes for which ML inference is activated as triggered by a policy on the MnS producer. For example, the sub scopes may be a list of cells or of geographical areas. The list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope if the policy evaluates to true.  allowedValues: N/A | type: ManagedActivationScope  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| AIMLInferenceFunction.managedActivationScope | It provides a list of sub scopes for which ML inference is activated as triggered by a policy on the MnS producer. For example, the sub scopes may be a list of cells or of geographical areas. The list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope if the policy evaluates to true.  allowedValues: N/A | type: AIMLManagementPolicy  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| ManagedActivationScope.dNList | It indicates the list of DN, the list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope.  allowedValues: N/A | type: DN  multiplicity: \*  isOrdered: True  isUnique: True  defaultValue: None  isNullable: False |
| ManagedActivationScope.timeWindow | It indicates the list of time window; the list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope.  allowedValues: N/A | type: TimeWindow  multiplicity: \*  isOrdered: True  isUnique: True  defaultValue: None  isNullable: False |
| ManagedActivationScope.geoPolygon | It indicates the list of GeoArea, the list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope.  allowedValues: N/A | type: GeoArea  multiplicity: \*  isOrdered: True  isUnique: True  defaultValue: None  isNullable: False |
| usedByFunctionRefList | It provides the DNs of the functions supported by the AIMLInferenceFunction.  allowedValues: N/A | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| inferenceOutputId | It identifies an inference output within an AIMLinferenceReport. | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| inferenceOutputs | It indicates the Outputs that have been derived by the AIMLInferenceFunction instance from a specific ML model.  Each ML model, inferenceOutputs may be a set of values.  allowedValues: N/A. | type: InferenceOutput  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| inferencePerformance | It indicates the performance score of the ML model during Inference.  allowedValues: N/A. | type: ModelPerformance  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| inferenceOutputTime | It indicates the time at which the inference output is generated.  allowedValues: N/A | type: DateTime  multiplicity: \*  isOrdered: True  isUnique: True  defaultValue: None  isNullable: False |
| outputResult | It indicates the result of an inference. | type: AttributeValuePair  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: Null  isNullable: False |
| mLCapabilitiesInfoList | It indicates information about what an ML model can generate inference for.  allowedValues: N/A. | type: MLCapabilityInfo  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| capabilityName | It indicates the name of a capability for which an ML model can generate inference. The capability is defined by Mns producer which can be traffic analysis capability, coverage analysis capability, mobility analysis capability or vendor specific extensions.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLCapabilityParameters | It indicates a set of optional parameters that apply for an aIMLInferenceName capabilityName.  allowedValues: N/A | type: AttributeValuePair  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| aIMLInferenceReportRefList | It indicates a list of DN of the AIMLInferenceReport MOI that represents an AIML inference report. | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| mLModelRefList | It identifies the list of MLModel DN. | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| mLKnowledge | It indicates an instance of ML Knowledge available at the ML training function. | type: MmLKnowledge  multiplicity: \*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| mLKnowledgeName | It identifies the ML Knowledge.  It is unique in each MnS producer. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| KnowledgeType | It identifies the type of ML Knowledge as either a  Statistic, a regression or a Table of input-output value(s)  Allowed values: TABLE , STATISTIC, REGRESSION | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| PredictorResponseArray | It identifies the predictor and corresponding response data for a piece of ML Knowledge. For exapme, it represents one of the following:  - the input and output data for a table  - the predictor and response for a statistic,  - the input and output data for a regression  NOTE: The nature of the data is not scope of this specification | type: pair<String, String>  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| inferenceExplanationInfo | It indicates the inference explanation information of the ML model Inference results. E.g. the critical features in the training or inference data. | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| mLTrainingType | It indicates the type of ML training (e.g., initial-training, re-training, pre-specialised training, fine-tuning) requested by the MnS consumer.  allowed values: initial training, pre-specialised training, re-training, fine-tuning. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| expectedInferenceScope | It indicates the inference capabilities that the ML model is expected to support, where the inference scope contains a list of aIMLInferenceName that the ML model can be potential adapted to support. | type: AIMLInferenceName  multiplicity: \*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| inferenceScope | It indicates the inference capabilities that the ML model after pre-specialized training can be fine-tuned to support, where the inference scope contains a list of aIMLInferenceName that the ML model can be potentially adapted to support. | type: AIMLInferenceName  multiplicity: \*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| distributedTrainingExpectation | It indicates distributed traning expectations provided by MnS consumer.  allowedValues: N/A. | type: DistributedTrainingExpectation  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| expectedTrainingTime | It indicates the expected training duration provided by MnS consumer, in unit of minites.  allowedValues: Integer | type: Integer  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| dataSplitIndication | This is a Boolean attribute specifying whether the provided training data should be split or not. The value FALSE specify that the training data shall not be spilt.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: False  isNullable: False | |
| suggestedTrainingNodeList | It indicates a list of suggested training nodes provided by MnS consumer.  allowedValues: Not applicable. | type: DN  multiplicity: 0..\*  isOrdered: N/A  isUnique: True  defaultValue: None  isNullable: False | |
| trainingDataStatisticalProperties | It indicates the training data statistical properties to be considered by the MnS producer when training an ML model. | type: DataStatisticalProperties  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| uniformlyDistributedTrainingData | It indicates the need for using training data that are uniformly distributed according to the different aspects (e.g., equivalent data samples for each UE in the training data, equivalent data samples for each type of slice in the training data, equivalent data samples from each GeoArea in the training data) of the aIMLinferenceName.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False | |
| trainingDataWithOrWithoutOutliers | It indicates that the training data samples should consider or disregard data samples that are at the extreme boundaries of the value range.  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False | |
| potentialImpactInfo | This datatype define the potential network impacts due to the inference output results | type: PotentialImpactInfo  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| impactedScope | This will specify the scope of affect, the inference output may have on the network including entities performing the recommended actions in the inference output and entities impacted due to implementation of the recommended actions  The choice attribuite dNList defines Identifier of the network functions that may be affected by the output result of the inference function.  The choice attribute timeWindow defines a time duration indicating that the related network function(s) may be affected during this time duration by the inference output result.  The choice attribute geoPolygon defines a Geographical location indicating that the network function(s) in that location may be affected by the inference output result. | type: ManagedActivationScope  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| impactedPM | This will identify the potential performance metrics that may be degraded/improved due to the implementation of recommendations provided as part of inference output. | type: ImpactedPM  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| pMIdentifier | This indicates the performance measurement or the KPI that may be impacted by the ML model. This will be the name of PM and KPI as defined in 3GPP TS 28.552 and 28.554 respectively (e.g. for Managing NG-RAN AI/ML-based distributed Load Balancing function, the PM can be measurements related to MLB, UE throughput and Radio resource utilization etc). | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| supportedLearningTechnology | It identifies the learning technologies including Reinforcement learning, Federated learning and Distributed training which supported by the ML training function. | type: SupportedLearningTechnology  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| rLRequirement | It identifies the expected performanc and performed scope for the ML model training when Reinforcement learning is supported. | type: RLRequirement  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| learningTechnologyName | It indicates a list of learning technology names used to represent the learning technics supported by the ML training function.  allowedValues: RL, FL, DL  where RL indicates Reinforcement learning, FL indicates Federated learning and DL indicates of Distributed training. | type: Enum  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| supportedEnvironment | It indicates the supported RL environments. When the ML training MnS producer supports RL, this attribute is included in the SupportedLearningTechnology datatype, which indicates the supported environment of the ML training function for ML model training.  allowedValues: SIMULATION ENVIONMENTS, REAL NETWORK ENVIONMENTS. | type: Enum  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| supportedInferenceNameList | It indicates a list of inference name that the learning technologies can be applied.  allowedValues: see clause 7.4.10 | type: AIMLInferenceName  multiplicity: 1..\*  isOrdered: False  isUnique: N/A  defaultValue: None  isNullable: False | |
| rLEnvironmentType | It indicates the simulated environment or real network where the ML model should be traind.  allowedValues: SIMULATION ENVIONMENTS, REAL NETWORK ENVIONMENTS | type: Enum  multiplicity: 0..\*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| rLEnvironmentScope | It indicates the specific environment scope for the entities that the RL process should be performed, i.e, where the RL agent is located. | type: EnvironmentScope  multiplicity: 1..\*  isOrdered: False  isUnique: N/A  defaultValue: None  isNullable: False | |
| rLImpactedScope | It indicates the specific environment scope for the entities that may be impacted by the RL process, i.e., scope may be impacted by actions of the RL agent. | type: EnvironmentScope  multiplicity: 1..\*  isOrdered: False  isUnique: N/A  defaultValue: None  isNullable: False | |
| rLPerformanceRequirements | It indicates a list of thresholds for the network performance requirements, when the RL training process(es) is performed. | type: ThresholdInfo  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| clusteringInfo | It containes information that indicates the clustering criteria for the ML models that can be grouped together for training | type: ClusteringCriteria  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False | |
| ClusteringCriteria.performanceMetric | This defines clustering criteria based on the performance metric for which the ML model is mainly evaluated. That is, the models, which intend to achieve same performance characteristic (e.g. accuracy, precision, F1 score etc) can be clustered together for training. It indicates the performance metric used to evaluate the performance of an ML model  allowedValues: N/A | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| taskType | This defines grouping criteria based on the task the ML model is trained for. For example, this can be aIMLInferenceName or capabilityName as defined in 3GPP TS 28.105.  Note: Whether the taskType can be aIMLInferenceName here is FFS. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| allowedClusterTrainingTime | This defines the combined time limit within which the training of ML models cluster shall be completed. A cluster of ML models takes more time to train together as compared to time taken for training an individual ML model. The criteria allows accommodating only those ML models whose training time does not exceed the set combined time limit | type: TimeWindow  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: True | |
| preferredModelDiversity | This defines the MnS consumer preferred model diversity types that is to be considered for models clustering. For example, decision trees, neural networks, linear regression and like so | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |
| NOTE: When the performanceScore is to indicate the performance score for ML model training, the data set is the training data set. When the performanceScore is to indicate the performance score for ML validation, the data set is the validation data set. When the performanceScore is to indicate the performance score for ML model testing, the data set is the testing data set. | | |

***Next change***

9 Solution Set (SS)

9.X1 OpenAPI document for provisioning MnS

The OpenAPI/YAML definitions for provisioning MnS are specified in 3GPP Forge, refer to clause 4.3 (OpenAPI Definitions) of TS 28.623 [19] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag\_Rel19\_SA105/".

Directory: OpenAPI

File: TS28532\_ProvMnS.yaml

9.X2 OpenAPI document for AI/ML management

The present document defines the following NRM Solution Set definitions for AI/ML management:

The OpenAPI/YAML definitions are specified in 3GPP Forge, refer to clause 4.3 of TS 28.623 [19] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag\_Rel19\_SA105/".

Directory: OpenAPI

File: TS28105\_AiMlNrm.yaml

***Next change***

X Stage 3 definition for AI/ML Management

X.1 RESTful HTTP-based solution set

The RESTful HTTP-based solution set for generic provisioning management service is defined in clause 12.1.1 in 3GPP TS 28.532 [11]. Corresponding className is ML model training, ML model testing, AI/ML inference emulation, ML model deployment, and AI/ML inference.

X.1.1 ML model training

Table X.1.1-1describes the SS to support ML model training request management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.1-1: SS to support ML model training request management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model training request management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Create an ML model training request | createMOI operation | PUT | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTrainingRequest}={id} |
| Modify an ML model training request | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTrainingRequest}={id} |
| Create, Delete, and Update ML training request | changeMOIs operation | PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTrainingRequest}={id} |

Table X.1.1-2 describes the SS to support ML model training report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.1-2: SS to support ML model training report management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model training report management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Query an ML model training report | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTrainingReport}={id} |

Table X.1.1-3 describes the SS to support ML model training process based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.1-3: SS to support ML model training process management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model training process management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Modify an ML model training process | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTrainingProcess}={id} |
| Query an ML model training process | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTrainingProcess}={id} |

X.1.2 ML model testing

Table X.1.2-1 describes the SS to support ML model testing management based on Table 12.1.1.1.1-1in TS 28.532 [11].

**Table X.1.2-1: SS to support ML model testing management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model testing request management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Create an ML model testing request | createMOI operation | PUT | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingRequest}={id} |
| Modify an ML model testing request | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingRequest}={id} |
| Create, Delete, and Update ML testing request | changeMOIs operation | PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingRequest}={id} |

Table X.1.2-2 describes the SS to support ML model testing report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.2-2: SS to support ML model testing report management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model testing report management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Query an ML model testing report | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingreport}={id} |
| Subscribe an ML model testing report | createMOI operation | PUT | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingreport}={id} |
| Unsubscribe an ML model testing report | deleteMOI operation | DELETE | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingreport}={id} |
| Query an ML model testing report subscription | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLTestingreport}={id} |

X.1.3 AI/ML inference emulation

Table X.1.3-1 describes the SS to support AI/ML inference emulation report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.3-1: SS to support AI/ML inference emulation report management**

|  |  |  |  |
| --- | --- | --- | --- |
| **AI/ML inference emulation report management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Query an AI/ML inference emulation report | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{AIMLInferenceReport}={id} |

X.1.4 ML model deployment

Table X.1.4-1 describes the SS to support ML model loading request management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.4-1: SS to support ML model loading request management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model loading management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Create an ML model loading request | createMOI operation | PUT | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingRequest}={id} |
| Modify an ML model loading request | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingRequest}={id} |
| Create, Delete, and Update ML model Loading request | changeMOIs operation | PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingRequest}={id} |

Table X.1.4-2 describes the SS to support ML model loading policy based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.4-2: SS to support ML model loading policy management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model loading policy management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Create an ML model loading policy | createMOI operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingPolicy}={id} |
| Delete an ML model loading policy | deleteMOI operation | DELETE | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingPolicy}={id} |
| Modify an ML model loading policy | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingPolicy}={id} |
| Query an ML model loading policy | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingPolicy}={id} |
| Create, Delete, and Update ML model Loading policy | changeMOIs operation | PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingPolicy}={id} |

Table X.1.4-3 describes the SS to support ML model loading process management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.4-3: SS to support ML model loading process management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model loading process management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Modify an ML model loading process | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingProcess}={id} |
| Query an ML model loading process | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLModelLoadingProcess}={id} |

X.1.5 AI/ML inference

Table X.1.5-1 describes the SS to support ML model update request management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.5-1: SS to support ML model update request management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model update request management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Create an ML model update request | createMOI operation | PUT | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLUpdateRequest}={id} |
| Modify an ML model update request | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLUpdateRequest}={id} |
| Create, Delete, and Update ML model Update request | changeMOIs operation | PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLUpdateRequest}={id} |

Table X.1.5-2 describes the SS to support ML model update report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.5-2: SS to support ML model update report management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model update report management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Query an ML model update report | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLUpdateReport}={id} |

Table X.1.5-3 describes the SS to support ML model update process management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.5-3: SS to support ML model update process management**

|  |  |  |  |
| --- | --- | --- | --- |
| **ML model update process management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Modify an ML model update process | modifyMOIAttributes operation | PUT  PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLUpdateProcess}={id} |
| Query an ML model update process | getMOIAttributes operation | PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{MLUpdateProcess}={id} |

Table X.1.5-4 describes the SS to support AI/ML infernece report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

**Table X.1.5-4: SS to support AI/ML infernece report management**

|  |  |  |  |
| --- | --- | --- | --- |
| **AI/ML inference report management** | **IS operation** | **HTTP Method** | **Resource URI** |
| Query an AI/ML inference report | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{AIMLInferenceReport}={id} |

**Next change**

# Annex A (informative): PlantUML source code for NRM class diagrams

## A.1 General

This annex contains the PlantUML source code for the NRM diagrams defined in clause 7.2a of the present document.

## A.12 PlantUML code for Figure 7.3a.4.1.1-2: NRM fragment for AI/ML inference function

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

skinparam nodesep 60

class AIMLInferenceFunction <<InformationObjectClass>>

class AIMLInferenceReport <<InformationObjectClass>>

class MLModel <<InformationObjectClass>>

class ManagedEntity <<ProxyClass>>

class AIMLSupportedFunction <<ProxyClass>>

ManagedEntity "1" \*-- "\*" AIMLInferenceFunction : <<names>>

AIMLInferenceFunction "\*" <-l-> "\*" AIMLSupportedFunction

MLModel "\*" <-r-> "\*" AIMLSupportedFunction

MLModel "\*" <-r-> "\*" AIMLInferenceFunction

MLModel "1..\*" <-r-> "\*" AIMLInferenceReport

AIMLInferenceFunction "1" \*-- "\*" AIMLInferenceReport : <<names>>

note right of ManagedEntity #white

Represents the IOCs:

ManagedElement or

SubNetwork or

ManagedFunction

end note

note top of AIMLSupportedFunction #white

Represents the IOCs:

DMROFunction or

DLBOFunction or

DESManagementFunction or

MDAFunction or

AnLFFunction or

LMFFunction

end note

@enduml

***End of changes***