|  |  |
| --- | --- |
| 3GPP TS 28.105 V18.3.0 (2024-03) | |
| Technical Specification | |
| 3rd Generation Partnership Project;  Technical Specification Group Services and System Aspects;  Management and orchestration;  Artificial Intelligence / Machine Learning (AI/ML) management (Release 18) | |
|  | |
|  | 3GPP-logo_web |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
|  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 9

1 Scope 11

2 References 11

3 Definitions of terms, symbols and abbreviations 12

3.1 Terms 12

3.2 Symbols 12

3.3 Abbreviations 13

4 Concepts and overview 13

4.1 Overview 13

4a AI/ML management functionality and service framework 14

4a.0 AI/ML operational workflow 14

4a.1 Functionality and service framework for ML training 15

4a.2 AI/ML functionalities management scenarios 15

5 Void 17

6 AI/ML management use cases and requirements 17

6.1 General 17

6.2 Void 18

6.2a ML training phase 18

6.2a.1 ML training 18

6.2a.1.1 Description 18

6.2a.1.2 Use cases 18

6.2a.1.2.1 ML training requested by consumer 18

6.2a.1.2.2 ML training initiated by producer 19

6.2a.1.2.3 ML entity selection 19

6.2a.1.2.4 Managing ML training processes 19

6.2a.1.2.5 Handling errors in data and ML decisions 20

6.2a.1.2.6 ML entity joint training 20

6.2a.1.2.7 ML entity validation performance reporting 21

6.2a.1.2.8 Training data effectiveness reporting 21

6.2a.1.3 Requirements for ML training 21

6.2a.2 Performance management for ML training and testing 23

6.2a.2.1 Description 23

6.2a.2.2 Use cases 23

6.2a.2.2.1 Performance indicator selection for ML training and testing 23

6.2a.2.2.2 ML entity performance indicators query and selection for ML training and testing 24

6.2a.2.2.3 MnS consumer policy-based selection of ML entity performance indicators for ML training and testing 24

6.2a.2.3 Requirements for ML training and testing performance management 25

6.2a.3 ML testing 25

6.2a.3.1 Description 25

6.2a.3.2 Use cases 25

6.2a.3.2.1 Consumer-requested ML entity testing 25

6.2a.3.2.2 Producer-initiated ML entity testing 26

6.2a.3.2.3 Joint testing of multiple ML entities 26

6.2a.3.3 Requirements for ML testing 26

6.3 AI/ML emulation phase 26

6.3.1 Description 26

6.3.2 Use cases 26

6.3.2.1 AI/ML Inference emulation 26

6.3.3 Requirements for Managing AI/ML Inference emulation 27

6.4 ML entity deployment phase 27

6.4.1 ML entity loading 27

6.4.1.1 Description 27

6.4.1.2 Use cases 27

6.4.1.2.1 Consumer requested ML entity loading 27

6.4.1.2.2 Control of producer-initiated ML entity loading 27

6.4.1.2.3 ML entity registration 28

6.4.1.3 Requirements for ML entity loading 28

6.5 AI/ML inference phase 28

6.5.1 AI/ML inference performance management 28

6.5.1.1 Description 28

6.5.1.2 Use cases 29

6.5.1.2.1 AI/ML inference performance evaluation 29

6.5.1.2.2 AI/ML performance measurements selection based on MnS consumer policy 29

6.5.1.3 Requirements for AI/ML inference performance management 29

6.5.2 AI/ML update control 30

6.5.2.1 Description 30

6.5.2.2 Use cases 30

6.5.2.2.1 Availability of new capabilities or ML entities 30

6.5.2.2.2 Triggering ML entity update 30

6.5.2.3 Requirements for AIML update control 31

6.5.3 AI/ML inference capabilities management 31

6.5.3.1 Description 31

6.5.3.2 Use cases 32

6.5.3.2.1 Identifying capabilities of ML entities 32

6.5.3.2.2 Mapping of the capabilities of ML entities 32

6.5.3.3 Requirements for AI/ML inference capabilities management 32

6.5.4 AI/ML inference capability configuration management 33

6.5.4.1 Description 33

6.5.4.2 Use cases 33

6.5.4.2.1 Managing NG-RAN AI/ML-based distributed Network Energy Saving 33

6.5.4.2.2 Managing NG-RAN AI/ML-based distributed Mobility Optimization 33

6.5.4.2.3 Managing NG-RAN AI/ML-based distributed Load Balancing 33

6.5.4.3 Requirements for AI/ML inference management 34

6.5.5 Executing AI/ML Inference 34

6.5.5.1 Description 34

6.5.5.2 Use cases 34

6.5.5.2.1 AI/ML Inference History - tracking inferences and context 34

6.5.5.3 Requirements for Executing AI/ML Inference 35

7 Information model definitions for AI/ML management 36

7.1 Imported and associated information entities 36

7.1.1 Imported information entities and local labels 36

7.1.2 Associated information entities and local labels 36

7.2 Void 36

7.2a Common information model definitions for AI/ML management 37

7.2a.1 Class diagram 37

7.2a.1.1 Relationships 37

7.2a.1.2 Inheritance 37

7.2a.2 Class definitions 37

7.2a.2.1 MLEntity 37

7.2a.2.1.1 Definition 37

7.2a.2.1.2 Attributes 38

7.2a.2.1.3 Attribute constraints 38

7.2a.2.1.4 Notifications 38

7.2a.2.2 MLEntityRepository 38

7.2a.2.2.1 Definition 38

7.2a.2.2.2 Attributes 38

7.2a.2.2.3 Attribute constraints 39

7.2a.2.2.4 Notifications 39

7.2a.2.3 MLEntityCoordinationGroup 39

7.2a.2.3.1 Definition 39

7.2a.2.3.2 Attributes 39

7.2a.2.3.3 Attribute constraints 39

7.2a.2.3.4 Notifications 39

7.3 Void 39

7.3a Information model definitions for AI/ML operational phases 39

7.3a.1 Information model definitions for ML Training 39

7.3a.1.1 Class diagram 39

7.3a.1.1.1 Relationships 39

7.3a.1.1.2 Inheritance 41

7.3a.1.2 Class definitions 41

7.3a.1.2.1 MLTrainingFunction 41

7.3a.1.2.1.1 Definition 41

7.3a.1.2.1.2 Attributes 41

7.3a.1.2.1.3 Attribute constraints 41

7.3a.1.2.1.4 Notifications 41

7.3a.1.2.2 MLTrainingRequest 42

7.3a.1.2.2.1 Definition 42

7.3a.1.2.2.2 Attributes 43

7.3a.1.2.2.3 Attribute constraints 43

7.3a.1.2.2.4 Notifications 43

7.3a.1.2.3 MLTrainingReport 43

7.3a.1.2.3.1 Definition 43

7.3.1.2.3.2 Attributes 43

7.3a.1.2.3.3 Attribute constraints 44

7.3a.1.2.3.4 Notifications 44

7.3a.1.2.4 MLTrainingProcess 44

7.3a.1.2.4.1 Definition 44

7.3a.1.2.4.2 Attributes 45

7.3a.1.2.4.3 Attribute constraints 45

7.3a.1.2.4.4 Notifications 45

7.3a.1.2.5 MLTestingFunction 45

7.3a.1.2.5.1 Definition 45

7.3a.1.2.5.2 Attributes 46

7.3a.1.2.5.3 Attribute constraints 46

7.3a.1.2.5.4 Notifications 46

7.3a.1.2.6 MLTestingRequest 46

7.3a.1.2.6.1 Definition 46

7.3a.1.2.6.2 Attributes 46

7.3a.1.2.6.3 Attribute constraints 47

7.3a.1.2.6.4 Notifications 47

7.3a.1.2.7 MLTestingReport 47

7.3a.1.2.7.1 Definition 47

7.3a.1.2.7.2 Attributes 47

7.3a.1.2.7.3 Attribute constraints 47

7.3a.1.2.7.4 Notifications 47

7.3a.2 Information model definitions for ML emulation Phase 48

7.3a.2.1 Class diagram 48

7.3a.2.1.1 Relationships 48

7.3a.2.1.2 Inheritance 48

7.3a.2.2 Class definitions 48

7.3a.2.2.1 AIMLInferenceEmulationFunction 48

7.3a.2.2.1.1 Definition 48

7.3a.2.2.1.2 Attributes 49

7.3a.2.2.1.3 Attribute constraints 49

7.3a.2.2.1.4 Notifications 49

7.3a.3 Information model definitions for ML deployment phase 49

7.3a.3.1 Class diagram 49

7.3a.3.1.1 Relationships 49

7.3a.3.1.2 Inheritance 50

7.3a.3.2 Class definitions 50

7.3a.3.2.1 MLEntityLoadingRequest 50

7.3a.3.2.1.1 Definition 50

7.3a.3.2.1.2 Attributes 50

7.3a.3.2.1.3 Attribute constraints 50

7.3a.3.2.1.4 Notifications 50

7.3a.3.2.2 MLEntityLoadingPolicy 50

7.3a.3.2.2.1 Definition 50

7.3a.3.2.2.2 Attributes 51

7.3a.3.2.2.3 Attribute constraints 51

7.3a.3.2.2.4 Notifications 51

7.3a.3.2.3 MLEntityLoadingProcess 51

7.3a.3.2.3.1 Definition 51

7.3a.3.2.3.2 Attributes 52

7.3a.3.2.3.3 Attribute constraints 52

7.3a.3.2.3.4 Notifications 52

7.3a.4 Information model definitions for ML inference phase 52

7.3a.4.1 Class diagram 52

7.3a.4.1.1 Relationships 52

7.3a.4.1.2 Inheritance 53

7.3a.4.2 Class definitions 53

7.3a.4.2.1 MLUpdateFunction 53

7.3a.4.2.1.1 Definition 53

7.3a.4.2.1.2 Attributes 54

7.3a.4.2.1.3 Attribute constraints 54

7.3a.4.2.1.4 Notifications 54

7.3a.4.2.2 MLUpdateRequest 54

7.3a.4.2.2.1 Definition 54

7.3a.4.2.2.2 Attributes 55

7.3a.4.2.2.3 Attribute constraints 55

7.3a.4.2.2.4 Notifications 55

7.3a.4.2.3 MLUpdateProcess 55

7.3a.4.2.3.1 Definition 55

7.3a.4.2.3.2 Attributes 56

7.3a.4.2.3.3 Attribute constraints 56

7.3a.4.2.3.4 Notifications 56

7.3a.4.2.4 MLUpdateReport 56

7.3a.4.2.4.1 Definition 56

7.3a.4.2.4.2 Attributes 56

7.3a.4.2.4.3 Attribute constraints 57

7.3a.4.2.4.4 Notifications 57

7.3a.4.2.5 AIMLInferenceFunction 57

7.3a.4.2.5.1 Definition 57

7.3a.4.2.5.2 Attributes 57

7.3a.4.2.5.3 Attribute constraints 57

7.3a.4.2.5.4 Notifications 57

7.3a.4.2.6 AIMLInferenceReport 58

7.3a.4.2.6.1 Definition 58

7.3a.4.2.6.2 Attributes 58

7.3a.4.2.6.3 Attribute constraints 58

7.3a.4.2.6.4 Notifications 58

7.4 Data type definitions 58

7.4.1 ModelPerformance <<dataType>> 58

7.4.1.1 Definition 58

7.4.1.2 Attributes 58

7.4.1.3 Attribute constraints 59

7.4.1.4 Notifications 59

7.4.2 Void 59

7.4.3 MLContext <<dataType>> 59

7.4.3.1 Definition 59

7.4.3.2 Attributes 59

7.4.3.3 Attribute constraints 59

7.4.3.4 Notifications 59

7.4.4 SupportedPerfIndicator <<dataType>> 59

7.4.4.1 Definition 59

7.4.4.2 Attributes 60

7.4.4.3 Attribute constraints 60

7.4.4.4 Notifications 60

7.4.5 AvailMLCapabilityReport <<dataType>> 60

7.4.5.1 Definition 60

7.4.5.2 Attributes 60

7.4.5.3 Attribute constraints 61

7.4.5.4 Notifications 61

7.4.6 AIMLManagementPolicy <<dataType>> 61

7.4.6.1 Definition 61

7.4.6.2 Attributes 61

7.4.6.3 Attribute constraints 61

7.4.6.4 Notifications 61

7.4.7 ManagedActivationScope <<choice>> 61

7.4.7.1 Definition 61

7.4.7.2 Attributes 62

7.4.7.3 Attribute constraints 62

7.4.7.4 Notifications 62

7.4.8. MLCapabilityInfo <<dataType>> 62

7.4.8.1. Definition 62

7.4.8.2 Attributes 62

7.4.8.3 Attribute constraints 62

7.4.8.4 Notifications 62

7.4.9 InferenceOutput <<dataType>> 62

7.4.9.1 Definition 62

7.4.9.2 Attributes 63

7.4.9.3 Attribute constraints 63

7.4.9.4 Notifications 63

7.5 Attribute definitions 63

7.5.1 Attribute properties 63

7.5.2 Constraints 74

7.6 Common notifications 74

7.6.1 Configuration notifications 74

8 Service components 75

8.1 Service components for ML model training MnS 75

9 Solution Set (SS) 75

Annex A (informative): PlantUML source code for NRM class diagrams 76

A.1 General 76

A.2 PlantUML code for Figure 7.3a.1.1.1-1: NRM fragment for ML model training 76

A.3 PlantUML code for Figure 7.3a.1.1.2-1: Inheritance Hierarchy for ML model training related NRMs 77

A.4 PlantUML code for Figure 7.2a.1.2-1: Inheritance Hierarchy for common information models for AI/ML management 78

A.5 PlantUML code for Figure 7.2a.1.1-1: Relationships for common information models for AI/ML management 78

A.6 PlantUML code for Figure 7.3a.1.1.1-2: NRM fragment for ML entity testing 78

A.7 PlantUML code for Figure 7.3a.1.1.2-2: Inheritance Hierarchy for ML entity testing related NRMs 79

A.8 PlantUML code for Figure 7.3a.4.1.1-1: NRM fragment for ML update 79

A.9 PlantUML code for Figure 7.3a.4.1.2-1: Inheritance Hierarchy for ML update related NRMs 80

A.10 PlantUML code for Figure 7.3a.3.1.1-1: NRM fragment for ML entity loading 80

A.11 PlantUML code for Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML entity loading related NRMs 81

A.12 PlantUML code for Figure 7.3a.4.1.1-2: NRM fragment for AI/ML inference function 81

A.13 PlantUML code for Figure 7.3a.4.1.2-2: Inheritance Hierarchy for AI/ML inference function 82

A.14 PlantUML code for Figure 7.3a.2.1.1-1: NRM fragment for AI/ML inference emulation Control 82

A.15 PlantUML code for Figure 7.3a.2.1.2-1: AI/ML inference emulation Inheritance Relations 83

Annex B (normative): OpenAPI definition of the AI/ML NRM 84

B.1 General 84

B.2 Solution Set (SS) definitions 84

B.2.1 OpenAPI document "TS28105\_AiMlNrm.yaml" 84

Annex C (informative): Change history 95

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

# The present document specifies the Artificial Intelligence / Machine Learning (AI/ML) management capabilities and services for 5GS where AI/ML is used, 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]) and NG-RAN (see TS 38.300 [16] and TS 38.401 [17]).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".

# 3 Definitions of terms, symbols and abbreviations

## 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 entity:** a manageable artifact of an ML model.

NOTE 1: An ML entity may contain metadata related to the model. Metadata may include e.g. the applicable runtime context for the ML model.

**ML model:** mathematical algorithm that can be "trained" by data and human expert input as examples to replicate a decision an expert would make when provided that same information.

NOTE 2: The ML models are proprietary and not in scope for standardization.

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

**ML initial training:** the ML model training that generates the initial version of an ML entity.

**ML re-training:** The process of training of a previously trained ML model.

NOTE 3: A new version of a trained ML entity supports the same type of inference as the previous version of the ML entity, i.e., the data type of inference input and data type of inference output remain unchanged between the two versions of the ML entity, but parameter values might be different for the re-trained model.

**ML joint training:** the ML training for a group of ML models that are trained and targeted for inference.

**ML training:** refers to the end-to-end processes to enable an ML training function to perform ML model initial training or re-training (as defined above).

NOTE 4: ML training may include interaction with other parties to collect and format the data required for ML model training.

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

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

**AI/ML inference function**: a logical function that employs an ML model to conduct inference.

## 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

# 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 means the input and output parameters are explicitly labelled for each training data example. | | | | |

**- 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 AI/ML management related capabilities and services, which include the followings:

- ML training.

# 4a AI/ML management functionality and service framework

## 4a.0 AI/ML operational workflow

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



Figure 4a.0-1: AI/ML operational workflow

The workflow involves 4 main operational phases; namely training, emulation, deployment, and inference phase. The main tasks for each phase are briefly described below:

**Training phase:**

**- ML model training:** training, including initial training and re-training, of an ML model or a group of ML models. It also includes validation of the ML entity to evaluate the performance when the ML entity performs on the training data and validation data. If the validation result does not meet the expectation (e.g., the variance is not acceptable), the ML model associated with that entity needs to be re-trained. The ML model training is the initial phase of the workflow.

**- ML testing:** testing of the validated ML entity to evaluate the performance of the trained ML model when it performs on testing data. If the testing result meets the expectation, the ML entity may proceed to the next phase, otherwise the ML model associated with that entity may need to be re-trained.

**Emulation phase:**

**- ML emulation:** running an ML entity for inference in an emulation environment. The purpose is to evaluate the inference performance of the ML entity in the emulation environment prior to applying it to the target network or system.

NOTE: The emulation phase is considered optional and can be skipped in the AI/ML operational workflow.

**Deployment phase:**

**- ML entity loading:** the process (a.k.a. a sequence of atomic actions) of making a trained ML entity available for use at the target AI/ML inference function.

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

**Inference phase:**

**- AI/ML inference:** performing inference using a trained ML entity by the AI/ML inference function.

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

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

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



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

The internal business logic of ML 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 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 28.532 [11].

- 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.

## 4a.2 AI/ML functionalities management scenarios

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 of MDAF. The AI/ML inference function is in the MDAF.

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

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.

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.

A diagram of a network

Description automatically generated

Figure 4a.2-1: Management for RAN domain analytics

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

**Scenario 2:**

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 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 training and AI/ML inference are both located in gNB

# 5 Void

# 6 AI/ML management use cases and requirements

## 6.1 General

Each operational step in the workflow (see clause 5.0) is supported by one or more AI/ML management capabilities as depicted below for each of the operational phases.

**Management capabilities for ML training**

**- ML training management**: allowing the MnS consumer to request the ML training, consume and control the producer-initiated training, and manage the ML training/re-training process. The training management capability may include training performance management and setting a policy for the producer-initiated ML training.

**- ML validation**: ML training capability also includes validation to evaluate the performance of the ML entity when performing on the validation data, and to identify the variance of the performance on the training and validation data. If the variance is not acceptable, the ML entity would need to be tuned (re-trained) before being made available for the next step in the operational workflow (e.g., ML entity testing).

**- ML testing management**: allowing the MnS consumer to request the ML entity testing, and to receive the testing results for a trained ML entity. It may also include capabilities for selecting the specific performance metrics to be used or reported by the ML testing function. MnS consumer may also be allowed to trigger ML re-training based on the ML entity testing performance requirements.

**Management capabilities for ML emulation phase:**

* **AI/ML inference emulation:** a capability allowing an MnS consumer to request an ML inference emulation for a specific ML entity or entities (after the training, validation, and testing) to evaluate the inference performance in an emulation environment prior to applying it to the target network or system.

**Management capabilities for ML entity deployment phase:**

**- ML entity loading management**: allowing the MnS consumer to trigger, control and/or monitor the ML entity loading process.

**Management capabilities for AI/ML inference phase:**

**- AI/ML inference management:** allowing an MnS consumer to control the inference, i.e., activate/deactivate the inference function and/or ML entity/entities, configure the allowed ranges of the inference output parameters. The capabilities also allow the MnS consumer to monitor and evaluate the inference performance and when needed trigger an update of an ML entity or an AI/ML inference function.

The use cases and corresponding requirements for AI/ML management capabilities are specified in the following clauses for each phase of the operational workflow.



## 6.2 Void6.2a ML training phase

### 6.2a.1 ML training

#### 6.2a.1.1 Description

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

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).

#### 6.2a.1.2 Use cases

##### 6.2a.1.2.1 ML training requested by consumer

The ML training capabilities are provided by an ML training MnS producer to one or more consumer(s).

A black background with green lines

Description automatically generated

Figure 6.2a.1.2.1-1: ML training requested by ML training MnS consumer

The ML training may be triggered by the request(s) from one or more ML training MnS consumer(s). The consumer may be for example a network function, a management function, an operator, or another functional differentiation. Figure 6.2a.1.2.1-1 highlights the high-level overview of the process and the relevant sequence.

To trigger an initial ML training, the MnS consumer needs to specify in the ML training request the inference type which indicates the function or purpose of the ML entity, e.g. CoverageProblemAnalysis [see TS 28.104 [2]]. The ML training MnS producer can perform the initial training according to the designated inference type. To trigger an ML re-training, the MnS consumer needs to specify in the ML training request the identifier of the ML entity to be re-trained.

The consumer may provide the data source(s) that contain(s) the training data which are considered as inputs candidates for training. To obtain the valid training outcomes, consumers may also designate their requirements for model performance (e.g. accuracy, etc) in the training request.

The performance of the ML entity depends on the degree of commonality between the distribution of the data used for training and the distribution of the data used for inference. As time progresses, the distribution of the input data used for inference might change as compared to the distribution of the data used for training. In such a scenario, the performance of the ML entity degrades over time. The ML training MnS producer may re-train the ML model associated to the entity if the inference performance of the ML entity falls below a certain threshold, which needs to be configurable by the MnS consumer.

Following the ML training request by the M training MnS consumer, the ML training MnS producer provides a response to the consumer indicating whether the request was accepted.

If the request is accepted, the ML training MnS producer decides when to start the ML training with consideration of the request(s) from the consumer(s). Once the training is decided, the producer performs the following:

- selects the training data, with consideration of the consumer provided candidate training data. Since the training data directly influences the algorithm and performance of the trained ML entity, the ML training MnS producer may examine the consumer's provided training data and decide to select none, some or all of them. In addition, the ML training MnS producer may select some other training data that are available;

- trains the ML model using the selected training data;

- provides the training results (including the identifier of the ML entity generated from the initially trained ML model or the version number of the ML entity associated with the re-trained model, training performance results, etc.) to the ML training MnS consumer(s).

##### 6.2a.1.2.2 ML training initiated by producer

The ML training or re-training may be initiated by the ML training MnS producer, for instance as a result of performance evaluation of the ML entity or based on feedback or new training data received from the consumer, or when new training data, which are not from the consumer, describing the new network status/events become available.

Therefore, there is a need to monitor the performance and/or the KPIs of the ML entity and use the thresholds that the ML training MnS consumer configured for the ML training MnS producer to trigger the training or re-training.

When the ML training MnS producer decides to start the ML training, the producer performs the followings:

- selects the training data;

- trains the ML model using the selected training data;

- provides the training results (including the identifier of the ML entity generated from the initially trained ML model or the version number of the ML entity associated with the re-trained model, training performance, etc.) to the ML training MnS consumer(s) who have subscribed to receive the ML training results.

##### 6.2a.1.2.3 ML entity selection

For a given machine learning-based use case, different entities that apply the respective ML model or AI/ML inference function may have different inference requirements and capabilities. For example, one consumer with specific responsibility wishes to have an AI/ML inference function supported by an ML model or entity trained for city central business district where mobile users move at speeds not exceeding 30 km/hr. On the other hand, another consumer, for the same use case may support a rural environment and as such wishes to have an ML model and AI/ML inference function fitting that type of environment. The different consumers need to know the available versions of ML entities, with the variants of trained ML models or entities and to select the appropriate one for their respective conditions.

Besides, there is no guarantee that the available ML models/entities have been trained according to the characteristics that the consumers expect. As such the consumers need to know the conditions for which the ML models or ML entities have been trained to then enable them to select the models that are best fit to their conditions and needs.

The models that have been trained may differ in terms of complexity and performance. For example, a generic comprehensive and complex model may have been trained in a cloud-like environment, but such a model cannot be used in the gNB and instead, a less complex model, trained as a derivative of this generic model, could be a better candidate. Moreover, multiple less complex models could be trained with different levels of complexity and performance which would then allow different relevant models to be delivered to different consumers depending on operating conditions and performance requirements. The consumers need to know the alternative models available and interactively request and replace them when needed and depending on the observed inference‑related constraints and performance requirements.

##### 6.2a.1.2.4 Managing ML training processes

This relates to the management and controlling of the ML training processes.

To achieve the desired outcomes of any machine learning relevant use-case or task, the ML model applied for such use case or task, needs to be trained with the appropriate data. The training may be undertaken in a managed function or in a management function.

In either case, the network management system not only needs to have the required training capabilities but needs to also have the means to manage the training process of the ML models. The consumers need to be able to interact with the training process, e.g., to suspend or restart the process; and also need to manage and control the requests related to such training process.

##### 6.2a.1.2.5 Handling errors in data and ML decisions

Ideally, the ML models/entities (e.g., ML entity1 and ML entity2 in figure 6.2a.1.2.5-1) are trained on good quality data, i.e. data that was collected correctly and reflected the real network status to represent the expected context in which the ML entity is meant to operate. However, this is not always the case in real world as data cannot be completely error-free. Good quality data is void of errors, such as:

- Imprecise measurements

- Missing values or records

- Records which are communicated with a significant delay (in case of online measurements).

Without errors, an ML entity can depend on a few precise inputs, and does not need to exploit the redundancy present in the training data. However, during inference, the ML entity is very likely to come across these inconsistencies. When this happens, the ML entity shows high error in the inference outputs, even if redundant and uncorrupted data are available from other sources.



Figure 6.2a.1.2.5-1: The propagation of erroneous information

As such, the training function should attempt to identify errors in the input data. If an entity has been trained on erroneous or inconsistent data, the consumer should be made aware of such.

##### 6.2a.1.2.6 ML entity joint training

Each ML entity supports a specific type of inference. An AI/ML inference function may use one or more ML entities to perform the inference(s). When multiple ML entities are employed, these ML entities may operate together in a coordinated way, such as in a sequence, or even in a more complicated structure. In this case, any change in the performance of one ML entity may impact another, and consequently impact the overall performance of the whole AI/ML inference function.

There are different ways in which the group of ML entities may coordinate. An example is the case where the output of one ML entity can be used as input to another ML entity forming a sequence of interlinked ML entities. Another example is the case where multiple ML entities provide the output in parallel (either the same output type where outputs may be merged (e.g., using weights), or their outputs are needed in parallel as input to another ML entity. The group of ML entities needs to be employed in a coordinated way to support an AI/ML inference function.

Therefore, it is desirable that the ML models associated with these coordinated ML entities can be trained or re-trained jointly, so that the group of these ML entities can complete a more complex task jointly with better performance.

The ML entity joint training may be initiated by the ML training MnS producer or the ML training MnS consumer, with the grouping of the ML entities shared by the ML training MnS producer with the ML training MnS consumer.

##### 6.2a.1.2.7 ML entity validation performance reporting

During the ML training process, the generated ML entity needs to be validated. The purpose of ML validation is to evaluate the performance of the ML entity when performing on the validation data, and to identify the variance of the performance on the training data and the validation data. The training data and validation data are of the same pattern as they normally split from the same data set with a certain ratio in terms of quantity of the data samples.

In the ML training, the ML entity is generated based on the learning from the training data and validated using the validation data. The performance of the ML entity has tight dependency on the data (i.e., training data) from which the ML entity is generated. Therefore, an ML entity performing well on the training data may not necessarily perform well on other data e.g., while conducting inference. If the performance of ML entity is not good enough according to the result of ML validation, the ML entity will be tuned (i.e., the model associated with it be re-trained) and validated again. The process of ML entity tuning and validation is repeated by the ML training function, until the performance of the ML entity meets the expectation on both training data and validation data. The MnS producer subsequently selects one or more ML entities with the best level of performance on both training data and validation data as the result of the ML training, and reports accordingly to the consumer. The performance of each selected ML entity on both training data and validation data also needs to be reported.

The performance result of the validation may also be impacted by the ratio of the training data and the validation data. MnS consumer needs to be aware of the ratio of training data and the validation data, coupled with the performance score on each data set, in order to be confident about the performance of ML entity.

##### 6.2a.1.2.8 Training data effectiveness reporting

Training data effectiveness refers to the process of evaluating the contribution of a single data instance or a type of input training data (e.g., one measurement type among all types of input training data) to ML model training process.

To efficiently train a ML model, high quality and large volume of training data instances are considered essential. The open use of all available data can be costly, both in terms of data collection process and from a computational resources perspective since the data also contains the unnecessary data samples that are computed through the ML model. It is better that the training function evaluates the usefulness of different data samples and indicates that level of usefulness to the consumer so that the data used for re-training can be further enhanced/optimized.

The 3GPP management system needs to support means to report the extent of effectiveness of the different training data samples used in ML training based on insight of how the different portion of data contribute differently to the trained model accuracy.

#### 6.2a.1.3 Requirements for ML training

Table 6.2a.1.3-1

| 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 training. | ML training requested by consumer (clause 6.2a.1.2.1) |
| **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 training. | ML training requested by consumer (clause 6.2a.1.2.1) |
| **REQ- ML\_TRAIN-FUN-03** | The ML training MnS producer shall have a capability allowing the authorized ML training MnS consumer to specify the inference type of the ML entity to be trained. | ML training requested by consumer (clause 6.2a.1.2.1) |
| **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 training requested by consumer (clause 6.2a.1.2.1), ML training initiated by producer (clause 6.2a.1.2.2) |
| **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 entity. (See Note) | ML training initiated by producer (clause 6.2a.1.2.2) |
| **REQ- ML\_TRAIN-FUN-06** | The ML training MnS producer shall have a capability to provide the version number of the ML entity and the time when it is generated by ML re-training to the authorized ML training MnS consumer. | ML training requested by consumer (clause 6.2a.1.2.1), /ML training initiated by producer (clause 6.2a.1.2.2) |
| **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 training. | ML training requested by consumer (clause 6.2a.1.2.1), ML training initiated by producer (clause 6.2a.1.2.2), ML entity joint training (clause 6.2a.1.2.6) |
| **REQ- ML\_TRAIN-FUN-08** | The ML training MnS producer should have a capability to provide the grouping of ML entities to an authorized ML training MnS consumer to enable coordinated inference. | ML entity joint training (clause 6.2a.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 entities. | ML entity joint training (clause 6.2a.1.2.6) |
| **REQ- ML\_TRAIN-FUN-10** | The ML training MnS producer should have a capability to jointly train a group of ML entities and provide the training results to an authorized consumer. | ML entity joint training (clause 6.2a.1.2.6) |
| **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 entities including the contexts under which each of the models associated with the ML entities were trained. | ML model and ML entity selection (clause 6.2a.1.2.3) |
| **REQ-ML\_SELECT-02** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer to select an ML entity to be used for inference. | ML models and ML entity selection (clause 6.2a.1.2.3) |
| **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 entities of differing complexity and performance. | ML model and ML entity selection (clause 6.2a.1.2.3) |
| **REQ-ML\_SELECT-04** | The 3GPP management system shall have a capability to provide a selected ML entity to the authorized ML training MnS consumer. | ML model and ML entity selection (clause 6.2a.1.2.3) |
| **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 training, e.g. to modify the request or to delete the request. | ML training requested by consumer (clause 6.2a.2.1), Managing ML Training Processes (clause 6.2a.1.2.4) |
| **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 training requested by consumer (clause 6.2a.1.2.1),  Managing ML training processes (clause 6.2a.1.2.4) |
| **REQ-ML\_TRAIN- MGT-03** | 3GPP management system shall have a capability to enable an authorized ML training MnS consumer (e.g. the function/entity different from the function that generated a request for ML training) to request for a report on the outcomes of a specific training instance. | Managing ML training processes (clause 6.2a.1.2.4) |
| **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 training processes (clause 6.2a.1.2.4) |
| **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 training process and/or report about the outcomes of any such ML training process. | Managing ML training processes (clause 6.2a.1.2.4) |
| **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 a 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 6.2a.1.2.5) |
| **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 6.2a.1.2.5) |
| **REQ-ML\_ERROR-03** | The 3GPP management system shall enable a producer of data services (e.g. a gNB) to provide to an authorized consumer (e.g. an ML training function) 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 6.2a.1.2.5) |
| **REQ-ML\_ERROR-04** | The 3GPP management system shall enable a 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 6.2a.1.2.5) |
| **REQ-ML\_VLD-01** | The ML training MnS producer should have a capability to validate the ML entities during the ML training process and report the performance of the ML entities on both the training data and validation data to the authorized consumer. | ML entity validation performance reporting (clause 6.2a.1.2.7) |
| **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 training and validation process. | ML entity validation performance reporting (clause 6.2a.1.2.7) |
| **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 6.2a.1.2.8) |
| NOTE: The performance measurements and KPIs are specific to each type (i.e., the inference type that the ML entity supports) of ML entity. | | |

### 6.2a.2 Performance management for ML training and testing

#### 6.2a.2.1 Description

In the ML model training phase (including training and validation), the performance of ML entity needs to be evaluated. The performance is the degree to which the ML entities fulfil the objectives for which they were trained and can be evaluated for training data as training performance or for testing data as testing performance. The related performance indicators need to be collected and analyzed.

#### 6.2a.2.2 Use cases

##### 6.2a.2.2.1 Performance indicator selection for ML training and testing

The ML model training function may support training for single or different kinds of ML models and may support the capability to evaluate each kind of ML entity by one or more performance indicators.

The MnS consumer may prefer to use some performance indicator(s) over others to evaluate one kind of ML entity. The performance indicators for training mainly include the following aspects:

- ML training process monitors performance indicators: the performance indicators of the system that trains the ML entity, including training duration indicator.

- ML training model performance indicators: performance indicators of the ML entity itself, including but not limited to:

- Accuracy indicator,

- Precision indicator,

- Recall indicator,

- F1 score indicator,

- MSE (Mean Squared Error) indicator, and

- MAE (Mean Absolute Error) indicator,

- RMSE (Root Mean Square Error) indicator.

The MnS consumer may prefer to use some performance indicator(s) over others to evaluate one kind of ML entity. The performance indicators for testing mainly include the following aspects:

- ML testing model performance indicators: performance indicators of the ML entity itself, including but not limited to:

- Accuracy indicator,

- Precision indicator,

- Recall indicator,

- F1 score indicator,

- MSE(Mean Squared Error) indicator,

- MAE(Mean Absolute Error) indicator, and

- RMSE(Root Mean Square Error) indicator.

Therefore, the MnS producer for ML training and testing needs to provide the name(s) of supported performance indicator(s) for the MnS consumer to query and select for ML entity performance evaluation. The MnS consumer may also need to provide the performance requirements of the ML entity using the selected performance indicators.

The MnS producer for ML training and testing uses the selected performance indicators for evaluating ML training and testing, and reports with the corresponding performance score in the ML training report or ML testing report when the training or testing is completed.

##### 6.2a.2.2.2 ML entity performance indicators query and selection for ML training and testing

The ML entity performance evaluation and management is needed during training and testing. The related performance indicators need to be collected and analyzed. The MnS producer of ML training or testing should determine which indicators are needed, i.e., select some indicators based on the use case and use these indicators for performance evaluation.

The ML MnS consumer or testing may have different requests on AI/ML performance, depending on its use case and requirements, which may imply that different performance indicators may be relevant for performance evaluation. The MnS producer for ML training/testing can be queried to provide the information on supported performance indicators referring to ML training/testing. Such performance indicators in training phase may be for example accuracy/precision/recall/F1-score/MSE/MAE, and in test phase may be data drift in data statistics. Based on supported performance indicators in different phase as well as based on consumer’s requirements, the MnS consumer for ML training or ML testing may request a sub-set of supported performance indicators to be monitored and used for performance evaluation. Management capabilities are needed to enable the MnS consumer for ML training or ML testing to query the supported performance indicators and select a sub-set of performance indicators in training phase to be used for performance evaluation.

##### 6.2a.2.2.3 MnS consumer policy-based selection of ML entity performance indicators for ML training and testing

ML entity performance evaluation and management is needed during ML training phase. The related performance indicators need to be collected and analysed. The MnS producer for ML training should determine which indicators are needed or may be reported, i.e., select some indicators based on the service and use these indicators for performance evaluation.

The MnS consumer for ML training or testing may have differentiated levels of interest in the different performance dimensions or metrics. Thus, depending on its use case, the AI/ML MnS consumer may indicate the preferred behaviour and performance requirement that needs to be considered during training or testing of/from the ML entity by the ML MnS producer for ML training or testing. These performance requirements need not indicate the technical performance indicators used for ML training, testing or inference, such as "accuracy" or "precision" or "recall" or "MSE" or "MAE" or “F1 score" etc. The ML MnS consumer for ML training or testing may not be capable enough to indicate the performance metrics to be used for training or testing.

#### 6.2a.2.3 Requirements for ML training and testing performance management

Table 6.2a.2.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-ML\_TRAIN\_PM-1** | The ML Training or Testing MnS producer shall have a capability to allow an authorized consumer to get the capabilities about what kind of ML models the ML training function or ML testing function is able to train or test. | Performance indicator selection for ML training (clause 6.2a.2.2.1) |
| **REQ-ML\_TRAIN\_PM-2** | The ML Training or Testing MnS producer shall have a capability to allow an authorized consumer to query what performance indicators are supported by the ML training function or ML testing function for each kind of ML entity. | Performance indicator selection for ML training (clause 6.2a.2.2.1) |
| **REQ-ML\_TRAIN\_PM-3** | The ML Training or Testing MnS producer shall have a capability to allow an authorized consumer to select the performance indicators from those supported by the ML training function or ML testing function for reporting the training or testing performance for each kind of ML entity. | Performance indicator selection for ML training (clause 6.2a.2.2.1) |
| **REQ-ML\_TRAIN\_PM-4** | The ML Training MnS producer shall 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. | Performance indicator selection for ML training (clause 6.2a.2.2.1) |

### 6.2a.3 ML testing

#### 6.2a.3.1 Description

During ML training phase, after the training and validation, the ML entity needs to be tested to evaluate the performance of the ML entity when it conducts inference using the testing data. Testing may involve interaction with third parties (besides the developer of the ML training function), e.g., the operators may use the ML training function or third-party systems/functions that may rely on the inference results computed by the ML entity for testing.

If the testing performance is not acceptable or does not meet the pre-defined requirements, the consumer may request the ML training producer to re-train the ML model with specific training data and/or performance requirements.

#### 6.2a.3.2 Use cases

##### 6.2a.3.2.1 Consumer-requested ML entity testing

After receiving an ML training report about a trained ML entity from the ML training MnS producer, the consumer may request the ML testing MnS producer to test the ML entity before applying it to the target inference function.

The ML testing is to conduct inference on the tested ML entity using the testing data as inference inputs and produce the inference output for each testing dataset example.

The ML testing MnS producer may be the same as or different from the ML training MnS producer.

After completing the ML testing, the ML testing MnS producer provides the testing report indicating the success or failure of the ML testing to the consumer. For a successful ML testing, the testing report contains the testing results, i.e., the inference output for each testing dataset example.

The ML testing MnS producer needs to have the capabilities to provide the services needed to enable the consumer to request testing and receive results on the testing of an ML entity.

##### 6.2a.3.2.2 Producer-initiated ML entity testing

The ML entity testing may also be initiated by the MnS producer, after the ML entity is trained and validated. A consumer (e.g., an operator) may still need to define the policies (e.g., allowed time window, maximum number of testing iterations, etc.) for the testing of a given ML entity. The consumer may pre-define performance requirements for the ML entity testing and allow the MnS producer to decide on whether re-training/validation need to be triggered. Re-training may be triggered by the testing MnS producer itself based on the performance requirements supplied by the MnS consumer.

##### 6.2a.3.2.3 Joint testing of multiple ML entities

A group of ML entities may work in a coordinated manner for complex use cases. In such cases an ML entity is just one step of the inference processes of an AI/ML inference function, with the inference outputs of an ML entity as the inputs to the next ML entity.

The group of ML entities is generated by the ML training function. The group, including all contained ML entities, needs to be tested. After the ML testing of the group, the MnS producer provides the testing results to the consumer.

NOTE: This use case is about the ML entities testing during the training phase and is irrelevant to the testing cases that the ML entities have been deployed.

#### 6.2a.3.3 Requirements for ML testing

Table 6.2a.3.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-ML\_TEST-1** | The ML testing MnS producer shall have a capability to allow an authorized consumer to request the testing of a specific ML entity. | Consumer-requested ML entity testing (clause 6.2a.3.2.1) |
| **REQ-ML\_TEST-2** | The ML testing MnS producer shall have a capability to trigger the testing of an ML entity and allow the MnS consumer to set the policy for the testing. | Producer-initiated ML entity testing (6.2a.3.2.2) |
| **REQ-ML\_TEST-3** | The ML testing MnS producer shall have a capability to report the performance of the ML entity when it performs inference on the testing data. | Consumer-requested ML entity testing (clause 6.2a.3.2.1), and  producer-triggered ML entity testing (clause 6.2a.3.2.2) |
| **REQ-ML\_TEST-4** | The ML testing MnS producer shall have a capability allowing an authorized consumer to request the testing of a group of ML entities. | Joint testing of multiple ML entities (clause 6.2a.3.2.3) |

## 6.3 AI/ML emulation phase

### 6.3.1 Description

Before the ML entity 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 entity is validated and tested during development, the MnS consumer may wish to receive information from an inference emulation process that indicates if the ML entity 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.

### 6.3.3 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 entity or inference Function. | AI/ML Inference emulation (clause 6.3.2.1) |

## 6.4 ML entity deployment phase

### 6.4.1 ML entity loading

#### 6.4.1.1 Description

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

#### 6.4.1.2 Use cases

##### 6.4.1.2.1 Consumer requested ML entity loading

After a trained ML entity or the coordination group of ML entities are tested and optionally emulated, if the performance of the ML entity or the coordination group of ML entities meet the MnS consumer’s requirements, the MnS consumer may request to load the one or more ML entities to one or more target inference function(s) where the ML entity will be used for conducting inference. Once the ML entity 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 entity loading, the ML entity 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 entity loading

To enable more autonomous AI/ML operations, the MnS producer is allowed to load the ML entity or the coordination group of ML entities 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 entities loaded by the MnS producer meet the performance target. The policy could be, for example, the threshold of the testing performance of the ML entities, the threshold of the inference performance of the existing ML model, the time schedule allowed for ML entity 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 entities may be loaded per the policy to adapt to a specific load/traffic pattern.

##### 6.4.1.2.3 ML entity registration

After multiple iterations, there could be a large number of ML entities with different versions, deployment environments, performance levels, and functionalities. ML entity registration refers to the process of recording, tracking, controlling those trained ML entities 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 entity along with its loading information, e.g., ML entity metadata and relevant information (e.g., description, version, version date, target inference function, deployment environment, etc.).

#### 6.4.1.3 Requirements for ML entity loading

Table 6.4.1.3-1

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

## 6.5 AI/ML inference phase

### 6.5.1 AI/ML inference performance management

#### 6.5.1.1 Description

In the AI/ML inference phase, the performance of the AI/ML inference function and ML entity need to be evaluated against the MnS consumer's provided performance expectations/targets, to identify and timely fix any problem. Actions to fix any problem would be e.g., to trigger the ML re-training, ML testing, or re-deployment.

#### 6.5.1.2 Use cases

##### 6.5.1.2.1 AI/ML inference performance evaluation

In the AI/ML inference phase, the AI/ML inference function (including e.g., MDAF, NWDAF or RAN functions) uses one or more ML entities for inference to generate the AI/ML inference output. The performance of a running ML entity may degrade over time due to changes in network state, which will affect the related network performance and service. Thus, it is necessary to evaluate performance of the ML entity during the AI/ML inference process. If the inference output is executed, the network performance related to each AI/ML inference function also needs to be evaluated.

The consumer (e.g., a Network or Management function) may take some actions according to the AI/ML inference output provided by the AI/ML inference function. If the actions are taken accordingly, the network performance is expected to be optimized. Each AI/ML inference function has its specific focus and will impact the network performance from different perspectives.

The consumer may choose to not take any actions for various reasons, e.g., lacking confidence in the inference output, avoiding potential conflict with other actions or when no actions are needed or recommended at all according to the inference output.

For evaluating the performance of the AI/ML inference function and ML entity, the MnS producer responsible for ML inference performance management needs to be able to get the inference output generated by each AI/ML inference function. Then, the MnS producer can evaluate the performance based on the inference output and related network measurements (i.e., the actual output).

Depending on the performance evaluation results, some actions (e.g., deactivate the running entity, start retraining, change the running entity with a new one, etc) can be taken to avoid generating the inaccurate inference output.

To monitor the performance in the AI/ML inference phase, the MnS producer responsible for AI/ML inference performance management can perform evaluation periodically. The performance evaluation period may be determined based on the network change speed. Besides, a consumer (e.g., an operator) may wish to control and manage the performance evaluation capability. For example, the operator may configure the performance evaluation period of a specified ML entity.

##### 6.5.1.2.2 AI/ML performance measurements selection based on MnS consumer policy

Evaluation and management of the performance of an ML entity is needed during inference phase. The related performance measurements need to be collected and analysed. The MnS producer for inference should determine which measurements are needed or may be reported, i.e., select some measurements based on the service and use these measurements for performance evaluation.

The MnS consumer for inference may have differentiated levels of interest in the different performance dimensions or metrics. Thus, depending on its use case, the MnS consumer may indicate the preferred behaviour and performance requirement that needs to be considered during inference from the ML entity by the AI/ML inference MnS Producer. The AI/ML inference MnS consumer may not be capable enough to indicate the performance metrics. Instead, the AI/ML MnS consumer may indicate the requirement using a policy or guidance that reflects the preferred performance characteristics of the ML entity. Based on the indicated policy/guidance, the AI/ML MnS producer may then deduce and apply the appropriate performance indicators for inference. Management capabilities are needed to enable the MnS consumer to indicate the behavioural and performance policy/guidance that may be translated by the MnS producer into one or more technical performance measurements during inference.

#### 6.5.1.3 Requirements for AI/ML inference performance management

Table 6.5.1.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ- AI/ML\_INF\_PE-01** | The MnS producer responsible for AI/ML inference management shall have a capability enabling an authorized consumer to get the inference output provided by an AI/ML inference function (e.g., MDAF, NWDAF or RAN function). | AI/ML inference performance evaluation (clause 6.5.1.2.1) |
| **REQ- AI/ML\_INF\_PE-02** | The MnS producer responsible for AI/ML inference management shall have a capability enabling an authorized consumer to get the performance evaluation of an AI/ML inference output as measured by a defined set of performance metrics | AI/ML inference performance evaluation (clause 6.5.1.2.1) |
| **REQ- AI/ML\_INF\_PE-03** | The MnS producer responsible for AI/ML inference management shall have a capability enabling an authorized consumer to provide feedback about an AI/ML inference output expressing the degree to which the inference output meets the consumer's expectations. | AI/ML inference performance evaluation (clause 6.5.1.2.1) |
| **REQ- AI/ML\_INF\_PE-04** | The MnS producer responsible for AI/ML inference management shall have a capability enabling an authorized consumer to be informed about the executed actions that were triggered based on the inference output provided by an AI/ML inference function (e.g., MDAF, NWDAF or RAN function). | AI/ML inference performance evaluation (clause 6.5.1.2.1) |
| **REQ- AI/ML\_INF\_PE-05** | The MnS producer responsible for AI/ML inference management shall have a capability enabling an authorized consumer to obtain the performance data related to an ML entity or an AI/ML inference function (e.g., MDAF, NWDAF or RAN function). | AI/ML inference performance evaluation (clause 6.5.1.2.1) |
| **REQ-AI/ML\_PERF-SEL-1** | The ML training MnS producer shall have a capability allowing an authorized MnS consumer to discover supported AI/ML performance measurements related to AI/ML inference and select some of the desired measurements based on the MnS consumer’s requirements. | AI/ML performance measurements selection based on MnS consumer policy (clause 6.5.1.2.2) |
| **REQ-AI/ML\_PERF-POL-1** | The AI/ML MnS producer shall have a capability allowing the authorized MnS consumer to indicate a performance policy related to AI/ML inference phase. | AI/ML performance measurements selection based on MnS consumer policy (clause 6.5.1.2.2) |

### 6.5.2 AI/ML update control

#### 6.5.2.1 Description

In many cases, network conditions change makes the capabilities of the ML entity/entities decay, or at least become inappropriate for the changed conditions. In such cases, the MnS consumer should still be enabled to trigger updates, e.g., when the consumer realizes that the insight or decisions generated by the function are no longer appropriate for the observed network states, when the consumer observes the inference performance of ML entity/entities is decreasing.

The MnS consumer may request the AI/ML inference MnS producer to use an updated ML entity/entities for the inference with some specific performance requirements. This gives flexibility to the AI/ML inference MnS producer on how to address the requirements by for example getting ML entity/entities updated, which may be loading the already trained ML entity/entities or may lead to requesting to train/re-train the ML entity/entities by utilizing the ML training MnS.

#### 6.5.2.2 Use cases

##### 6.5.2.2.1 Availability of new capabilities or ML entities

Depending on their configurations, AI/ML inference functions may learn new characteristics during their utilization, e.g., if they are configured to learn through reinforcement learning or if they are configured to download new versions of their constituent ML entities. In such cases, the authorized consumer of AI/ML may wish to be informed by the AI/ML Inference MnS producer (e.g., the operator, a management function, or a network function) about their new capabilities.

##### 6.5.2.2.2 Triggering ML entity update

When the inference capabilities of AI/ML inference functions degenerate, the typical action may be to trigger re-training of the constituent ML entities. It is possible, however, that the AI/ML inference MnS producer only offers inference capabilities and is not equipped with capabilities to update, train/re-train its constituent ML entities. Nevertheless, the authorized MnS consumer may still need to request for improvements in the capabilities of the AI/ML inference function. In such cases, the authorized MnS consumer may still wish to request for an improvement and may specify in its request e.g., a new version of the ML entities, i.e., to have the ML entities updated or re-trained. The corresponding internal actions taken by the AI/ML MnS inference producer may not be necessarily known by the consumer.

The AI/ML inference MnS consumer needs to request the AI/ML inference MnS producer to update its capabilities or its constituent ML entities and the AI/ML MnS producer should respond accordingly. For example, the AI/ML inference MnS producer may download new software that supports the required updates, download from a remote server a file containing configurations and parameters to update one or more of its constituent ML entities, or it may trigger one or more remote or local AI/ML-related processes (including training/re-training, testing, etc.) needed to generate the required updates. Related notifications for update can be sent to the AI/ML inference MnS consumer to indicate the information of the update process, e.g., the update is finished successfully, the maximum time taken to complete the update is reached but the performance does not achieve the requirements, etc.

Besides, an AI/ML inference MnS consumer may wish to manage the update process(es), e.g., to define policies on how often the update may occur, suspend or restart the update or adjust the update conditions or characteristics, the requirements could include, e.g., the times when the update may be executed, the expected achievable performance for updating, the expected time taken to complete the update, etc.

#### 6.5.2.3 Requirements for AIML update control

Table 6.5.2.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-AIML\_UPDATE-1** | The AI/ML Inference MnS producer should have a capability to inform an authorized MnS consumer of the availability of AI/ML capabilities or ML entities or versions thereof (e.g., as learned through a training process or as provided via a software update) and the readiness to update the AI/ML capabilities of the respective network function when triggered | Availability of new capabilities or ML entities (clause 6.5.2.2.1) |
| **REQ-AIML\_UPDATE-2** | The AI/ML Inference MnS producer should have a capability to inform an authorized MnS consumer of the expected performance gain if/when the AI/ML capabilities or ML entities of the respective network function are updated with/to the specific set of newly available AI/ML capabilities | Availability of new capabilities or ML entities (clause 6.5.2.2.1) |
| **REQ-AIML\_UPDATE-3** | The AI/ML Inference MnS producer should have a capability to allow an authorized MnS consumer to request the AI/ML MnS producer to update its ML entities using a specific version of newly available AI/ML capabilities or ML entities or using AI/ML capabilities or ML entities with requirements (e.g., the minimum achievable performance after updating, the maximum time taken to complete the update, etc).. | Triggering ML entity update (clause 6.5.2.2.2) |
| **REQ-AIML\_UPDATE-4** | The AI/ML Inference MnS producer should have a capability for the AI/ML MnS producer to inform an authorized MnS consumer about of the process or outcomes related to any request for updating the AI/ML capabilities or ML entities | Triggering ML entity update (clause 6.5.2.2.2) |
| **REQ-AIML\_UPDATE-5** | The AI/ML Inference MnS producer should have a capability for the AI/ML MnS producer to inform an authorized MnS consumer about of the achieved performance gain following the update of the AI/ML capabilities of a network function with/to the specific newly available ML entities or set of AI/ML capabilities | Triggering ML entity update (clause 6.5.2.2.2) |
| **REQ-AIML\_UPDATE-6** | The AI/ML Inference MnS producer should have a capability for an authorized MnS consumer (e.g., an operator or the function/entity that generated the request for updating the AI/ML capabilities) to manage the request and subsequent process, e.g. to suspend, re-activate or cancel the request or process; or to adjust the characteristics of the capability update; or to define how often the update may occur, suspend, restart or cancel the request or to further adjust the requirements of the update. | Triggering ML entity update (clause 6.5.2.2.2) |

### 6.5.3 AI/ML inference capabilities management

#### 6.5.3.1 Description

A network or management function that applies AI/ML to accomplish specific tasks may be considered to have one or more ML entities, each having specific capabilities.

Different network functions, e.g., MDA Functions, may need to rely on existing AI/ML capabilities to accomplish the desired inference. However, the details of such ML-based solutions (i.e., which ML entities are applied and how) for accomplishing those inference functionalities is not obvious. The management services are required to identify the capabilities of the involved ML entities and to map those capabilities to the desired logic.

#### 6.5.3.2 Use cases

##### 6.5.3.2.1 Identifying capabilities of ML entities

Network functions, especially network automation functions, may need to rely on capabilities of ML entities that are not internal to those network functions to accomplish the desired automation (inference). For example, as stated in TS 28.104 [2], “An MDA Function may optionally be deployed as one or more AI/ML inference function(s) in which the relevant ML entities are used for inference per the corresponding MDA.” Similarly, owing to the differences in the kinds and complexity of intents that need to be fulfilled, an intent fulfillment solution may need to employ the capabilities of existing AI/ML inference functions to fulfill the intents. In any such case, management services are required to identify the capabilities of those existing ML entities that are employed by AI/ML inference functions.



Figure 6.5.3.2.1-1: Request and reporting on AI/ML inference capabilities

Figure 6.5.3.2.1-1 shows that the consumer may wish to obtain information about the available AI/ML inference capabilities to determine how to use them for the consumer's needs, e.g., for fulfillment of intent targets or other automation targets.

##### 6.5.3.2.2 Mapping of the capabilities of ML entities

Besides the discovery of the capabilities of ML entities, services are needed for mapping the ML entities and capabilities. In other words, instead of the consumer discovering specific capabilities, the consumer may want to know the ML entities that can be used to achieve a certain outcome. For this, the producer should be able to inform the consumer of the set of available ML entities that together achieve the consumer's automation needs.

In the case of intents for example, the complexity of the stated intents may significantly vary - from simple intents which may be fulfilled with a call to a single ML entity to complex intents that may require an intricate orchestration of multiple ML entities. For simple intents, it may be easy to map the execution logic to one or multiple ML entities. For complex intents, it may be required to employ multiple ML entities along with a corresponding functionality that manages their interrelated execution. The usage of the ML entities requires the awareness of their capabilities and interrelations.

Moreover, given the complexity of the required mapping to the multiple ML entities, services should be supported to provide the mapping of ML entities and capabilities.

#### 6.5.3.3 Requirements for AI/ML inference capabilities management

Table 6.5.3.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-ML\_CAP-01** | The AI/ML inference MnS Producer shall have a capability allowing an authorized MnS consumer to request the capabilities of existing ML entities available within the AI/ML inference producer. | Identifying capabilities of ML entities (clause 6.5.3.2.1) |
| **REQ- ML\_CAP-02** | The AI/ML inference MnS Producer shall have a capability to report to an authorized MnS consumer the capabilities of an ML entity as a decision described as a triplet <object(s), parameters, metrics> with the entries respectively indicating: the object or object types for which the ML entity can undertake optimization or control; the configuration parameters on the stated object or object types, which the ML entity optimizes or controls to achieve the desired outcomes; and the network metrics which the ML entity optimizes through its actions. | Identifying capabilities of ML entities (clause 6.5.3.2.1) |
| **REQ-ML\_CAP-03** | The AI/ML inference MnS Producer shall have a capability to report to an authorized MnS consumer the capabilities of an ML entity as an analysis described as a tuple <object(s), characteristics> with the entries respectively indicating: the object or object types for which the ML entity can undertake analysis; and the network characteristics (related to the stated object or object types) for which the ML entity produces analysis | Identifying capabilities of ML entities (clause 6.5.3.2.1) |
| **REQ-ML\_CAP-04** | The AI/ML inference MnS Producer shall have a capability allowing an authorized MnS consumer to request a mapping of the consumer's inference targets to the capabilities of one or more ML entities. | Mapping of the capabilities of ML entities (clause 6.5.3.2.2) |

### 6.5.4 AI/ML inference capability configuration management

#### 6.5.4.1 Description

The AI/ML inference function and the associated ML entity may need to be managed and configured to conduct inference in the 5G system to align with the consumer´s expectation, e.g., to enable the AI/ML inference function to perform inference.

The MnS producer for AI/ML inference management needs to provide a capability for configuration of the AI/ML inference function.

#### 6.5.4.2 Use cases

##### 6.5.4.2.1 Managing NG-RAN AI/ML-based distributed Network Energy Saving

An NG-RAN AI/ML-based distributed Network Energy Saving capability may use one or more ML entities to derive energy saving recommendations.

This NG-RAN AI/ML-based distributed Network Energy Saving capability needs to be managed. The MnS consumer monitors the network performance and determines whether to, and when to activate or deactivate an AI/ML-based Distributed Network Energy Saving function. The activation and deactivation actions for AI/ML-based Distributed Network Energy Saving conducted by the MnS producer may also be triggered by some defined policies provided by the consumer.

##### 6.5.4.2.2 Managing NG-RAN AI/ML-based distributed Mobility Optimization

An AI/ML-based distributed Mobility Optimization capability may use one or more ML entities to derive handover recommendations.

This NG-RAN AI/ML-based distributed Mobility Optimization capability needs to be managed. The MnS consumer monitors the network performance and determines whether to, and when to activate or deactivate an AI/ML-based Distributed Mobility Optimization function. The activation and deactivation actions for AI/ML-based Distributed Mobility Optimization conducted by the MnS producer may also be triggered by some defined policies provided by the consumer.

##### 6.5.4.2.3 Managing NG-RAN AI/ML-based distributed Load Balancing

An NG-RAN AI/ML-based distributed Load Balancing capability may use one or more ML entities to derive load balancing recommendations.

This NG-RAN AI/ML-based distributed Load Balancing capability needs to be managed. The MnS consumer monitors the network performance and determines whether to, and when to activate or deactivate an AI/ML-based Distributed Load balancing function. The activation and deactivation actions for AI/ML-based Distributed Load balancing conducted by the MnS producer may also be triggered by some defined policies provided by the consumer.

#### 6.5.4.3 Requirements for AI/ML inference management

Table 6.5.4.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ- AI/ML\_INF-01** | The MnS producer of NG-RAN AI/ML-based distributed Network Energy Saving should enable an authorized MnS consumer to request to manage the Network Energy Saving inference capability | Managing AI/ML-based for NG-RAN distributed Network Energy Saving (clause 6.5.4.2.1) |
| **REQ- AI/ML\_INF-02** | The MnS producer of NG-RAN AI/ML-based distributed Mobility Optimization should enable an authorized MnS consumer to request to manage the Mobility Optimization inference capability | Managing AI/ML-based for NG-RAN distributed Mobility Optimization (clause 6.5.4.2.2) |
| **REQ- AI/ML\_INF-03** | The MnS producer of NG-RAN AI/ML-based distributed Load Balancing should enable an authorized MnS consumer to request to manage the Load Balancing inference capability | Managing AI/ML-based for NG-RAN distributed Load Balancing (clause 6.5.4.2.3) |
| **REQ-AIML\_ INF\_ACT-1** | The MnS producer for AI/ML inference management should have a capability allowing an authorized MnS consumer to activate and deactivate an ML inference function. | Managing AI/ML-enabled for Distributed Network Energy Saving (clause 6.5.4.2.1)  Managing AI/ML-enabled for distributed Mobility Optimization (clause 6.5.4.2.2)  Managing AI/ML-enabled for distributed Load balancing (clause 6.5.4.2.3) |
| **REQ-AIML\_ INF\_ACT-2** | The MnS producer for AI/ML inference management should have a capability to allow an authorized MnS consumer to provide the policy for activating and deactivating inference function.  Note: The policies instructing the ML MnS producer on how or/and when to activate which ML capabilities. | Managing AI/ML-enabled for Distributed Network Energy Saving (clause 6.5.4.2.1)  Managing AI/ML-enabled for distributed Mobility Optimization (clause 6.5.4.2.2)  Managing AI/ML-enabled for distributed Load balancing (clause 6.5.4.2.3) |

### 6.5.5 Executing AI/ML Inference

#### 6.5.5.1 Description

Different functionalities in the network or management domains may utilize AI/ML inference techniques to conduct their tasks under different contexts. Depending on the contexts, the outcome of the ML entity at inference might be different. The history of such inference outcome and the corresponding context within which they were taken may be of interest to different consumers.

#### 6.5.5.2 Use cases

##### 6.5.5.2.1 AI/ML Inference History - tracking inferences and context

For different automation requirements in specific network domain, management/automation functions (e.g., MDAS, SON) may apply ML functionality to make the appropriate inferences in different contexts. The context is the set of appropriate conditions under which the inference was made including network conditions, traffic characteristics, time of day, weather, and climate, etc. And depending on the contexts, the different inferences may have different outcomes. The inference history, which is the history of such inferences and the contexts within which they are taken, may be of interest to different consumers. The AI/ML inference history includes recommendations and insights derived by the ML entity and the contexts, e.g., network resources, time periods, traffic conditions, etc. under which those recommendations and insights were derived.

The inferences (need to be tracked for future reference, e.g., to evaluate the appropriateness/effectiveness of the inference outcome for those contexts or to evaluate degradations in the ML entity's performance. For this, the network not only needs to have the required inference capabilities but needs also to have the means to track and enable usage of the history of the inferences made by the ML entity. The MnS producer, i.e., a specific AI/ML inference function should also provide the capability for AI/ML inference history Control, the means to control the process of compiling and reporting on AI/ML inference history.

ML Inference History Control

MLEntity

Request ML Inference History

Report on ML Inference History

**ML MnS Producer (provides Inference History )**

**MnS Consumer**

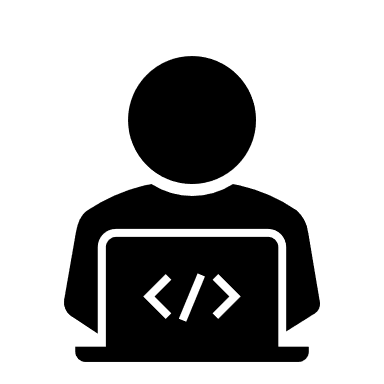
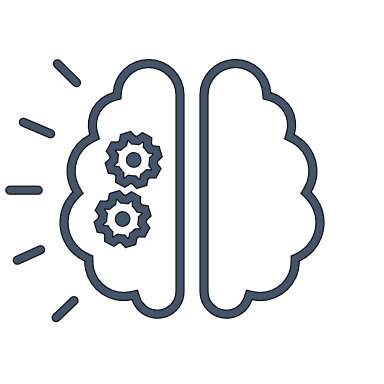


Figure 6.5.5.2.1-1: Example use and control of AI/ML inference history request and reporting.

#### 6.5.5.3 Requirements for Executing AI/ML Inference

Table 6.5.5.3-1

| Requirement label | Description | Related use case(s) |
| --- | --- | --- |
| **REQ-AI/ML\_INF-HIST-01** | The MnS producer for AI/ML inference management should have a capability allowing an authorized consumer to request the inference history of a specific ML entity. | AI/ML Inference History - tracking inferences and context (clause 6.5.5.2.1) |
| **REQ-AI/ML-INF-HIST-02** | The MnS producer for AI/ML inference management should have a capability enabling an authorized consumer to define the reporting characteristics (e.g., reporting period) related to a specific instance of ML inference history or the reporting thereof. | AI/ML Inference History - tracking inferences and context (clause 6.5.5.2.1) |

# 7 Information model definitions for AI/ML management

## 7.1 Imported and associated information entities

### 7.1.1 Imported information entities and local labels

Table 7.1.1-1

|  |  |
| --- | --- |
| Label reference | Local label |
| 3GPP TS 28.622 [12], IOC, Top | Top |
| 3GPP TS 28.622 [12], IOC, SubNetwork | SubNetwork |
| 3GPP TS 28.622 [12], IOC, ManagedElement | ManagedElement |
| 3GPP TS 28.622 [12], IOC, ManagedFunction | ManagedFunction |
| 3GPP TS 28.622 [12], IOC, ThresholdMonitor | ThresholdMonitor |
| 3GPP TS 28.541 [18], IOC, GNBCUCPFunction | GNBCUCPFunction |
| 3GPP TS 28.104 [2], IOC, MDAFunction | MDAFunction |
| 3GPP TS 28.622 [12], dataType, TimeWindow | TimeWindow |
| 3GPP TS 28.622 [12], dataType, GeoArea | GeoArea |
| 3GPP TS 28.622 [12], dataType, ThresholdInfo | ThresholdInfo |
| 3GPP TS 28.622 [12], dataType, ProcessMonitor | ProcessMonitor |

### 7.1.2 Associated information entities and local labels

Table 7.1.2-1

|  |  |
| --- | --- |
| Label reference | Local label |
| 3GPP TS 28.104 [2], IOC, MDAFunction | MDAFunction |
| 3GPP TS 28.541 [18], IOC, NWDAFFunction | NWDAFFunction |

## 7.2 Void

## 7.2a Common information model definitions for AI/ML management

### 7.2a.1 Class diagram

#### 7.2a.1.1 Relationships

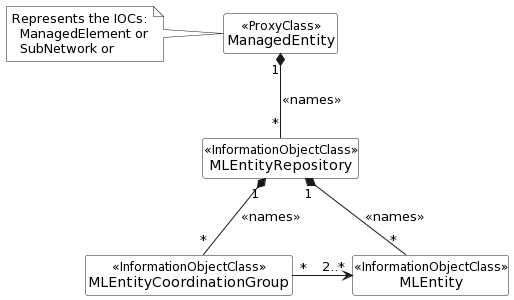


Figure 7.2a.1.1-1: Relations for common information models for AI/ML management

#### 7.2a.1.2 Inheritance

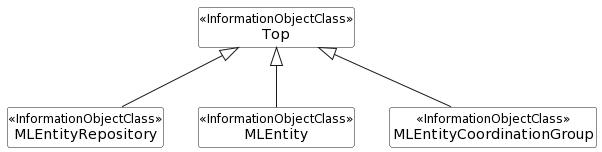


Figure 7.2a.1.2-1: Inheritance Hierarchy for common information models for AI/ML management

### 7.2a.2 Class definitions

#### 7.2a.2.1 MLEntity

##### 7.2a.2.1.1 Definition

This IOC represents the ML entity. ML model or ML entity are not subjects for standardization.

The MLEntity may contain 3 types of contexts - TrainingContext, ExpectedRunTimeContext and RunTimeContext which represent status and conditions of the MLEntity. 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 ThresholdMnonitor 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 MLEntity MOI has been instantiated, the MnS Consumer can request MnS producer to instantiate a ThresholdMonitor MOI and update the reference in the MLEntity MOI that can be used by the MnS producer to decide on the re-training of the MLEntity. The MnS producer can be ML Training MnS producer or ML Inference MnS Producer.

##### 7.2a.2.1.2 Attributes

Table 7.2a.2.1.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| mLEntityId | M | T | F | F | T |
| inferenceType | M | T | F | F | T |
| mLEntityVersion | 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 |
| **Attribute related to role** |  |  |  |  |  |
| retrainingEventsMonitorRef | O | T | T | F | T |
| sourceTrainedMLEntityRef | CM | T | F | F | T |

##### 7.2a.2.1.3 Attribute constraints

Table 7.2a.2.1.3-1

|  |  |
| --- | --- |
| Name | Definition |
| trainingContext Support Qualifier | Condition: The trainingContext represents the status and conditions related to training and should be added when training is completed. |
| sourceTrainedMLEntityRef Support Qualifier | Condition: The MLEntity MOI containing this attribute represents an ML entity loaded to an inference function. |

##### 7.2a.2.1.4 Notifications

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

#### 7.2a.2.2 MLEntityRepository

##### 7.2a.2.2.1 Definition

The IOC MLEntityRepository represents the repository that contains the ML entities .

The MLEntityRepository MOI may contain one or more MLEntity(s).

##### 7.2a.2.2.2 Attributes

Table 7.a.2.2.2-1

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

##### 7.2a.2.2.3 Attribute constraints

None.

##### 7.2a.2.2.4 Notifications

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

#### 7.2a.2.3 MLEntityCoordinationGroup

##### 7.2a.2.3.1 Definition

This IOC represents the group of ML entities, which can be trained and tested jointly and used to perform inference in a coordinated way.

One ML entity may have dependencies on one or more of the other ML entities of the same group.

One group is associated with at least two ML entities.

##### 7.2a.2.3.2 Attributes

Table 7.2a.2.3.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
|  |  |  |  |  |  |
| **Attribute related to role** |  |  |  |  |  |
| memberMLEntityRefList | M | T | F | F | T |

##### 7.2a.2.3.3 Attribute constraints

None.

##### 7.2a.2.3.4 Notifications

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

## 7.3 Void

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

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

#### 7.3a.1.1 Class diagram

##### 7.3a.1.1.1 Relationships

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

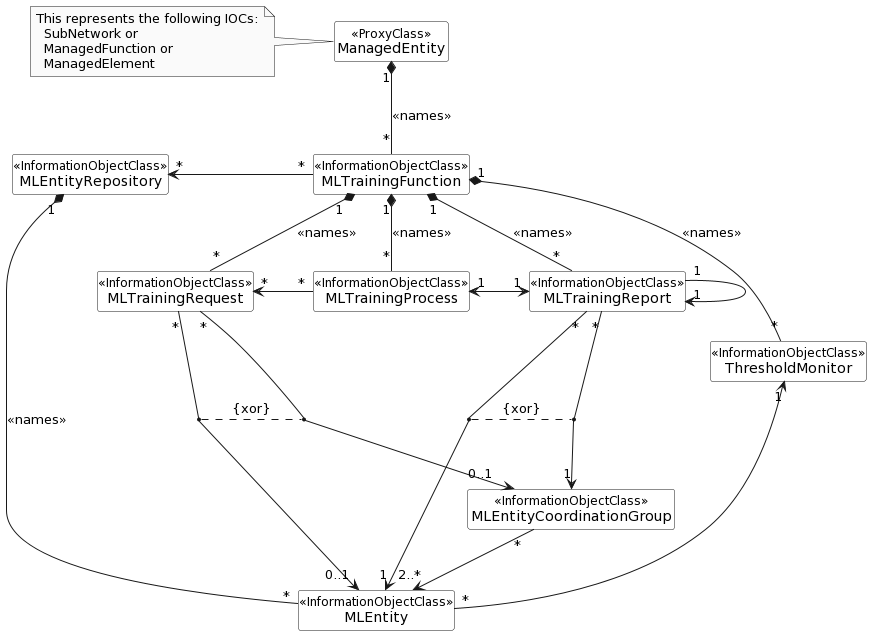


Figure 7.3a.1.1.1-1: NRM fragment for ML training

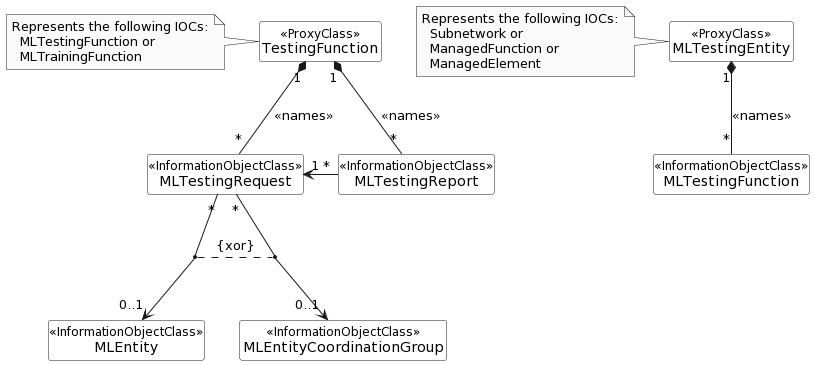


Figure 7.3a.1.1.1-2: NRM fragment for ML testing

##### 7.3a.1.1.2 Inheritance



Figure 7.3a.1.1.2-1: Inheritance Hierarchy for ML training related NRMs

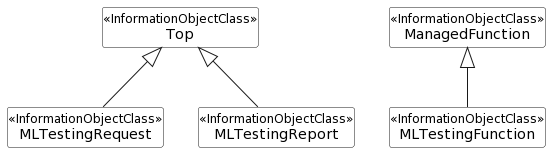


Figure 7.3a.1.1.2-2: Inheritance Hierarchy for ML testing related NRMs

#### 7.3a.1.2 Class definitions

##### 7.3a.1.2.1 MLTrainingFunction

###### 7.3a.1.2.1.1 Definition

The IOC MLTrainingFunction represents the entity that undertakes ML training. The MOI of MLTrainingFunction is also the container of the MLTrainingRequest MOI(s).

The entity represented by MLTrainingFunction MOI supports training of one or more MLEntity(s).

###### 7.3a.1.2.1.2 Attributes

Table 7.3a.1.2.1.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| **Attribute related to role** |  |  |  |  |  |
| mLEntityRepositoryRef | 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 created by the ML training MnS consumer.

The MLTrainingRequest MOI is contained under one MLTrainingFunction MOI.

The MLTrainingRequest MOI may represent the request for initial ML training or re-training. For ML re-training, the MLTrainingRequest is associated to one MLEntity for re-training a single ML entity, or associated to one MLEntityCoordinationGroup for re-training a group of coordinated ML entities.

The MLTrainingRequest may have a source to identify its origin, which may be used to prioritize the training resources for different sources. The sources may be for example the network functions, operator roles, or other functional differentiations.

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

In case the request is accepted, the ML training MnS producer decides when to start the ML training based on 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 consumer’s request provided candidate training data if any. The ML training MnS producer may examine the 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 consumer’s requirements for the MLentity training;

- trains the MLEntity 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".

###### 7.3a.1.2.2.2 Attributes

Table 7.3a.1.2.2.1-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| inferenceType | 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 |
| cancelRequest | O | T | T | F | T |
| suspendRequest | O | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| mLEntityToTrainRef | CM | T | F | F | T |
| mLEntityCoordinationGroupToTrainRef | CM | T | F | F | T |

###### 7.3a.1.2.2.3 Attribute constraints

Table 7.3a.1.2.2.3-1

|  |  |
| --- | --- |
| Name | Definition |
| inferenceType Support Qualifier | Condition: MLTrainingRequest MOI represents the request for initial ML training. |
| mLEntityToTrainRef Support Qualifier | Condition: MLTrainingRequest MOI represents the request for ML re-training. |
| mLEntityCoordinationGroupToTrainRef Support Qualifier | Condition: MLTrainingRequest MOI represents the request for joint training of a group of ML entities. |

###### 7.3a.1.2.2.4 Notifications

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

##### 7.3a.1.2.3 MLTrainingReport

###### 7.3a.1.2.3.1 Definition

The IOC MLTrainingReport represents the ML model training report that is provided by the training MnS producer.

The MLTrainingReport MOI is contained under one MLTrainingFunction MOI.

###### 7.3.1.2.3.2 Attributes

Table 7.3a.1.2.3.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| areConsumerTrainingDataUsed | M | T | F | F | T |
| usedConsumerTrainingData | CM | T | F | F | T |
| modelConfidenceIndication | O | T | F | F | T |
| modelPerformanceTraining | M | T | F | F | T |
| areNewTrainingDataUsed | M | T | F | F | T |
| **Attribute related to role** |  |  |  |  |  |
| trainingRequestRef | CM | T | F | F | T |
| trainingProcessRef | M | T | F | F | T |
| lastTrainingRef | CM | T | F | F | T |
| mLEnityGeneratedRef | M | T | F | F | T |
| mLEnityCoordinationGroupGeneratedRef | CM | T | F | F | T |
| mLEntityRef | M | T | F | F | T |

###### 7.3a.1.2.3.3 Attribute constraints

Table 7.3a.1.2.3.3-1

|  |  |
| --- | --- |
| Name | Definition |
| usedConsumerTrainingData Support Qualifier | Condition: The value of areConsumerTrainingDataUsed attribute is ALL or PARTIALLY. |
| trainingRequestRef Support Qualifier | Condition: The MLTrainingReport MOI represents the report for the ML model training that was requested by the MnS consumer (via MLTrainingRequest MOI). |
| lastTrainingRef Support Qualifier | Condition: The MLTrainingReport MOI represents the report for the ML model training that was not initial training (i.e. the model has been trained before). |
| mLEnityCoordinationGroupGeneratedRef Support Qualifier | Condition: The MLTrainingReport MOI represents the report for a joint training of a group of ML entities. |

###### 7.3a.1.2.3.4 Notifications

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

##### 7.3a.1.2.4 MLTrainingProcess

###### 7.3a.1.2.4.1 Definition

The IOC MLTrainingProcess represents the ML training process.

One MLTrainingProcess MOI may be instantiated for each MLTrainingRequest MOI or a set of MLTrainingRequest MOIs.

For each MLEntity under training, a MLTrainingProcess is instantiated, i.e. an MLTrainingProcess is associated with exactly one MLEntity.The MLTrainingProcess may be associated with one or more MLTrainingRequest MOI.

The MLTrainingProcess does not have to correspond to a specific MLTrainingRequest, i.e. a MLTrainingRequest does not have to be associated to a specific MLTrainingProcess. The MLTrainingProcess may be managed separately from the MLTrainingRequest MOIs, e.g. the MLTrainingRequest MOI may come from consumers which are network functions while the operator may wish to manage the MLTrainingProcess that is instantiated following the requests. Thus, the MLTrainingProcess may be associated to either one or more MLTrainingRequest MOI.

Each MLTrainingProcess instance needs to be managed differently from the related MLEntity, although the MLTrainingProcess may be associated to only one MLEntity. For example, the MLTrainingProcess may be triggered to start with a specific version of the MLEntity and multiple MLTrainingProcess instances may be triggered for different versions of the MLEntity. In either case the MLTrainingProcess instances are still associated with the same MLEntity but are managed separately from the MLEntity.

Each MLTrainingProcess has a priority that may be used to prioritize the execution of different MLTrainingProcess instances. By default, the priority of the MLTrainingProcess may be related in a 1:1 manner with the priority of the MLTrainingRequest for which the MLTrainingProcess is instantiated.

Each MLTrainingProcess may have one or more termination conditions used to define the points at which the MLTrainingProcess may terminate.

The "progressStatus" attribute represents the status of the ML model training and includes information the ML training MnS consumer can use to monitor the progress and results. The data type of this attribute is "ProcessMonitor" (see 3GPP TS 28.622 [12]). The following specializations are provided for this data type for the ML training process:

- The "status" attribute values are "RUNNING", "CANCELLING", "SUSPENDED", "FINISHED", and "CANCELLED". The other values are not used.

- The "timer" attribute is not used.

- When the "status" is equal to "RUNNING" the "progressStateInfo" attribute shall indicate one of the following states: "COLLECTING\_DATA", "PREPARING\_TRAINING\_DATA", "TRAINING".

- No specifications are provided for the "resultStateInfo" attribute. Vendor specific information may be provided though.

When the training is completed with "status" equal to "FINISHED", the MLT MnS producer provides the training report, by creating an MLTrainingReport MOI, to the MLT MnS consumer.

###### 7.3a.1.2.4.2 Attributes

Table 7.3a.1.2.4.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| priority | M | T | T | F | T |
| terminationConditions | M | T | T | F | T |
| progressStatus | M | T | F | F | T |
| cancelProcess | O | T | T | F | T |
| suspendProcess | O | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| trainingRequestRef | CM | T | F | F | T |
| trainingReportRef | M | T | F | F | T |
| mLEntityGeneratedRef | CM | T | F | F | T |
| mLEntityRef | M | T | F | F | T |

###### 7.3a.1.2.4.3 Attribute constraints

Table 7.3a.1.2.4.3-1

|  |  |
| --- | --- |
| Name | Definition |
| trainingRequestRef Support Qualifier | Condition: The MLTrainingReport MOI represents the report for the ML model training that was requested by the training MnS consumer (via MLTrainingRequest MOI). |
| mLEntityGeneratedRef Support Qualifier | Condition: The MLTrainingProcess MOI is instantiated to retrain an existing MLEntity. |

###### 7.3a.1.2.4.4 Notifications

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

##### 7.3a.1.2.5 MLTestingFunction

###### 7.3a.1.2.5.1 Definition

The ML entity testing may be conducted by the ML training function, or by a separate function.

In case the ML entity testing is conducted by a function separate from the ML training function, the IOC MLTestingFunction is instantiated and represents the logical function that undertakes ML entity testing.

The entity represented by MLTestingFunction MOI supports testing of one or more MLEntity(s).

###### 7.3a.1.2.5.2 Attributes

Table 7.3a.1.2.5.2-1

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

###### 7.3a.1.2.5.3 Attribute constraints

None.

###### 7.3a.1.2.5.4 Notifications

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

##### 7.3a.1.2.6 MLTestingRequest

###### 7.3a.1.2.6.1 Definition

The IOC MLTestingRequest represents the ML entity testing request that is created by the ML testing MnS consumer.

The MLTestingRequest MOI is contained under one MLTestingFunction MOI or MLTrainingFunction MOI which represents the logical function that conducts the ML entity testing. Each MLTestingRequest is associated to at least one MLEntity.

In case the request is accepted, the ML testing MnS producer decides when to start the ML testing. Once the MnS producer decides to start the testing based on the request, the ML testing MnS producer:

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

- prepares and selects the required testing data;

- tests the MLEntity by performing inference using the selected testing data, and

- reports the performance of the MLEntity when it performs on the selected testing data.

The MLTestingRequest may have a requestStatus field to represent the status of the request:

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

###### 7.3a.1.2.6.2 Attributes

Table 7.3a.1.2.6.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| requestStatus | M | T | F | F | T |
| cancelRequest | O | T | T | F | T |
| suspendRequest | O | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| mLEntityToTestRef | CM | T | F | F | T |
| mLEntityCoordinationGroupToTestRef | CM | T | F | F | T |

###### 7.3a.1.2.6.3 Attribute constraints

Table 7.3a.1.2.6.3-1

|  |  |
| --- | --- |
| Name | Definition |
| mLEntityToTestRef Support Qualifier | Condition: The MLTestingRequest MOI represents the request for testing of a single ML entity. |
| mLEntityCoordinationGroupToTestRef Support Qualifier | Condition: The MLTestingRequest MOI represents the request for joint testing of a group of ML entities. |

###### 7.3a.1.2.6.4 Notifications

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

##### 7.3a.1.2.7 MLTestingReport

###### 7.3a.1.2.7.1 Definition

The IOC MLTestingReport represents the ML testing report that is provided by the ML testing MnS producer.

The MLTestingReport MOI is contained under one MLTestingFunction MOI or MLTrainingFunction MOI which represents the logical function that conducts the ML entity testing.

For the joint testing of a group of ML entities, the ML testing report contains the testing results for every ML entity in the group.

###### 7.3a.1.2.7.2 Attributes

Table 7.3a.1.2.7.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| modelPerformanceTesting | M | T | F | F | T |
| mLTestingResult | M | T | F | F | T |
| **Attribute related to role** |  |  |  |  |  |
| testingRequestRef | CM | T | F | F | T |

###### 7.3a.1.2.7.3 Attribute constraints

Table 7.3a.1.2.7.3-1

|  |  |
| --- | --- |
| Name | Definition |
| testingRequestRef Support Qualifier | Condition: The MLTestingReport MOI represents the report for the ML model testing that was requested by the MnS consumer (via MLTestingRequest MOI). |

###### 7.3a.1.2.7.4 Notifications

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

### 7.3a.2 Information model definitions for ML emulation Phase

#### 7.3a.2.1 Class diagram

##### 7.3a.2.1.1 Relationships

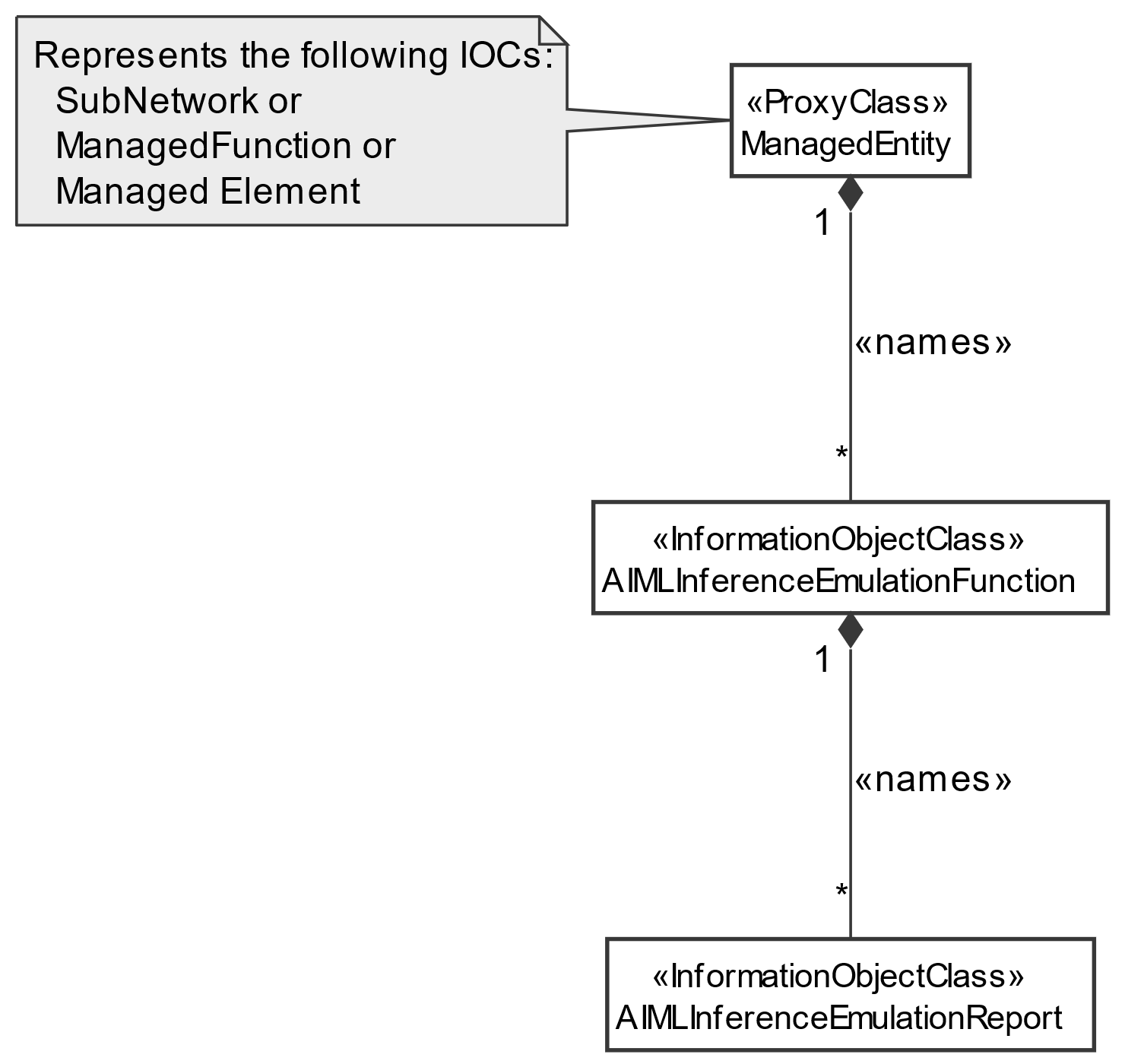


Figure 7.3a.2.1.1-1: NRM fragment for AI/ML inference emulation Control

##### 7.3a.2.1.2 Inheritance

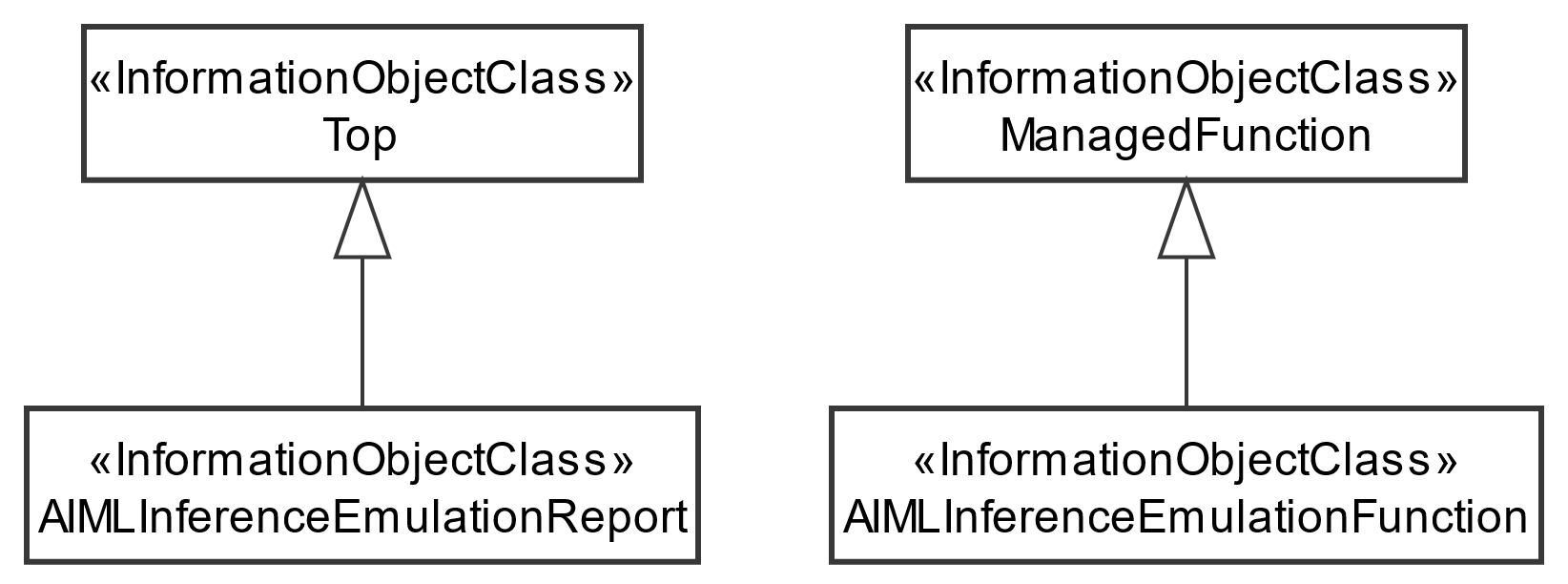


Figure 7.3a.2.1.2-1: AI/ML inference emulation Inheritance Relations

#### 7.3a.2.2 Class definitions

##### 7.3a.2.2.1 AIMLInferenceEmulationFunction

###### 7.3a.2.2.1.1 Definition

This IOC represents the properties of a function that undertakes AI/ML Inference Emulation. An AIMLInferenceEmulationFunction may be associated with one or more MLEntity(s). AIMLInferenceEmulationFunction is name contained with AIMLInferenceEmulationReport(s) that delivers the outcomes of the emulation processes.

NOTE: The way of triggering of an AI/ML inference emulation and the instantiation of the related AI/ML inference emulation process is not in the scope of the present document.

###### 7.3a.2.2.1.2 Attributes

The AIMLInferenceEmulationFunction IOC includes attributes inherited from ManagedFunction IOC (defined in TS 28.622[30]) and the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| aIMLInferenceEmulationFunctionId | M | T | F | F | F |
| **Attributes related to Role** |  |  |  |  |  |
| aIMLInferenceEmulationReportRef | M | T | F | F | F |

###### 7.3a.2.2.1.3 Attribute constraints

None.

###### 7.3a.2.2.1.4 Notifications

### The common notifications defined in clause 7.6 are valid for this IOC, without exceptions or additions.7.3a.3 Information model definitions for ML deployment phase

#### 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 deployment phase. For the UML semantics, see TS 32.156 [13].

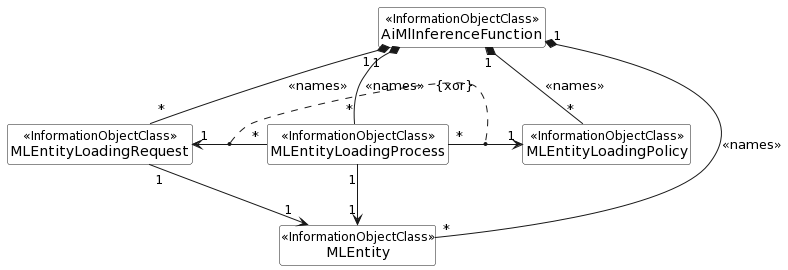


Figure 7.3a.3.1.1-1: NRM fragment for ML entity loading

##### 7.3a.3.1.2 Inheritance

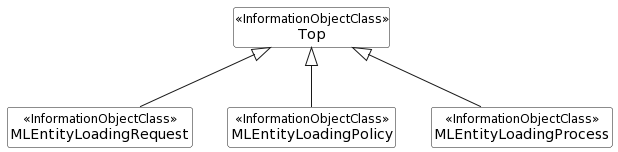


Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML entity loading related NRMs

#### 7.3a.3.2 Class definitions

##### 7.3a.3.2.1 MLEntityLoadingRequest

###### 7.3a.3.2.1.1 Definition

This IOC represents the ML entity loading request that is created by the MnS consumer. Using this IOC, the MnS consumer requests the MnS producer to load an ML entity to the target inference function.

This IOC has a requestStatus field to represent the status of the request:

- The attribute value is one of "NOT\_STARTED", "IN\_PROGRESS", "SUSPENDED", "FINISHED\_SUCCESS ", FINISHED\_FAILED" and "CANCELLED".

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

###### 7.3a.3.2.1.2 Attributes

Table 7.3a.3.2.1.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| requestStatus | M | T | T | F | T |
| cancelRequest | O | T | T | F | T |
| suspendRequest | O | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| mLEntityToLoadRef | M | T | F | F | T |

###### 7.3a.3.2.1.3 Attribute constraints

None.

###### 7.3a.3.2.1.4 Notifications

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

##### 7.3a.3.2.2 MLEntityLoadingPolicy

###### 7.3a.3.2.2.1 Definition

This IOC represents the ML entity loading policy set by the MnS consumer to the producer for loading an ML entity to the target inference function(s).

This IOC is used for the MnS consumer to set the conditions for the producer-initated ML entity loading. The MnS producer is only allowed to load the ML entity when all of the conditions are met.

###### 7.3a.3.2.2.2 Attributes

Table 7.3a.3.2.2.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| inferenceType | CM | T | T | F | T |
| policyForLoading | M | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| mLEntityRef | CM | T | F | F | F |

###### 7.3a.3.2.2.3 Attribute constraints

Table 7.3a.3.2.2.3-1

|  |  |
| --- | --- |
| Name | Definition |
| inferenceType Support Qualifier | Condition: The ML entity loading policy is related to an initially trained ML entity. |
| mLEntityRef Support Qualifier | Condition: The ML entity loading policy is related to a re-trained ML entity. |

###### 7.3a.3.2.2.4 Notifications

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

##### 7.3a.3.2.3 MLEntityLoadingProcess

###### 7.3a.3.2.3.1 Definition

This IOC represents the ML entity loading process.

For the consumer requested ML entity loading, one or more MLEntityLoadingProcess MOI(s) may be instantiated for each ML entity loading request presented by the MLEntityLoadingRequest MOI.

For the producer-initiated ML entity loading, one or more MLEntityLoadingProcess MOI(s) may be instantiated and associated with each MLEntityLoadingPolicy MOI.

One MLEntityLoadingProcess MOI represent the ML entity loading process(es) corresponding to one or more target inference function(s).

The "progressStatus" attribute represents the status of the ML entity loading process and includes information the MnS consumer can use to monitor the progress and results. The data type of this attribute is "ProcessMonitor" (see 3GPP TS 28.622 [12]). The following specializations are provided for this data type for the ML entity loading process:

- The "status" attribute values are "RUNNING", "CANCELLING", "SUSPENDED", "FINISHED", and "CANCELLED". The other values are not used.

- The "timer" attribute is not used.

- When the "status" is equal to "RUNNING" the "progressStateInfo" attribute shall indicate one of the following state: "LOADING".

- No specifications are provided for the "resultStateInfo" attribute. Vendor specific information may be provided though.

When the loading is completed with "status" equal to "FINISHED", the MnS producer creates the MOI(s) of loaded MLEntity under each MOI of the target inference function(s).

###### 7.3a.3.2.3.2 Attributes

Table 7.3a.3.2.3.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| progressStatus | M | T | F | F | T |
| cancelProcess | O | T | T | F | T |
| suspendProcess | O | T | T | F | T |
| resumeProcess | O | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
| MLEntityLoadingRequestRef | CM | T | F | F | T |
| MLEntityLoadingPolicyRef | CM | T | F | F | T |
| LoadedMLEntityRef | M | T | F | F | T |

###### 7.3a.3.2.3.3 Attribute constraints

Table 7.3a.3.2.3.3-1

|  |  |
| --- | --- |
| Name | Definition |
| MLEntityLoadingRequestRef Support Qualifier | Condition: The MLEntityLoadingProcess MOI is corresponding to the ML entity loading requested by the MnS consumer. |
| MLEntityLoadingPolicyRef Support Qualifier | Condition: The MLEntityLoadingProcess MOI is corresponding to the ML entity loading initiated by the MnS producer. |

###### 7.3a.3.2.3.4 Notifications

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

### 7.3a.4 Information model definitions for ML inference phase

#### 7.3a.4.1 Class diagram

##### 7.3a.4.1.1 Relationships

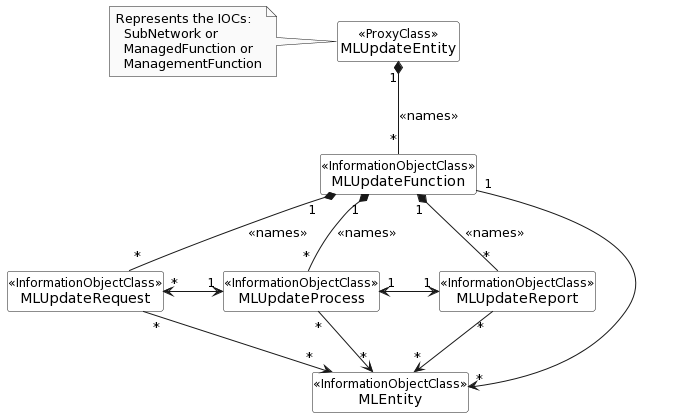
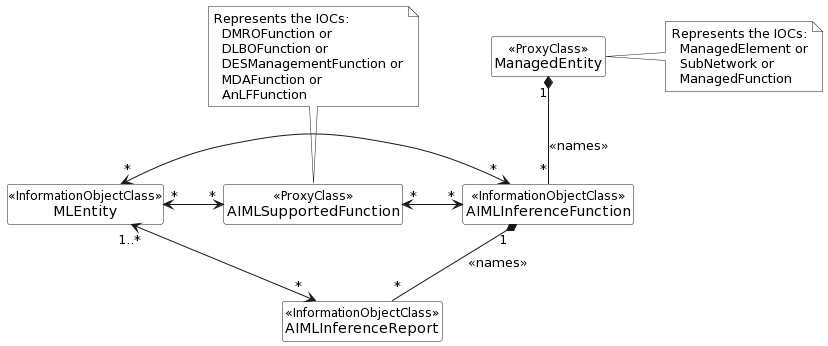


Figure 7.3a.4.1.1-1: NRM fragment for ML update



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

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

##### 7.3a.4.1.2 Inheritance

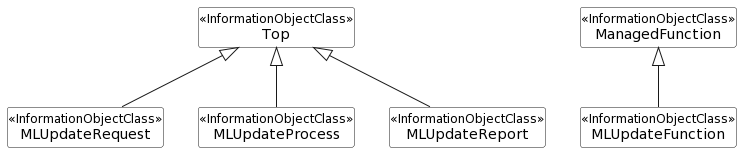


Figure 7.3a.4.1.2-1: Inheritance Hierarchy for ML update related NRMs

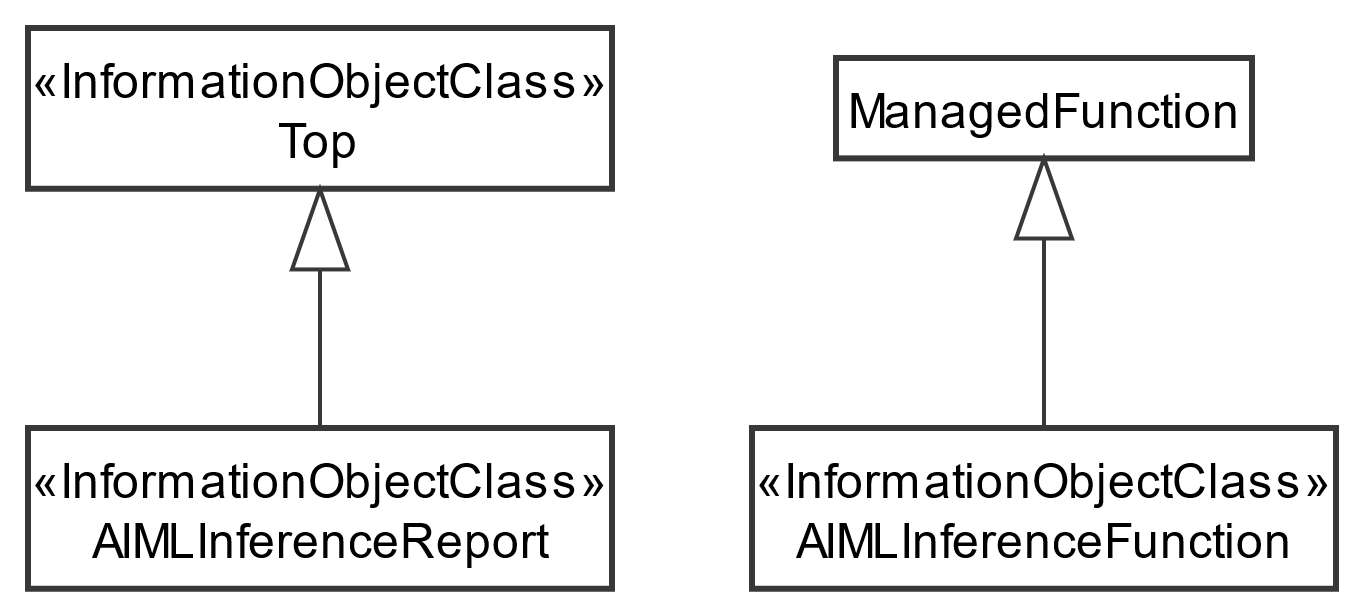


Figure 7.3a.4.1.2-2: Inheritance Hierarchy for AI/ML inference function

#### 7.3a.4.2 Class definitions

##### 7.3a.4.2.1 MLUpdateFunction

###### 7.3a.4.2.1.1 Definition

This IOC represents the function responsible for ML update.

The MOI of MLUpdateFunction is name-contained in an MOI of either a subnetwork, a managedFunction or a managementFunction.

The MLUpdateFunction is be associated with one or more ML entities.

The MLUpdateFunction contains one or more MLUpdateRequest(s)as well as one or more MLUpdateProcess(s), where an MLUpdateProcess is instantiated corresponding to one received MLUpdateRequest.

###### 7.3a.4.2.1.2 Attributes

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

Table 7.3a.4.2.1.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| availMLCapabilityReport | M | T | F | F | F |
| **Attributes related to Role** |  |  |  |  |  |
| mLEntityRef | M | T | F | F | F |

###### 7.3a.4.2.1.3 Attribute constraints

None.

###### 7.3a.4.2.1.4 Notifications

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

##### 7.3a.4.2.2 MLUpdateRequest

###### 7.3a.4.2.2.1 Definition

This IOC represents the properties of MLUpdateRequest.

For each request to update the ML capabilities, a consumer creates a new MOI of MLUpdateRequest on the MLUpdateFunction, i.e., MLUpdateRequest is instantiated for each request for updating ML capabilities:

- Each MLUpdateRequest is associated to at least one MLEntity

- Each MLUpdateRequest may have a RequestStatus field that is used to track the status of the specific MLUpdateRequest or the associated MLUpdateProcess. The RequestStatus is updated by MnS producer when there is a change in status of the update progress. The RequestStatus is an enumeration with the values: NOT\_STARTED, IN\_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED

- Each MLUpdateRequest may contain specific reporting requirements including an mLUpdateReportingPeriod that defines the time duration upon which the MnS consumer expects the ML update is reported. The reporting requirements contained in the MLUpdateRequest are mapped to an existing MLUpdateProcess instance.

- The MLUpdateRequest may specify a performanceGainThreshold which defines the minimum performance gain that shall be achieved with the capability update. This implies that the difference in the performances between the existing capabilities and the new capabilities needs to be at least performanceGainThreshold, otherwise the new capabilities shall not be applied. A threshold of performanceGainThreshold=0% implies that the capabilities should be applied even if there is no noticeable performance gain.

- The MLUpdateRequest may indicates the maximum time that should be taken to complete the update.

###### 7.3a.4.2.2.2 Attributes

The MLUpdateRequest IOC includes attributes inherited from Top IOC (defined in TS 28.622 [30]) and the following attributes:

Table 7.3a.4.2.2.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| performanceGainThreshold | O | T | T | T | F |
| newCapabilityVersionId | O | T | T | T | F |
| updateTimeDeadline | O | T | T | T | F |
| requestStatus | M | T | T | F | T |
| mLUpdateReportingPeriod | O | T | T | F | T |
| cancelRequest | O | T | T | F | T |
| suspendRequest | O | T | T | F | T |
| **Attributes related to Role** |  |  |  |  |  |
| mLUpdateProcessRef | M | T | F | F | F |
| mLEntityRef | M | T | F | F | F |

###### 7.3a.4.2.2.3 Attribute constraints

None.

###### 7.3a.4.2.2.4 Notifications

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

##### 7.3a.4.2.3 MLUpdateProcess

###### 7.3a.4.2.3.1 Definition

This IOC represents the ML update process.

For each MLUpdateRequest to update the ML capabilities, the MLUpdateProcess is instantiated for the MLUpdateRequest unless the MLUpdateRequest is associated with an ongoing MLUpdateProcess if the MLUpdateProcess is updating the same MLEntity(s) as stated in the MLUpdateRequest i.e., the MLUpdateProcess is associated with at least one MLUpdateRequest. Relatedly, the MLUpdateProcess is associated with at least one MLEntity.

- Each MLUpdateProcess may have a status attribute (i.e., progressStatus) used to indicate progress status of theupdate process.

- The MLUpdateProcess has the capability of compiling and delivering reports and notifications relating to the ML update request or process.

- Each MLUpdateProcess may have attributes specifying the ML capability update reporting characteristics (e.g. periodically, after completion, etc.).

###### 7.3a.4.2.3.2 Attributes

The MLUpdateProcess IOC includes attributes inherited from Top IOC (defined in TS 28.622 [30]) and the following attributes:

Table 7.3a.4.2.3.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| progressStatus | M | T | T | F | T |
| **Attributes related to Role** |  |  |  |  |  |
| mLEntityRef | M | T | F | F | F |
| MLUpdateRequestRef | M | T | F | F | F |
| MLUpdateReportRef | M | T | F | F | F |

###### 7.3a.4.2.3.3 Attribute constraints

None.

###### 7.3a.4.2.3.4 Notifications

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

##### 7.3a.4.2.4 MLUpdateReport

###### 7.3a.4.2.4.1 Definition

This IOC represents the properties of ML update report.

- The ML update process may generate one or more MLUpdateReport(s),

- Each MLUpdateReport is associated to one or more MLEntity(s) to indicate ML entities that have been updated.

- The MLUpdateReport may indicate the achieved performance gain for the specific ML capability update, which is the gain in performance of the new capabilities compared with the original capabilities.

- MLUpdateReport provides reports about MLEntity(s) or MLUpdateProcess(s) that themselves are associated with MLEntity(s) for which update is requested and/or executed. Correspondingly, both the MLUpdateRequest(s)and the MLUpdateProcess(s) are conditionally mandatory in that at least one of them must be associated with an instance of MLUpdateReport.

###### 7.3a.4.2.4.2 Attributes

Table 7.3a.4.2.4.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| UpdatedMLCapability | M | T | F | F | F |
| **Attributes related to Role** |  | | | | |
| mLEntityRef | M | T | F | F | F |
| mLUpdateProcessRef | M | T | F | F | F |

###### 7.3a.4.2.4.3 Attribute constraints

None.

###### 7.3a.4.2.4.4 Notifications

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

##### 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.

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 consumer.

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

NOTE: The IOCs representing the functions/functionalities (Note) that use the AI/ML inference function include MDAFunction, AnLFFunction, DMROFunction, DLBOFunction, 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.4.2.5.2 Attributes

Table 7.3a.4.2.5.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| activationStatus | M | T | T | F | T |
| managedActivationScope | O | T | T | F | T |
| **Attributes related to role** |  |  |  |  |  |
| usedByFunctionRefList | M | T | F | F | T |
| MLEntityRef | M | T | F | T | T |

###### 7.3a.4.2.5.3 Attribute constraints

None.

###### 7.3a.4.2.5.4 Notifications

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

##### 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 MLEntity.

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

###### 7.3a.4.2.6.2 Attributes

The AIMLInferenceReport includes inherited attributes from Top IOC (defined in TS28.622 [12] ) and the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| inferenceOutputs | M | T | F | F | T |
| **Attributes related to role** |  |  |  |  |  |
| mLEntityRef | 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.

## 7.4 Data type definitions

### 7.4.1 ModelPerformance <<dataType>>

#### 7.4.1.1 Definition

This data type specifies the performance of an ML entity when performing inference. The performance score is provided for each inference output.

#### 7.4.1.2 Attributes

Table 7.4.1.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| inferenceOutputName | M | T | T/F (NOTE) | F | T |
| performanceScore | M | T | T/F (NOTE) | F | T |
| performanceMetric | M | T | T/F (NOTE) | F | T |
| decisionConfidenceScore | O | T | F | F | T |
| NOTE: The isWritable qualifier is “T” if the attribute is used in MLTrainingRequest. The isWritable qualifier is "F" otherwise. | | | | | |

#### 7.4.1.3 Attribute constraints

None.

#### 7.4.1.4 Notifications

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

### 7.4.2 Void

### 7.4.3 MLContext <<dataType>>

#### 7.4.3.1 Definition

The MLContext represents the status and conditions related to the MLEntity. There are three types of context - the ExpectedRunTimeContext, the TrainingContext and the RunTimeContext, see clause 7.5.1 for details of each type.

#### 7.4.3.2 Attributes

Table 7.4.3.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| inferenceEntityRef | CM | T | F | F | F |
| dataProviderRef | M | T | F | F | F |

#### 7.4.3.3 Attribute constraints

Table 7.4.3.3-1

|  |  |
| --- | --- |
| Name | Definition |
| inferenceEntityRef Support Qualifier | Condition: The MLContext is used for ExpectedRunTimeContext, TrainingContext or RunTimeContext. |

#### 7.4.3.4 Notifications

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

### 7.4.4 SupportedPerfIndicator <<dataType>>

#### 7.4.4.1 Definition

This data type specifies a Performance indicator of an ML entity. The data type may be used to indicate which performance indicators shall be applicable to either of training, testing or inference.

#### 7.4.4.2 Attributes

Table 7.4.4.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| performanceIndicatorName | M | T | F | F | T |
| isSupportedForTraining | CM | T | F | F | T |
| isSupportedForTesting | CM | T | F | F | T |

#### 7.4.4.3 Attribute constraints

Table 7.4.4.3-1

|  |  |
| --- | --- |
| Name | Definition |
| isSupportedForTraining Support Qualifier | Condition: if the performance indicator named performanceIndicatorName is applicable for training, the isSupportedforTraining must be stated |
| isSupportedForTesting Support Qualifier | Condition: if the performance indicator named performanceIndicatorName is applicable for testing, the isSupportedForTesting must be stated |

#### 7.4.4.4 Notifications

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

### 7.4.5 AvailMLCapabilityReport <<dataType>>

#### 7.4.5.1 Definition

This dataType represents the the report of available ML capabilities following the update for specific ML capability(es).

- The ML update process may generate one or more AvailMLCapabilityReport(s), which indicate to the consumer that new ML capability(es) is/are available and can be applied.

- Each AvailMLCapabilityReport is associated to one or more MLEntity(s) and may indicate the one or more MLEntity(s) to which it applies.

- The AvailMLCapabilityReport may include CapabilityVersions which indicate that there are multiple candidate sets of available ML capabilities with a different version number for each set.

- The AvailMLCapabilityReport may include the expectedPerformanceGains, which provides information on the expected performance gain if/when the ML capabilities of the respective network function are updated with/to the specific set of newly available ML capabilities.

- associated to one or more MLEntity(s) and may indicate the one or more MLEntity(s) to which it applies.

#### 7.4.5.2 Attributes

The AvailMLCapabilityReport includes the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table 7.4.5.2-1Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| availMLCapabilityReportID | M | T | F | F | F |
| mLCapabilityVersionId | M | T | F | F | F |
| expectedPerformanceGains | O | T | F | F | F |
| **Attributes related to Role** |  |  |  |  |  |
| mLEntityRef | M | T | F | F | F |

#### 7.4.5.3 Attribute constraints

None.

#### 7.4.5.4 Notifications

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

### 7.4.6 AIMLManagementPolicy <<dataType>>

#### 7.4.6.1 Definition

This data type represents the properties of a policy for AI/ML management.

#### 7.4.6.2 Attributes

Table 7.4.6.2-1

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

#### 7.4.6.3 Attribute constraints

None.

#### 7.4.6.4 Notifications

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

### 7.4.7 ManagedActivationScope <<choice>>

#### 7.4.7.1 Definition

This <<choice>> defines the scopes for activating or deactivating the ML Inference function. It is a choice between the scopes parameter required for the activation or deactivation.

#### 7.4.7.2 Attributes

Table 7.4.7.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| CHOICE\_1.1 dNList | CM | T | T | F | T |
| CHOICE\_1.2 timeWindow | CM | T | T | F | T |
| CHOICE\_1.3 geoPolygon | CM | T | T | F | T |

#### 7.4.7.3 Attribute constraints

Table 7.4.7.3-1

|  |  |
| --- | --- |
| Name | Definition |
| dNList Support Qualifier CM | Condition: if the sub scope is per list of managed elements (e.g., DN list) |
| timeWindow Support Qualifier CM | Condition: if the sub scope is per list of time window. |
| geoPolygon Support Qualifier CM | Condition: if the sub scope is per list of GeoArea. |

#### 7.4.7.4 Notifications

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

### 7.4.8. MLCapabilityInfo <<dataType>>

#### 7.4.8.1. Definition

This dataType represents information about what the ML entity can make inference for. The inferenceOutputName is used as the identifier for the ML capability.

#### 7.4.8.2 Attributes

The MLCapabilityInfo <<dataType>> includes the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| inferenceType | M | T | F | F | T |
| capabilityName | O | T | F | F | T |
| mLCapabilityParameters | O | T | F | F | T |

#### 7.4.8.3 Attribute constraints

None.

#### 7.4.8.4 Notifications

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

### 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.

#### 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 |
| inferenceType | M | T | F | F | T |
| inferenceOutputTime | M | T | F | F | T |
| inferencePerformance | 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.

## 7.5 Attribute definitions

### 7.5.1 Attribute properties

Table 7.5.1-1

| Attribute Name | Documentation and Allowed Values | Properties |
| --- | --- | --- |
| mLEntityId | It identifies the ML entity.  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 |
| inferenceType | It indicates the type of inference that the ML model supports.  allowedValues: the values of the MDA type (see 3GPP TS 28.104 [2]), Analytics ID(s) of NWDAF (see 3GPP TS 23.288 [3]), types of inference for RAN, and vendor's specific extensions. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| areConsumerTrainingDataUsed | It indicates whether the consumer provided training data have been used for the ML model training.  allowedValues: ALL, PARTIALLY, NONE. | type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| usedConsumerTrainingData | It provides the address(es) where lists of the consumer-provided training data are located, which have been used for the ML model training.  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).  allowedValues: DN. | 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.  allowedValues: DN. | 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 training.  allowedValues: DN. | 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.  allowedValues: DN. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: True |
| 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 describes the entity that requested to instantiate the MLTrainingRequest MOI.  This attribute can be of type String or DN. | type: <<CHOICE>>  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLTrainingRequest.requestStatus | It describes the status of a particular ML 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 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 MLtraining MnS producer to terminate a specific training process.  allowedValues: MODEL UPDATED\_IN\_INFERENCE\_FUNCTION, INFERENCE FUNCTION\_TERMINATED, INFERENCE FUNCTION\_UPGRADED, INFERENCE\_CONTEXT\_CHANGED. | type: Enum  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 |
| mLEntityVersion | It indicates the version number of the ML entity.  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 entity 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 entity 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 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 MLEntity 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 entity.  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 entity, 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 entity 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 indicates whether the ML training MnS consumer cancels the ML training request.  Setting this attribute to "TRUE" cancels the ML 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTrainingRequest.suspendRequest | It indicates whether the ML training MnS consumer suspends the /ML training request.  Setting this attribute to "TRUE" suspends the ML training process. Suspension is possible when the requestStatus is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTrainingProcess.cancelProcess | It indicates whether the ML training MnS consumer cancels the ML training process.  Setting this attribute to "TRUE" cancels the ML training request. Cancellation is possible when the " mLTrainingProcess.progressStatus.status" is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTrainingProcess.suspendProcess | It indicates whether the ML training MnS consumer suspends the ML training process.  Setting this attribute to "TRUE" suspends the ML 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.  Default value is set to "FALSE".  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 entity 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 entity e.g. for training or inference | Type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| areNewTrainingDataUsed | It indicates whether the other new training data have been 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 MLEntity is expected to be applied.  allowedValues: N/A | Type: MLContext  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| trainingContext | This specify the context under which the MLEntity 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 entity is being applied.  allowedValues: N/A | Type: MLContext  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLEntityToTrainRef | It identifies the DN of the MLEntity requested to be trained.  allowedValues: DN | Type: DN  multiplicity: 0..1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLEnityGeneratedRef | It identifies the DN of the MLEntity generated by the ML training.  allowedValues: DN | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLEntityRepositoryRef | It identifies the DN of the MLEntityRepository. | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  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 entity when performing on the validation data.  allowedValues: N/A | type: ModelPerformance  multiplicity: \*  isOrdered: N/A  isUnique: N/A  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 |
| mLEntityIdList | It identifies a list of ML entities.  allowedValues: N/A. | type: String  multiplicity: \*  isOrdered: N/A  isUnique: True  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 indicates whether the ML testing MnS consumer cancels 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLTestingRequest.suspendRequest | It indicates whether the ML testing MnS consumer suspends 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| mLEntityToTestRef | It identifies the DN of the MLEntity requested to be tested.  allowedValues: DN | Type: DN  multiplicity: 0..1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| modelPerformanceTesting | It indicates the performance score of the ML entity when performing on the testing data.  allowedValues: N/A. | type: ModelPerformance  multiplicity: \*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLTestingResult | It provides the address where the testing result (including the inference result for each testing data example) is provided.  The detailed testing result format is vendor specific.  allowedValues: N/A. | type: String  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| testingRequestRef | It identifies the DN of the MLTestingRequest MOI.  allowedValues: DN | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| supportedPerformanceIndicators | This parameter lists specific PerformanceIndicator(s) of an ML entity.  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 training for the ML entity Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: FALSE  isNullable: False |
| isSupportedForTesting | It indicates whether the specific performance indicator is supported a performance metric of ML testing for the ML entity.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | type: Boolean  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: FALSE  isNullable: False |
| mLUpdateProcessRef | It is the DN of the mLUpdateProcess MOI that represents the process of updating an ML entity.  allowedValues: DN. | Type:  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLUpdateRequestRef | It is the DN of the MLUpdateRequest MOI that represents an  ML update request.  allowedValues: DN. | Type:  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| mLUpdateReportRef | It is the DN of the MLUpdateReport MOI that represents an ML update report.  allowedValues: DN. | Type:  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: False  isUnique: True  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 |
| 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: False  isUnique: True  defaultValue: None  isNullable: False |
| mLEntityRef | It indicates the list of references to MLEntity instances that can be updated. | Type: DN  multiplicity: 1 .. \*  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 indicates whether the MnS consumer cancels 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLUpdateRequest.suspendRequest | It indicates whether the MnS consumer suspends 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| memberMLEntityRefList | It identifies the list of member ML entities within a level of an ML entity coordination group.  allowedValues: DN list | Type: DN  multiplicity: 2..\*  isOrdered: True  isUnique: True  defaultValue: None  isNullable: False |
| mLEntityCoordinationGroupToTrainRef | It identifies the DN of the MlEntityCoordinationGroup requested to be trained.  allowedValues: DN | Type: DN  multiplicity: 0..1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLEnityCoordinationGroupGeneratedRef | It identifies the DN of the MlEntityCoordinationGroup generated by the ML training.  allowedValues: DN | Type: DN  multiplicity: 0..1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLEntityCoordinationGroupToTestRef | It identifies the DN of the MlEntityCoordinationGroup requested to be tested.  allowedValues: DN | Type: DN  multiplicity: 0..1  isOrdered: False  isUnique: True  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 MLEntity. | Type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| sourceTrainedMLEntityRef | It identifies the DN of the source trained MLEntity whose copy has been loaded from the ML entity repository to the inference function.  allowedValues: DN | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| MLEntityLoadingRequest.requestStatus | It describes the status of a particular ML entity 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 |
| MLEntityLoadingRequest.cancelRequest | It indicates whether the MnS consumer cancels the ML entity loading request.  Setting this attribute to "TRUE" cancels the ML entity 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLEntityLoadingRequest.suspendRequest | It indicates whether the MnS consumer suspends the ML entity loading request.  Setting this attribute to "TRUE" suspends the ML entity 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.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| mLEntityToLoadRef | It identifies the DN of a trained MLEntity requested to be loaded to the target inference function(s). | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| policyForLoading | It provides the policy for controlling ML entity loading triggered by the MnS producer.  This policy contains two thresholds in the thresholdList attribute. The first threshold is related to the ML entity to be loaded, and the second threshold is related to the existing ML entity being used for inference. | Type: AIMLManagementPolicy  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| thresholdList | It provides the list of threshold. | Type: ThresholdInfo  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| MLEntityLoadingProcess.progressStatus.progressStateInfo | It provides the following specialization for the "progressStateInfo" attribute of the "ProcessMonitor" data type for the "MLEntityLoadingProcess.progressStatus".  When the ML loading is in progress, and the " MLEntityLoadingProcess.progressStatus.status " is equal to "RUNNING", it provides the more detailed progress information.  allowedValues for " MLEntityLoadingProcess.progressStatus.status " = "RUNNING":  The allowed values for " MLEntityLoadingProcess.progressStatus.status " = "CANCELLING" are vendor specific.  The allowed values for " MLEntityLoadingProcess.progressStatus.status " = "NOT\_STARTED" are vendor specific. | Type: String  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| MLEntityLoadingProcess.cancelProcess | It indicates whether the MnS consumer cancels the ML entity loading process.  Setting this attribute to "TRUE" cancels the process. Cancellation is possible when the "MLEntityLoadingProcess.progressStatus.status" is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLEntityLoadingProcess.suspendProcess | It indicates whether the MnS consumer suspends the ML entity 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 "MLEntityLoadingProcess.progressStatus.status" is not the "FINISHED", "CANCELLING" or "CANCELLED" state. Setting the attribute to "FALSE" has no observable result.  Default value is set to "FALSE".  allowedValues: TRUE, FALSE. | Type: Boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: FALSE  isNullable: False |
| MLEntityLoadingRequestRef | It identifies the DN of the associated MLEntityLoadingRequest.  allowedValues: DN. | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| MLEntityLoadingPolicyRef | It identifies the DN of the associated MLEntityLoadingPolicyRef.  allowedValues: DN. | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| LoadedMLEntityRef | It identifies the DN of the MLEntity that has been loaded to the inference function.  allowedValues: DN | Type: DN  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: True |
| activationStatus | It describes the activation status.  allowedValues: ACTIVATED, DEACTIVATED. | Type: Enum  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| 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: False  isUnique: True  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 entity.  Each ML entity, inferenceOutputs may be a set of values.  allowedValues: N/A. | type: InferenceOutput  multiplicity:f 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| inferencePerformance | It indicates the performance score of the ML entity 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 |
| AIMLInferenceEmulationReportRefs | It indicates the DNs of set of reports generated on AIMLInferenceEmulationFunction. The AIMLInferenceEmulationReport has the same structure as the AIMLInferenceReport.  allowedValues: N/A. | type: DN of AIMLInferenceReport  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| mLCapabilitiesInfoList | It indicates information about what an ML entity 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 entity can generate inference.  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 inferenceType and capabilityName.  allowedValues: N/A | Type: AttributeValuePair  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| NOTE: When the performanceScore is to indicate the performance score for ML 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 testing, the data set is the testing data set. | | |

### 7.5.2 Constraints

None.

## 7.6 Common notifications

### 7.6.1 Configuration notifications

This clause presents a list of notifications, defined in 3GPP TS 28.532 [11], that an MnS consumer may receive. The notification header attribute objectClass/objectInstance shall capture the DN of an instance of a class defined in the present document.

Table 7.6.1-1

| Name | Qualifier | Notes |
| --- | --- | --- |
| notifyMOICreation | O | -- |
| notifyMOIDeletion | O | -- |
| notifyMOIAttributeValueChanges | O | -- |
| notifyEvent | O | -- |

# 8 Service components

## 8.1 Service components for ML model training MnS

The components for ML model training MnS are listed in table 8.1-1.

Table 8.1-1: Components for ML model training MnS

|  |  |  |
| --- | --- | --- |
| Management service component type A | Management service component type B | Management service component type C |
| The operations and notifications for generic provisioning management service (see clause 11.1.1 of 3GPP TS 28.532 [11]). | MLTrainingFunction IOC; MLTrainingRequest IOC;  MLTrainingReport IOC;  MLTrainingProcess IOC. | N/A |

# 9 Solution Set (SS)

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

- YAML based Solution Set (Annex B).

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.2 PlantUML code for Figure 7.3a.1.1.1-1: NRM fragment for ML model training

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

skinparam nodesep 60

class ManagedEntity <<ProxyClass>>

class MLEntity <<InformationObjectClass>>

class MLEntityCoordinationGroup <<InformationObjectClass>>

class MLTrainingFunction <<InformationObjectClass>>

class MLTrainingRequest <<InformationObjectClass>>

class MLTrainingReport <<InformationObjectClass>>

class MLTrainingProcess <<InformationObjectClass>>

class MLEntityRepository <<InformationObjectClass>>

class ThresholdMonitor <<InformationObjectClass>>

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

MLTrainingFunction "1" \*-- "\*" MLTrainingProcess: <<names>>

MLTrainingFunction "1" \*-- "\*" MLTrainingRequest: <<names>>

MLTrainingFunction "1" \*-- "\*" MLTrainingReport: <<names>>

'SubNetwork "1" \*-- "\*" MLEntityRepository: <<names>>

MLEntityRepository "1" \*-- "\*" MLEntity : <<names>>

MLTrainingFunction "1" \*-- "\*" ThresholdMonitor : <<names>>

MLTrainingFunction "\*" -l-> "\*" MLEntityRepository

MLTrainingProcess "1" <-r-> "1" MLTrainingReport

MLTrainingReport "1" --> "1" MLTrainingReport

MLTrainingProcess "\*" -l-> "\*" MLTrainingRequest

MLTrainingRequest "\*" --> "0..1" MLEntity

MLTrainingRequest "\*" --> "0..1" MLEntityCoordinationGroup

MLTrainingReport "\*" --> "1" MLEntity

MLTrainingReport "\*" --> "1" MLEntityCoordinationGroup

MLEntityCoordinationGroup "\*" --> "2..\*" MLEntity

MLEntity"\*" -u-> "1" ThresholdMonitor

(MLTrainingReport, MLEntity) ... (MLTrainingReport, MLEntityCoordinationGroup) : {xor}

(MLTrainingRequest, MLEntity) ... (MLTrainingRequest, MLEntityCoordinationGroup) : {xor}

note left of ManagedEntity

This represents the following IOCs:

SubNetwork or

ManagedFunction or

ManagedElement

end note

@enduml

# A.3 PlantUML code for Figure 7.3a.1.1.2-1: Inheritance Hierarchy for ML model training related NRMs

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class Top <<InformationObjectClass>>

class ManagedFunction <<InformationObjectClass>>

class MLTrainingFunction <<InformationObjectClass>>

class MLTrainingRequest <<InformationObjectClass>>

class MLTrainingProcess <<InformationObjectClass>>

class MLTrainingReport <<InformationObjectClass>>

ManagedFunction <|-- MLTrainingFunction

Top <|-- MLTrainingRequest

Top <|-- MLTrainingProcess

Top <|-- MLTrainingReport

@enduml

# A.4 PlantUML code for Figure 7.2a.1.2-1: Inheritance Hierarchy for common information models for AI/ML management

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class Top <<InformationObjectClass>>

class MLEntityRepository <<InformationObjectClass>>

class MLEntity <<InformationObjectClass>>

class MLEntityCoordinationGroup <<InformationObjectClass>>

Top <|-- MLEntityRepository

Top <|-- MLEntity

Top <|-- MLEntityCoordinationGroup

@enduml

# A.5 PlantUML code for Figure 7.2a.1.1-1: Relationships for common information models for AI/ML management

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

skinparam nodesep 60

class MLEntityRepository <<InformationObjectClass>>

class MLEntity <<InformationObjectClass>>

class MLEntityCoordinationGroup <<InformationObjectClass>>

class SubNetwork <<InformationObjectClass>>

MLEntityRepository "1" \*-- "\*" MLEntity: <<names>>

MLEntityRepository "1" \*-- "\*" MLEntityCoordinationGroup: <<names>>

SubNetwork"1" \*-- "\*" MLEntityRepository : <<names>>

MLEntityCoordinationGroup "\*" -r-> "2..\*" MLEntity

@enduml

# A.6 PlantUML code for Figure 7.3a.1.1.1-2: NRM fragment for ML entity testing

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class MLTestingEntity <<ProxyClass>>

class TestingFunction <<ProxyClass>>

class MLEntity <<InformationObjectClass>>

class MLEntityCoordinationGroup <<InformationObjectClass>>

class MLTestingFunction <<InformationObjectClass>>

class MLTestingRequest <<InformationObjectClass>>

class MLTestingReport <<InformationObjectClass>>

MLTestingEntity "1" \*-- "\*" MLTestingFunction: <<names>>

TestingFunction "1" \*-- "\*" MLTestingRequest : <<names>>

TestingFunction "1" \*-- "\*" MLTestingReport : <<names>>

MLTestingRequest "\*" --> "0..1" MLEntity

MLTestingRequest "\*" --> "0..1" MLEntityCoordinationGroup

MLTestingReport "\*" -l-> "1" MLTestingRequest

(MLTestingRequest, MLEntity) ... (MLTestingRequest, MLEntityCoordinationGroup) : {xor}

note left of MLTestingEntity

Represents the following IOCs:

Subnetwork or

ManagedFunction or

ManagedElement

end note

note left of TestingFunction

Represents the following IOCs:

MLTestingFunction or

MLTrainingFunction

end note

@enduml

# A.7 PlantUML code for Figure 7.3a.1.1.2-2: Inheritance Hierarchy for ML entity testing related NRMs

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class Top <<InformationObjectClass>>

class ManagedFunction <<InformationObjectClass>>

class MLTestingFunction <<InformationObjectClass>>

class MLTestingRequest <<InformationObjectClass>>

class MLTestingReport <<InformationObjectClass>>

ManagedFunction <|-- MLTestingFunction

Top <|-- MLTestingRequest

Top <|-- MLTestingReport

@enduml

# A.8 PlantUML code for Figure 7.3a.4.1.1-1: NRM fragment for ML update

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

skinparam nodesep 60

class MLUpdateEntity <<ProxyClass>>

class MLUpdateFunction <<InformationObjectClass>>

class MLUpdateRequest <<InformationObjectClass>>

class MLUpdateProcess <<InformationObjectClass>>

class MLUpdateReport <<InformationObjectClass>>

class MLEntity <<InformationObjectClass>>

MLUpdateEntity "1" \*-- "\*" MLUpdateFunction:<<names>>

MLUpdateFunction "1" \*-- "\*" MLUpdateRequest:<<names>>

MLUpdateFunction "1" \*-- "\*" MLUpdateProcess:<<names>>

MLUpdateFunction "1" \*-- "\*" MLUpdateReport:<<names>>

MLUpdateFunction "1" --> "\*" "MLEntity"

MLUpdateRequest "\*" <-r-> "1" "MLUpdateProcess"

MLUpdateProcess "1" <-r-> "1" "MLUpdateReport"

MLUpdateProcess "\*" --> "\*" "MLEntity"

MLUpdateReport "\*" --> "\*" "MLEntity"

MLUpdateRequest "\*" --> "\*" "MLEntity"

note left of MLUpdateEntity

Represents the IOCs:

SubNetwork or

ManagedFunction or

ManagementFunction

end note

@enduml

# A.9 PlantUML code for Figure 7.3a.4.1.2-1: Inheritance Hierarchy for ML update related NRMs

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class Top <<InformationObjectClass>>

class ManagedFunction <<InformationObjectClass>>

class MLUpdateFunction <<InformationObjectClass>>

class MLUpdateRequest <<InformationObjectClass>>

class MLUpdateProcess <<InformationObjectClass>>

class MLUpdateReport <<InformationObjectClass>>

ManagedFunction <|-- MLUpdateFunction

Top <|-- MLUpdateRequest

Top <|-- MLUpdateProcess

Top <|-- MLUpdateReport

@enduml

# A.10 PlantUML code for Figure 7.3a.3.1.1-1: NRM fragment for ML entity loading

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class AiMlInferenceFunction <<InformationObjectClass>>

class MLEntity <<InformationObjectClass>>

class MLEntityLoadingRequest <<InformationObjectClass>>

class MLEntityLoadingPolicy <<InformationObjectClass>>

class MLEntityLoadingProcess <<InformationObjectClass>>

AiMlInferenceFunction "1" \*-- "\*" MLEntityLoadingRequest : <<names>>

AiMlInferenceFunction "1" \*-- "\*" MLEntityLoadingPolicy : <<names>>

AiMlInferenceFunction "1" \*-- "\*" MLEntityLoadingProcess : <<names>>

MLEntityLoadingRequest "1" <-r- "\*" MLEntityLoadingProcess

MLEntityLoadingProcess "\*" -r-> "1" MLEntityLoadingPolicy

MLEntityLoadingRequest "1" --> "1" MLEntity

MLEntityLoadingProcess "1" --> "1" MLEntity

AiMlInferenceFunction "1" \*-- "\*" MLEntity : <<names>>

(MLEntityLoadingProcess, MLEntityLoadingRequest) ... (MLEntityLoadingProcess, MLEntityLoadingPolicy) : {xor}

@enduml

# A.11 PlantUML code for Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML entity loading related NRMs

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class Top <<InformationObjectClass>>

class MLEntityLoadingRequest <<InformationObjectClass>>

class MLEntityLoadingPolicy <<InformationObjectClass>>

class MLEntityLoadingProcess <<InformationObjectClass>>

Top <|-- MLEntityLoadingRequest

Top <|-- MLEntityLoadingPolicy

Top <|-- MLEntityLoadingProcess

@enduml

# 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 MLEntity <<InformationObjectClass>>

class ManagedEntity <<ProxyClass>>

class AIMLSupportedFunction <<ProxyClass>>

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

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

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

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

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

MLEntity "1..\*" <--> "\*" AIMLInferenceReport

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

end note

@enduml

# A.13 PlantUML code for Figure 7.3a.4.1.2-2: Inheritance Hierarchy 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

class Top <<InformationObjectClass>>

class AIMLInferenceFunction << InformationObjectClass >>

class AIMLInferenceReport <<InformationObjectClass>>

ManagedFunction <|-- AIMLInferenceFunction

Top <|-- AIMLInferenceReport

@enduml

# A.14 PlantUML code for Figure 7.3a.2.1.1-1: NRM fragment for AI/ML inference emulation Control

@startuml

scale max 350 height

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class ManagedEntity <<ProxyClass>>

class AIMLInferenceEmulationFunction <<InformationObjectClass>>

class AIMLInferenceEmulationReport << InformationObjectClass >>

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

AIMLInferenceEmulationFunction "1" \*-- "\*" AIMLInferenceEmulationReport : <<names>>

note left of ManagedEntity

Represents the following IOCs:

SubNetwork or

ManagedFunction or

Managed Element

end note

@enduml

# A.15 PlantUML code for Figure 7.3a.2.1.2-1: AI/ML inference emulation Inheritance Relations

@startuml

skinparam ClassStereotypeFontStyle normal

skinparam ClassBackgroundColor White

skinparam shadowing false

skinparam monochrome true

hide members

hide circle

'skinparam maxMessageSize 250

class Top <<InformationObjectClass>>

class ManagedFunction <<InformationObjectClass>>

class AIMLInferenceEmulationFunction << InformationObjectClass >>

class AIMLInferenceEmulationReport << InformationObjectClass >>

ManagedFunction <|-- AIMLInferenceEmulationFunction

Top <|-- AIMLInferenceEmulationReport

@enduml

Annex B (normative):  
OpenAPI definition of the AI/ML NRM

# B.1 General

This annex contains the OpenAPI definition of the AI/ML NRM in YAML format.

The information models of the AI/ML NRM are defined in clause 7.

Mapping rules to produce the OpenAPI definition based on the information model are defined in 3GPP TS 32.160 [14].

# B.2 Solution Set (SS) definitions

## B.2.1 OpenAPI document "TS28105\_AiMlNrm.yaml"

<CODE BEGINS>

openapi: 3.0.1

info:

title: AI/ML NRM

version: 18.3.0

description: >-

OAS 3.0.1 specification of the AI/ML NRM

© 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).

All rights reserved.

externalDocs:

description: 3GPP TS 28.105; AI/ML Management

url: http://www.3gpp.org/ftp/Specs/archive/28\_series/28.105/

paths: {}

components:

schemas:

#-------- Definition of types-----------------------------------------------------

MLContext:

type: object

properties:

inferenceEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

dataProviderRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

RequestStatus:

type: string

enum:

- NOT\_STARTED

- IN\_PROGRESS

- SUSPENDED

- FINISHED

- CANCELLED

- CANCELLING

ModelPerformance:

type: object

properties:

inferenceOutputName:

type: string

performanceMetric:

type: string

performanceScore:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Float'

decisionConfidenceScore:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Float'

ProcessMonitor:

description: >-

This data type is the "ProcessMonitor" data type defined in âœgenericNrm.yamlâ

with specialisations for usage in TS 28.105.

type: object

properties:

status:

type: string

progressPercentage:

type: integer

minimum: 0

maximum: 100

progressStateInfo:

type: string

resultStateInfo:

type: string

AIMLManagementPolicy:

description: >-

This data type represents the properties of a policy for AI/ML management.

type: object

properties:

thresholdList:

type: array

items:

$ref: 'TS28623\_ThresholdMonitorNrm.yaml#/components/schemas/ThresholdInfo'

SupportedPerfIndicator:

type: object

properties:

performanceIndicatorName:

type: string

isSupportedForTraining:

type: boolean

isSupportedForTesting:

type: boolean

ManagedActivationScope:

oneOf:

- type: object

properties:

dNList:

type: array

items:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

- type: object

properties:

timeWindow:

type: array

items:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/TimeWindow'

- type: object

properties:

geoPolygon:

type: array

items:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/GeoArea'

MLCapabilityInfo:

type: object

properties:

inferenceType:

type: string

capabilityName:

type: string

mLCapabilityParameters:

description: A map (list of key-value pairs) for an inferenceType and capabilityName

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/AttributeNameValuePairSet'

AvailMLCapabilityReport:

type: object

properties:

mLCapabilityVersionId:

type: array

items:

type: string

expectedPerformanceGains:

type: array

items:

$ref: '#/components/schemas/ModelPerformance'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

InferenceOutput:

type: object

properties:

inferenceOutputId:

type: array

items:

type: string

inferenceType:

type: string

inferenceOutputTime:

type: array

items:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DateTime'

# FIXME, isOrder/isUnique both as True

inferencePerformance:

$ref: '#/components/schemas/ModelPerformance'

outputResult:

description: A map (list of key-value pairs) for Inference result name and it's value

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/AttributeNameValuePairSet'

#-------- Definition of types for name-containments ------

SubNetwork-ncO-AiMlNrm:

type: object

properties:

MLTrainingFunction:

$ref: '#/components/schemas/MLTrainingFunction-Multiple'

MLTestingFunction:

$ref: '#/components/schemas/MLTestingFunction-Multiple'

MLEntityRepository:

$ref: '#/components/schemas/MLEntityRepository-Multiple'

MLUpdateFunction:

$ref: '#/components/schemas/MLUpdateFunction-Multiple'

AIMLInferenceFunction:

$ref: '#/components/schemas/AIMLInferenceFunction-Multiple'

ManagedElement-ncO-AiMlNrm:

type: object

properties:

MLTrainingFunction:

$ref: '#/components/schemas/MLTrainingFunction-Multiple'

MLTestingFunction:

$ref: '#/components/schemas/MLTestingFunction-Multiple'

MLEntityRepository:

$ref: '#/components/schemas/MLEntityRepository-Multiple'

MLUpdateFunction:

$ref: '#/components/schemas/MLUpdateFunction-Multiple'

AIMLInferenceFunction:

$ref: '#/components/schemas/AIMLInferenceFunction-Multiple'

#-------- Definition of concrete IOCs --------------------------------------------

MLTrainingFunction-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-Attr'

- type: object

properties:

mLEntityRepositoryRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-ncO'

- type: object

properties:

MLTrainingRequest:

$ref: '#/components/schemas/MLTrainingRequest-Multiple'

MLTrainingProcess:

$ref: '#/components/schemas/MLTrainingProcess-Multiple'

MLTrainingReport:

$ref: '#/components/schemas/MLTrainingReport-Multiple'

ThresholdMonitors:

$ref: 'TS28623\_ThresholdMonitorNrm.yaml#/components/schemas/ThresholdMonitor-Multiple'

MLTrainingRequest-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

inferenceType:

type: string

candidateTrainingDataSource:

type: array

items:

type: string

trainingDataQualityScore:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Float'

trainingRequestSource:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

requestStatus:

$ref: '#/components/schemas/RequestStatus'

expectedRuntimeContext:

$ref: '#/components/schemas/MLContext'

performanceRequirements:

type: array

items:

$ref: '#/components/schemas/ModelPerformance'

cancelRequest:

type: boolean

suspendRequest:

type: boolean

mLEntityToTrainRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEntityCoordinationGroupToTrainRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLTrainingProcess-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

priority:

type: integer

terminationConditions:

type: string

enum:

- UPDATED\_IN\_INFERENCE\_FUNCTION

- INFERENCE FUNCTION\_TERMINATED

- INFERENCE FUNCTION\_UPGRADED

- INFERENCE\_CONTEXT\_CHANGED

progressStatus:

$ref: '#/components/schemas/ProcessMonitor'

cancelProcess:

type: boolean

suspendProcess:

type: boolean

trainingRequestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

trainingReportRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

MLTrainingReport-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

areConsumerTrainingDataUsed:

type: string

enum:

- ALL

- PARTIALLY

- NONE

usedConsumerTrainingData:

type: array

items:

type: string

modelconfidenceIndication:

type: integer

modelPerformanceTraining:

type: array

items:

$ref: '#/components/schemas/ModelPerformance'

modelPerformanceValidation:

type: array

items:

$ref: '#/components/schemas/ModelPerformance'

dataRatioTrainingAndValidation:

type: integer

areNewTrainingDataUsed:

type: boolean

trainingRequestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

trainingProcessRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

lastTrainingRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEnityGeneratedRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEntityCoordinationGroupGeneratedRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

MLTestingFunction-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-Attr'

- type: object

properties:

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-ncO'

- type: object

properties:

MLTestingRequest:

$ref: '#/components/schemas/MLTestingRequest-Multiple'

MLTestingReport:

$ref: '#/components/schemas/MLTestingReport-Multiple'

MLTestingRequest-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

requestStatus:

$ref: '#/components/schemas/RequestStatus'

cancelRequest:

type: boolean

suspendRequest:

type: boolean

mLEntityToTestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEntityCoordinationGroupToTestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLTestingReport-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

modelPerformanceTesting:

type: array

items:

$ref: '#/components/schemas/ModelPerformance'

mLTestingResult:

type: string

testingRequestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLEntityLoadingRequest-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

requestStatus:

$ref: '#/components/schemas/RequestStatus'

cancelRequest:

type: boolean

suspendRequest:

type: boolean

mLEntityToLoadRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLEntityLoadingPolicy-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

inferenceType:

type: string

policyForLoading:

$ref: '#/components/schemas/AIMLManagementPolicy'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

MLEntityLoadingProcess-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

progressStatus:

$ref: '#/components/schemas/ProcessMonitor'

cancelProcess:

type: boolean

suspendProcess:

type: boolean

resumeProcess:

type: boolean

MLEntityLoadingRequestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLEntityLoadingPolicyRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

LoadedMLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLEntity-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

type: object

properties:

mLEntityId:

type: string

inferenceType:

type: string

mLEntityVersion:

type: string

expectedRunTimeContext:

$ref: '#/components/schemas/MLContext'

trainingContext:

$ref: '#/components/schemas/MLContext'

runTimeContext:

$ref: '#/components/schemas/MLContext'

supportedPerformanceIndicators:

$ref: '#/components/schemas/SupportedPerfIndicator'

mLCapabilitiesInfoList:

type: array

items:

$ref: '#/components/schemas/MLCapabilityInfo'

retrainingEventsMonitorRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

sourceTrainedMLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLEntityRepository-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

type: object

properties:

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

- type: object

properties:

MLEntity:

$ref: '#/components/schemas/MLEntity-Multiple'

MLEntityCoordinationGroup:

$ref: '#/components/schemas/MLEntityCoordinationGroup-Multiple'

MLEntityCoordinationGroup-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

type: object

properties:

memberMLEntityRefList:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

## 7.3a.4.1 IOC

MLUpdateFunction-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-Attr'

- type: object

properties:

availMLCapabilityReport:

$ref: '#/components/schemas/AvailMLCapabilityReport'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-ncO'

- type: object

properties:

MLUpdateRequest:

$ref: '#/components/schemas/MLUpdateRequest-Multiple'

MLUpdateProcess:

$ref: '#/components/schemas/MLUpdateProcess-Multiple'

MLUpdateReport:

$ref: '#/components/schemas/MLUpdateReport-Multiple'

MLUpdateRequest-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

type: object

properties:

performanceGainThreshold:

type: array

items:

$ref: '#/components/schemas/ModelPerformance'

newCapabilityVersionId:

type: array

items:

type: string

updateTimeDeadline:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/TimeWindow'

requestStatus:

$ref: '#/components/schemas/RequestStatus'

mLUpdateReportingPeriod:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/TimeWindow'

cancelRequest:

type: boolean

suspendRequest:

type: boolean

mLUpdateProcessRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

MLUpdateProcess-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

type: object

properties:

progressStatus:

$ref: '#/components/schemas/ProcessMonitor'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

mLUpdateRequestRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

mLUpdateReportRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

MLUpdateReport-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

type: object

properties:

updatedMLCapability:

$ref: '#/components/schemas/AvailMLCapabilityReport'

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

mLUpdateProcessRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/Dn'

AIMLInferenceFunction-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-Attr'

- type: object

properties:

activationStatus:

type: string

enum:

- ACTIVATED

- DEACTIVATED

managedActivationScope:

$ref: '#/components/schemas/ManagedActivationScope'

usedByFunctionRefList:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

mLEntityRef: # FIXME S5-240805,S5-240917 both define here

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-ncO'

- type: object

properties:

AIMLInferenceReport:

$ref: '#/components/schemas/AIMLInferenceReport-Multiple'

AIMLInferenceReport-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- type: object

properties:

inferenceOutputs: #stage 2: attribute table name as: aimlInferenceOutputs FIXME

type: array

items:

$ref: '#/components/schemas/InferenceOutput'

minItems: 1

mLEntityRef:

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

AIMLInferenceEmulationFunction-Single:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/Top'

- type: object

properties:

attributes:

allOf:

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-Attr'

- type: object

properties:

AIMLInferenceEmulationReportRefs: # FIXME stage 2 of IOC AIMLInferenceEmulationReport missing

$ref: 'TS28623\_ComDefs.yaml#/components/schemas/DnList'

- $ref: 'TS28623\_GenericNrm.yaml#/components/schemas/ManagedFunction-ncO'

#-------- Definition of JSON arrays for name-contained IOCs ----------------------

MLTrainingFunction-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTrainingFunction-Single'

MLTrainingRequest-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTrainingRequest-Single'

MLTrainingProcess-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTrainingProcess-Single'

MLTrainingReport-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTrainingReport-Single'

MLEntity-Multiple:

type: array

items:

$ref: '#/components/schemas/MLEntity-Single'

MLEntityRepository-Multiple:

type: array

items:

$ref: '#/components/schemas/MLEntityRepository-Single'

MLEntityCoordinationGroup-Multiple:

type: array

items:

$ref: '#/components/schemas/MLEntityCoordinationGroup-Single'

MLTestingFunction-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTestingFunction-Single'

MLTestingRequest-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTestingRequest-Single'

MLTestingReport-Multiple:

type: array

items:

$ref: '#/components/schemas/MLTestingRequest-Single'

MLEntityLoadingRequest-Multiple:

type: array

items:

$ref: '#/components/schemas/MLEntityLoadingRequest-Single'

MLEntityLoadingProcess-Multiple:

type: array

items:

$ref: '#/components/schemas/MLEntityLoadingProcess-Single'

MLEntityLoadingPolicy-Multiple:

type: array

items:

$ref: '#/components/schemas/MLEntityLoadingPolicy-Single'

MLUpdateFunction-Multiple:

type: array

items:

$ref: '#/components/schemas/MLUpdateFunction-Single'

MLUpdateRequest-Multiple:

type: array

items:

$ref: '#/components/schemas/MLUpdateRequest-Single'

MLUpdateProcess-Multiple:

type: array

items:

$ref: '#/components/schemas/MLUpdateProcess-Single'

MLUpdateReport-Multiple:

type: array

items:

$ref: '#/components/schemas/MLUpdateReport-Single'

AIMLInferenceFunction-Multiple:

type: array

items:

$ref: '#/components/schemas/AIMLInferenceFunction-Single'

AIMLInferenceReport-Multiple:

type: array

items:

$ref: '#/components/schemas/AIMLInferenceReport-Single'

AIMLInferenceEmulationFunction-Multiple:

type: array

items:

$ref: '#/components/schemas/AIMLInferenceEmulationFunction-Single'

#-------- Definitions in TS 28.104 for TS 28.532 ---------------------------------

resources-AiMlNrm:

oneOf:

- $ref: '#/components/schemas/MLTrainingFunction-Single'

- $ref: '#/components/schemas/MLTrainingRequest-Single'

- $ref: '#/components/schemas/MLTrainingProcess-Single'

- $ref: '#/components/schemas/MLTrainingReport-Single'

- $ref: '#/components/schemas/MLEntity-Single'

- $ref: '#/components/schemas/MLEntityRepository-Single'

- $ref: '#/components/schemas/MLEntityCoordinationGroup-Single'

- $ref: '#/components/schemas/MLTestingFunction-Single'

- $ref: '#/components/schemas/MLTestingRequest-Single'

- $ref: '#/components/schemas/MLTestingReport-Single'

- $ref: '#/components/schemas/MLEntityLoadingRequest-Single'

- $ref: '#/components/schemas/MLEntityLoadingProcess-Single'

- $ref: '#/components/schemas/MLEntityLoadingPolicy-Single'

- $ref: '#/components/schemas/MLUpdateFunction-Single'

- $ref: '#/components/schemas/MLUpdateRequest-Single'

- $ref: '#/components/schemas/MLUpdateProcess-Single'

- $ref: '#/components/schemas/MLUpdateReport-Single'

- $ref: '#/components/schemas/AIMLInferenceFunction-Single'

- $ref: '#/components/schemas/AIMLInferenceReport-Single'

- $ref: '#/components/schemas/AIMLInferenceEmulationFunction-Single'

<CODE ENDS>

Annex C (informative):  
Change history

| **Change history** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2022-06 | SA#96 |  |  |  |  | Upgrade to change control version | 17.0.0 |
| 2022-09 | SA#97e | SP-220851 | 0003 | - | F | Corrections to the terms and definition description and corresponding updates | 17.1.0 |
| 2022-09 | SA#97e | SP-220850 | 0004 | 1 | F | fix incorrect yaml file name in TS28.105 | 17.1.0 |
| 2022-09 | SA#97e | SP-220851 | 0005 | 1 | F | Clarifications and corrections of Use cases | 17.1.0 |
| 2022-09 | SA#97e | SP-220851 | 0006 | 1 | F | Clarifications and corrections into the Class definitions and Attribute properties | 17.1.0 |
| 2022-09 | SA#97e | SP-220851 | 0007 | 1 | F | Correction and clarifications of the Requirements | 17.1.0 |
| 2022-09 | SA#97e |  |  |  |  | Alignment with content with FORGE | 17.1.1 |
| 2022-12 | SA#98e | SP-221166 | 0008 | 2 | F | Adding missing attributes | 17.2.0 |
| 2022-12 | SA#98e | SP-221166 | 0009 | - | F | Correction of stage 3 openAPI | 17.2.0 |
| 2023-03 | SA#99 | SP-230193 | 0011 | - | F | Adding the missing definition of attributes Stage 2 and Stage 3 | 17.3.0 |
| 2023-03 | SA#99 | SP-230193 | 0013 | 1 | F | Correcting the attribute properties | 17.3.0 |
| 2023-03 | SA#99 | SP-230193 | 0014 | 1 | F | Correction of the Handling errors in data and ML decisions | 17.3.0 |
| 2023-03 | SA#99 | SP-230193 | 0015 | 1 | F | Correction of terminologies | 17.3.0 |
| 2023-03 | SA#99 | SP-230193 | 0017 | 1 | F | Correct AI/ML related terms | 17.3.0 |
| 2023-03 | SA#99 | SP-230193 | 0018 | 1 | F | Correct formatting and spelling errors | 17.3.0 |
| 2023-03 | SA#99 | SP-230193 | 0019 | 1 | F | Correct attribute definitions | 17.3.0 |
| 2023-06 | SA#100 | SP-230655 | 0022 | 1 | F | Correcting the attribute properties | 17.4.0 |
| 2023-06 | SA#100 | SP-230649 | 0024 | 1 | F | Grammatical Corrections | 17.4.0 |
| 2023-06 | SA#100 | SP-230655 | 0030 | - | F | Removal of SW loading from training phase | 17.4.0 |
| 2023-06 | SA#100 | SP-230655 | 0031 | 1 | F | Correction of the figure for ML training function | 17.4.0 |
| 2023-06 | SA#100 | SP-230668 | 0023 | 1 | C | Not implemented due to violation of drafting rules. It will be modified and included in a future CR (MCC). | 18.0.0 |
| 2023-09 | SA#101 | SP-230948 | 0023 | 3 | C | Modelling ML Entity | 18.1.0 |
| 2023-09 | SA#101 | SP-230948 | 0035 |  | A | Clarify ML models as proprietary | 18.1.0 |
| 2023-09 | SA#101 | SP-230948 | 0039 | 1 | A | Restore the wrongly voided clause “5 Service and functional framework” | 18.1.0 |
| 2023-12 | SA#102 | SP-231459 | 0041 | 1 | F | Rel-18 CR TS 28.105 Adding the missing relation between ML entity and ML process – Partially implemented (1st change could not be implemented due to a clash with CR 066) | 18.2.0 |
| 2023-12 | SA#102 | SP-231467 | 0043 | 1 | A | Correction on ModelPerformance | 18.2.0 |
| 2023-12 | SA#102 | SP-231490 | 0045 | - | A | Rel-18 CR TS 28.105 Corrections of ML training related use cases description | 18.2.0 |
| 2023-12 | SA#102 | SP-231490 | 0047 | - | A | Rel 18 CR TS 28.105 Remove unused decision entity term | 18.2.0 |
| 2023-12 | SA#102 | SP-231490 | 0049 | - | A | Rel 18 CR TS 28.105 Clarify the description of confidenceIndication attribute | 18.2.0 |
| 2023-12 | SA#102 | SP-231467 | 0061 | - | A | Rel 18 CR TS 28.105 Remove unused attribute mLEntityList | 18.2.0 |
| 2023-12 | SA#102 | SP-231467 | 0063 | 1 | A | CR TS 28.105 Rel-18 Correction of IOC name | 18.2.0 |
| 2023-12 | SA#102 | SP-231467 | 0065 | 1 | A | TS 28.105 Rel-18 Correction of attribute properties | 18.2.0 |
| 2023-12 | SA#102 | SP-231459 | 0066 | 1 | F | TS 28.105 Rel-18 Correction of MLTrainingFunction constraints – Partially implemented (1st change could not be implemented due to a clash with CR 041) | 18.2.0 |
| 2023-12 | SA#102 | SP-231467 | 0068 | 1 | A | Rel 18 CR TS 28.105 Resolve issues related to the usage of confidenceIndication attribute | 18.2.0 |
| 2023-12 | SA#102 | SP-231490 | 0069 | - | A | Rel 18 CR TS 28.105 Fix incorrect figure label | 18.2.0 |
| 2023-12 |  |  |  |  |  | Alignment with the Forge | 18.2.0 |
| 2024-03 | SA#103 | SP-240186 | 0073 | - | F | TS28.105 Rel18 correction to Schema definition Issues for SubNetwork and ManagedElement of OpenAPI SS | 18.3.0 |
| 2024-03 | SA#103 | SP-240155 | 0075 | - | A | Rel-18 Correct trainingRequestSource attribute type | 18.3.0 |
| 2024-03 | SA#103 | SP-240155 | 0076 | 1 | B | Enhancements for AI-ML management | 18.3.0 |
| 2024-03 | SA#103 | SP-240162 | 0080 | 1 | A | Rel 18 CR TS 28.105 Add additional reference related to NWDAF | 18.3.0 |
| 2024-03 | SA#103 | SP-240162 | 0082 | 1 | A | Rel 18 CR TS 28.105 Add missing abbreviations | 18.3.0 |