

ETSI TS 128 105 V19.3.0 (2025-10)



**5G;
Management and orchestration;
Artificial Intelligence/ Machine Learning (AI/ML) management
(3GPP TS 28.105 version 19.3.0 Release 19)**



Reference

RTS/TSGS-0528105vj30

Keywords

5G

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from the
[ETSI Search & Browse Standards application](#).

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format on [ETSI deliver repository](#).

Users should be aware that the present document may be revised or have its status changed,
this information is available in the [Milestones listing](#).

If you find errors in the present document, please send your comments to
the relevant service listed under [Committee Support Staff](#).

If you find a security vulnerability in the present document, please report it through our
[Coordinated Vulnerability Disclosure \(CVD\)](#) program.

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2025.
All rights reserved.

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the [ETSI IPR online database](#).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

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

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at [3GPP to ETSI numbering cross-referencing](#).

Modal verbs terminology

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

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

Contents

Intellectual Property Rights	2
Legal Notice	2
Modal verbs terminology.....	2
Foreword.....	10
1 Scope	12
2 References	12
3 Definitions of terms, symbols and abbreviations	13
3.1 Terms.....	13
3.2 Symbols.....	14
3.3 Abbreviations	14
4 Concepts and overview	15
4.1 Overview	15
4.2 Management of AI/ML capabilities for RAN	16
4.3 Management of AI/ML capabilities for 5GC	16
4.4 Management of AI/ML capabilities for MDA.....	16
4a AI/ML management functionality and service framework	16
4a.0 ML model lifecycle	16
4a.1 Functionality and service framework for ML model training	17
4a.2 AI/ML functionalities management scenarios (relation with managed AI/ML features).....	18
5 Void.....	21
6 AI/ML management use cases and requirements	21
6.1 ML model lifecycle management capabilities.....	21
6.2 Void.....	22
6.2a Void.....	22
6.2b ML model training.....	22
6.2b.1 Description.....	22
6.2b.2 Use cases.....	22
6.2b.2.1 ML model training requested by consumer.....	22
6.2b.2.2 ML model training initiated by producer	23
6.2b.2.3 ML model selection.....	23
6.2b.2.4 Managing ML model training processes	23
6.2b.2.5 Handling errors in data and ML decisions	24
6.2b.2.6 ML model joint training	24
6.2b.2.7 ML model validation performance reporting	25
6.2b.2.8 Training data effectiveness reporting.....	25
6.2b.2.9 Performance management for ML model training	25
6.2b.2.9.1 Overview	25
6.2b.2.9.2 Performance indicator selection for MLmodel training.....	25
6.2b.2.9.3 ML model performance indicators query and selection for ML model training.....	26
6.2b.2.9.4 MnS consumer policy-based selection of ML model performance indicators for ML model training.....	26
6.2b.2.10 ML-Knowledge-based Transfer Learning.....	26
6.2b.2.10.1 Description	26
6.2b.2.10.2 Use cases	27
6.2b.2.11 ML model training for multiple contexts	28
6.2b.2.14 Management of distributed training	29
6.2b.2.15 Management of Federated learning.....	30
6.2b.2.15.1 Description	30
6.2b.2.15.2 Use cases	30
6.2b.2.16 Management of Reinforcement Learning.....	31
6.2b.2.16.1 Description	31
6.2b.2.16.2 Use cases	32

6.2b.2.17	Training data statistics.....	33
6.2b.3	Requirements for ML model training	34
6.2c	ML model testing	39
6.2c.1	Description.....	39
6.2c.2	Use cases.....	39
6.2c.2.1	Consumer-requested ML model testing	39
6.2c.2.2	Producer-initiated ML model testing	39
6.2c.2.3	Joint testing of multiple ML models	39
6.2c.2.4	Performance management for ML model testing.....	39
6.2c.2.4.1	Overview	39
6.2c.2.4.2	Performance indicator selection for ML model testing	39
6.2c.2.4.3	ML model performance indicators query and selection for ML model testing	40
6.2c.2.4.4	MnS consumer policy-based selection of ML model performance indicators for ML model testing	40
6.2c.3	Requirements for ML model testing	40
6.3	AI/ML inference emulation.....	41
6.3.1	Description.....	41
6.3.2	Use cases.....	41
6.3.2.1	AI/ML inference emulation.....	41
6.3.2.2	ML inference emulation environment selection.....	41
6.3.3	Requirements for Managing AI/ML inference emulation.....	42
6.4	ML model deployment	42
6.4.1	ML model loading	42
6.4.1.1	Description	42
6.4.1.2	Use cases	42
6.4.1.2.1	Consumer requested ML model loading.....	42
6.4.1.2.2	Control of producer-initiated ML model loading	42
6.4.1.2.3	ML model registration.....	43
6.4.1.3	Requirements for ML model loading	43
6.5	AI/ML inference.....	43
6.5.1	AI/ML inference performance management.....	43
6.5.1.1	Description	43
6.5.1.2	Use cases	43
6.5.1.2.1	AI/ML inference performance evaluation	43
6.5.1.2.2	AI/ML performance measurements selection based on MnS consumer policy.....	44
6.5.1.3	Requirements for AI/ML inference performance management.....	44
6.5.2	AI/ML update control	45
6.5.2.1	Description.....	45
6.5.2.2	Use cases	45
6.5.2.2.1	Availability of new capabilities or ML models	45
6.5.2.2.2	Triggering ML model update	45
6.5.2.3	Requirements for AIML update control.....	46
6.5.3	AI/ML inference capabilities management.....	46
6.5.3.1	Description	46
6.5.3.2	Use cases	47
6.5.3.2.1	Identifying capabilities of ML models	47
6.5.3.2.2	Mapping of the capabilities of ML models.....	47
6.5.3.3	Requirements for AI/ML inference capabilities management	47
6.5.4	AI/ML inference capability configuration management.....	48
6.5.4.1	Description	48
6.5.4.2	Use cases	48
6.5.4.2.1	Managing NG-RAN AI/ML-based distributed Network Energy Saving	48
6.5.4.2.2	Managing NG-RAN AI/ML-based distributed Mobility Optimization.....	48
6.5.4.2.3	Managing NG-RAN AI/ML-based distributed Load Balancing.....	48
6.5.4.3	Requirements for AI/ML inference management.....	49
6.5.5	AI/ML Inference History	49
6.5.5.1	Description	49
6.5.5.2	Use cases	49
6.5.5.2.1	AI/ML Inference History - tracking inferences and context.....	49
6.5.5.3	Requirements for AI/ML Inference History.....	50
6.5.6	Managing ML models in use in a live network.....	50
6.5.6.1	Description.....	50

6.5.6.2	Use cases	50
6.5.6.2.1	Handling of underperforming ML trained models in live networks	50
6.5.6.2.2	Performance monitoring of Network Functions with ML trained models in live networks	50
6.5.6.3	Requirements for Managing ML models in use in a live network	51
6.5.7.2.1	Management of explanation in AI/ML inference	51
6.5.7.3	Requirements for AI/ML inference explainability management	51
7	Information model definitions for AI/ML management	52
7.1	Imported and associated information entities	52
7.1.1	Imported information entities and local labels	52
7.1.2	Associated information entities and local labels	52
7.2	Void	52
7.2a	Common information model definitions for AI/ML management	52
7.2a.1	Class diagram	52
7.2a.1.1	Relationships	52
7.2a.1.2	Inheritance	53
7.2a.2	Class definitions	53
7.2a.2.1	MLModel	53
7.2a.2.1.1	Definition	53
7.2a.2.1.2	Attributes	54
7.2a.2.1.3	Attribute constraints	54
7.2a.2.1.4	Notifications	54
7.2a.2.2	MLModelRepository	54
7.2a.2.2.1	Definition	54
7.2a.2.2.2	Attributes	54
7.2a.2.2.3	Attribute constraints	54
7.2a.2.2.4	Notifications	54
7.2a.2.3	MLModelCoordinationGroup	55
7.2a.2.3.1	Definition	55
7.2a.2.3.2	Attributes	55
7.2a.2.3.3	Attribute constraints	55
7.2a.2.3.4	Notifications	55
7.3	Void	55
7.3a	Information model definitions for AI/ML operational phases	55
7.3a.1	Information model definitions for ML model training	55
7.3a.1.1	Class diagram	55
7.3a.1.1.1	Relationships	55
7.3a.1.1.2	Inheritance	56
7.3a.1.2	Class definitions	56
7.3a.1.2.1	MLTrainingFunction	56
7.3a.1.2.1.1	Definition	56
7.3a.1.2.1.2	Attributes	57
7.3a.1.2.1.3	Attribute constraints	57
7.3a.1.2.1.4	Notifications	57
7.3a.1.2.2	MLTrainingRequest	57
7.3a.1.2.2.1	Definition	57
7.3a.1.2.2.2	Attributes	58
7.3a.1.2.2.3	Attribute constraints	59
7.3a.1.2.2.4	Notifications	59
7.3a.1.2.3	MLTrainingReport	59
7.3a.1.2.3.1	Definition	59
7.3a.1.2.3.2	Attributes	59
7.3a.1.2.3.3	Attribute constraints	60
7.3a.1.2.3.4	Notifications	60
7.3a.1.2.4	MLTrainingProcess	60
7.3a.1.2.4.1	Definition	60
7.3a.1.2.4.2	Attributes	61
7.3a.1.2.4.3	Attribute constraints	62
7.3a.1.2.4.4	Notifications	62
7.3a.2	Information model definitions for AI/ML inference emulation	65
7.3a.2.1	Class diagram	65

7.3a.2.1.1	Relationships	65
7.3a.2.1.2	Inheritance	66
7.3a.2.2	Class definitions	66
7.3a.2.2.1	AIMLInferenceEmulationFunction	66
7.3a.2.2.1.1	Definition	66
7.3a.2.2.1.2	Attributes	66
7.3a.2.2.1.3	Attribute constraints	66
7.3a.2.2.1.4	Notifications	66
7.3a.3	Information model definitions for ML model deployment	66
7.3a.3.1	Class diagram	66
7.3a.3.1.1	Relationships	66
7.3a.3.1.2	Inheritance	67
7.3a.3.2	Class definitions	67
7.3a.3.2.1	MLModelLoadingRequest	67
7.3a.3.2.1.1	Definition	67
7.3a.3.2.1.2	Attributes	68
7.3a.3.2.1.3	Attribute constraints	68
7.3a.3.2.1.4	Notifications	68
7.3a.3.2.2	MLModelLoadingPolicy	68
7.3a.3.2.2.1	Definition	68
7.3a.3.2.2.2	Attributes	68
7.3a.3.2.2.3	Attribute constraints	68
7.3a.3.2.2.4	Notifications	69
7.3a.3.2.3	MLModelLoadingProcess	69
7.3a.3.2.3.1	Definition	69
7.3a.3.2.3.2	Attributes	69
7.3a.3.2.3.3	Attribute constraints	70
7.3a.3.2.3.4	Notifications	70
7.3a.4	Information model definitions for AI/ML inference	70
7.3a.4.1	Class diagram	70
7.3a.4.1.1	Relationships	70
7.3a.4.1.2	Inheritance	71
7.3a.4.2	Class definitions	71
7.3a.4.2.1	MLUpdateFunction	71
7.3a.4.2.1.1	Definition	71
7.3a.4.2.1.2	Attributes	72
7.3a.4.2.1.3	Attribute constraints	72
7.3a.4.2.1.4	Notifications	72
7.3a.4.2.2	MLUpdateRequest	72
7.3a.4.2.2.1	Definition	72
7.3a.4.2.2.2	Attributes	73
7.3a.4.2.2.3	Attribute constraints	73
7.3a.4.2.2.4	Notifications	73
7.3a.4.2.3	MLUpdateProcess	73
7.3a.4.2.3.1	Definition	73
7.3a.4.2.3.2	Attributes	74
7.3a.4.2.3.3	Attribute constraints	74
7.3a.4.2.3.4	Notifications	74
7.3a.4.2.4	MLUpdateReport	74
7.3a.4.2.4.1	Definition	74
7.3a.4.2.4.2	Attributes	75
7.3a.4.2.4.3	Attribute constraints	75
7.3a.4.2.4.4	Notifications	75
7.3a.4.2.5	AIMLInferenceFunction	75
7.3a.4.2.5.1	Definition	75
7.3a.4.2.5.2	Attributes	75
7.3a.4.2.5.3	Attribute constraints	76
7.3a.4.2.5.4	Notifications	76
7.3a.4.2.6	AIMLInferenceReport	76
7.3a.4.2.6.1	Definition	76
7.3a.4.2.6.2	Attributes	76

7.3a.4.2.6.3	Attribute constraints	76
7.3a.4.2.6.4	Notifications	76
7.4	Data type definitions	77
7.4.1	ModelPerformance <<dataType>>	77
7.4.1.1	Definition	77
7.4.1.2	Attributes	77
7.4.1.3	Attribute constraints	77
7.4.1.4	Notifications	77
7.4.2	Void	77
7.4.3	MLContext <<dataType>>	77
7.4.3.1	Definition	77
7.4.3.2	Attributes	77
7.4.3.3	Attribute constraints	77
7.4.3.4	Notifications	78
7.4.4	SupportedPerfIndicator <<dataType>>	78
7.4.4.1	Definition	78
7.4.4.2	Attributes	78
7.4.4.3	Attribute constraints	78
7.4.4.4	Notifications	78
7.4.5	AvailMLCapabilityReport <<dataType>>	78
7.4.5.1	Definition	78
7.4.5.2	Attributes	79
7.4.5.3	Attribute constraints	79
7.4.5.4	Notifications	79
7.4.6	AIMLManagementPolicy <<dataType>>	79
7.4.6.1	Definition	79
7.4.6.2	Attributes	79
7.4.6.3	Attribute constraints	79
7.4.6.4	Notifications	79
7.4.7	ManagedActivationScope <<choice>>	79
7.4.7.1	Definition	79
7.4.7.2	Attributes	80
7.4.7.3	Attribute constraints	80
7.4.7.4	Notifications	80
7.4.8	MLCapabilityInfo <<dataType>>	80
7.4.8.1	Definition	80
7.4.8.2	Attributes	80
7.4.8.3	Attribute constraints	80
7.4.8.4	Notifications	80
7.4.9	InferenceOutput <<dataType>>	80
7.4.9.1	Definition	80
7.4.9.2	Attributes	81
7.4.9.3	Attribute constraints	81
7.4.9.4	Notifications	81
7.4.10	AIMLINferenceName <<choice>>	81
7.4.10.1	Definition	81
7.4.10.2	Attributes	81
7.4.10.3	Attribute constraints	81
7.4.10.4	Notifications	81
7.4.11	DataStatisticalProperties <<dataType>>	82
7.4.11.1	Definition	82
7.4.11.2	Attributes	82
7.4.11.3	Attribute constraints	82
7.4.11.4	Notifications	82
7.4.12	DistributedTrainingExpectation <<dataType>>	82
7.4.12.1	Definition	82
7.4.12.2	Attributes	82
7.4.12.3	Attribute constraints	82
7.4.12.4	Notifications	82
7.4.13	PotentialImpactInfo <<dataType>>	82
7.4.13.1	Definition	82

7.4.13.2	Attributes.....	83
7.4.13.3	Attribute constraints	83
7.4.13.4	Notifications.....	83
7.4.14	ImpactedPM <<dataType>>	83
7.4.14.1	Definition	83
7.4.14.2	Attributes.....	83
7.4.14.3	Attribute constraints	83
7.4.14.4	Notifications.....	83
7.4.15	MLKnowledge <<dataType>>	83
7.4.15.1	Definition	83
7.4.15.2	Attributes.....	84
7.4.15.3	Attribute constraints	84
7.4.15.4	Notifications.....	84
7.4.16	EnvironmentScope <<choice>>	84
7.4.16.1	Definition	84
7.4.16.2	Attributes.....	84
7.4.16.3	Attribute constraints	84
7.4.16.4	Notifications.....	84
7.4.17	SupportedLearningTechnology <<dataType>>	85
7.4.17.1	Definition	85
7.4.17.2	Attributes.....	85
7.4.17.3	Attribute constraints	85
7.4.17.4	Notifications.....	85
7.4.18	RLRequirement <<dataType>>	85
7.4.18.2	Attributes.....	86
7.4.18.3	Attribute constraints	86
7.4.18.4	Notifications.....	86
7.4.19	ClusteringCriteria <<dataType>>	86
7.4.19.1	Definition	86
7.4.19.2	Attributes.....	86
7.4.19.3	Attribute constraints	87
7.4.19.4	Notifications.....	87
7.4.20	FLParticipationInfo <<dataType>>	87
7.4.20.1	Definition	87
7.4.20.2	Attributes.....	87
7.4.20.3	Attribute constraints	87
7.4.20.4	Notifications.....	87
7.4.21	FLRequirement <<dataType>>	87
7.4.21.1	Definition	87
7.4.21.2	Attributes.....	88
7.4.21.3	Attribute constraints	88
7.4.21.4	Notifications.....	88
7.4.22	FLClientSelectionCriteria <<dataType>>	88
7.4.22.1	Definition	88
7.4.22.2	Attributes.....	88
7.4.22.3	Attribute constraints	88
7.4.22.4	Notifications.....	88
7.4.23	FLReportPerClient <<dataType>>	88
7.4.23.1	Definition	88
7.4.23.2	Attributes.....	89
7.4.23.3	Attribute constraints	89
7.4.23.4	Notifications.....	89
7.4a	Enumerations.....	89
7.4a.1	NgRanInferenceType <<enumeration>>	89
7.5	Attribute definitions	90
7.5.1	Attribute properties	90
7.5.2	Constraints	109
7.6	Common notifications	109
7.6.1	Configuration notifications	109
8	Service components.....	109
8.0	General	109

8.1	Lifecycle management operations for AI/ML management MnS	109
9	Solution Set (SS)	111
9.1	OpenAPI document for provisioning MnS	111
9.2	OpenAPI document for AI/ML management	111
10	Stage 3 definition for AI/ML Management	112
10.1	RESTful HTTP-based solution set	112
10.1.1	ML model training	112
10.1.2	ML model testing	113
10.1.3	AI/ML inference emulation	113
10.1.4	ML model deployment	114
10.1.5	AI/ML inference	115
Annex A (informative): PlantUML source code for NRM class diagrams.....		117
A.1	General	117
A.2	PlantUML code for Figure 7.3a.1.1.1-1: NRM fragment for ML model training	117
A.3	PlantUML code for Figure 7.3a.1.1.2-1: Inheritance Hierarchy for ML model training related NRMs	118
A.4	PlantUML code for Figure 7.2a.1.2-1: Inheritance Hierarchy for common information models for AI/ML management	119
A.5	PlantUML code for Figure 7.2a.1.1-1: Relationships for common information models for AI/ML management	119
A.6	PlantUML code for Figure 7.3a.1.1.1-2: NRM fragment for ML model testing	119
A.7	PlantUML code for Figure 7.3a.1.1.2-2: Inheritance Hierarchy for ML model testing related NRMs	120
A.8	PlantUML code for Figure 7.3a.4.1.1-1: NRM fragment for ML update	120
A.9	PlantUML code for Figure 7.3a.4.1.2-1: Inheritance Hierarchy for ML update related NRMs.....	121
A.10	PlantUML code for Figure 7.3a.3.1.1-1: NRM fragment for ML model loading	121
A.11	PlantUML code for Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML model loading related NRMs	122
A.12	PlantUML code for Figure 7.3a.4.1.1-2: NRM fragment for AI/ML inference function.....	122
A.13	PlantUML code for Figure 7.3a.4.1.2-2: Inheritance Hierarchy for AI/ML inference function	123
A.14	PlantUML code for Figure 7.3a.2.1.1-1: NRM fragment for AI/ML inference emulation Control.....	123
A.15	PlantUML code for Figure 7.3a.2.1.2-1: AI/ML inference emulation Inheritance Relations	124
Annex B (normative): OpenAPI definition of the AI/ML NRM		125
B.1	General	125
B.2	Solution Set (SS) definitions	125
B.2.1	OpenAPI document "TS28105_AiMLNrm.yaml"	125
Annex C (informative): Change history		126
History		130

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".

- [19] 3GPP TS 28.623: "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Solution Set (SS) definitions".
- [20] 3GPP TS 29.520: "5G System; Network Data Analytics Services; Stage 3".
- [21] 3GPP TS 28.319: "Management and orchestration; Access Control for Management services".
- [22] 3GPP TS 28.111: "Management and orchestration; Fault Management (FM)".

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 model: a manageable representation of an ML model algorithm.

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

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

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

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

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

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

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

ML model pre-specialized training: the process of training an ML model on a dataset not specific to any type of inference.

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

NOTE 5: The pre-specialised trained model supports an inference scope that may be potentially adapted to support a list of inference types, such as MDA types in MDA, analytics types in NWDAF, type of AI/ML supported functions in NG-RAN, or vendor-specific extensions.

NOTE 6: The inference scope refers to a list of inference types that the ML model may be potentially adapted to support.

NOTE 7: The type of inference represents the specific type of ML inference supported by the model, such as MDA types in MDA, Analytics types in NWDAF, type of AI/ML supported functions in NG-RAN, or vendor-specific extensions.

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

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

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

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

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

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

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

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

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

NOTE 8: Examples of experience include statistics (e.g. a distribution) or summaries (e.g. tables) indicating the ML model's recommended output for a given set of input data.

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

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

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

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

NOTE 9: The inference represents the process to realize the AI capabilities by utilizing a trained ML model and other AI enablers if needed, hence the AI/ML prefix is used when referring to inference as compared to training and testing.

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

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

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

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

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

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

3.2 Symbols

Void.

3.3 Abbreviations

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

AI	Artificial Intelligence
ML	Machine Learning
FL	Federated Learning
HFL	Horizontal Federated Learning
VFL	Vertical Federated Learning

4 Concepts and overview

4.1 Overview

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

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

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

- **Learning methods**

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

Table 4.1-1: Comparison of Learning methods

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

- **Learning complexity:**

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

- **Learning architecture**

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

- **Learning continuity**

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

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

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

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

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

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

- ML model training.
- ML model testing.

- AI/ML inference emulation.
- ML model deployment.
- AI/ML inference.

4.2 Management of AI/ML capabilities for RAN

The management of AI/ML capabilities for the RAN covers scenarios where both the ML model training and AI/ML inference are located in the NG-RAN node, as well as scenarios where the ML model training is located in the management system and the AI/ML inference is located in the NG-RAN node. In either case, the NG-RAN AI/ML-based feature defined in clause 16.20 of TS 38.300 [16] can be supported.

4.3 Management of AI/ML capabilities for 5GC

The management of AI/ML capabilities for the 5GC covers scenarios where both the ML model training and AI/ML inference functions are located in the 5GC. In this case, the NWDAF feature defined in clause 6 of TS 23.288 [3] can be supported.

4.4 Management of AI/ML capabilities for MDA

For MDA, the ML training function can be located either inside or outside the MDAF while the AI/ML inference function is located in the MDAF. In this case, the MDA capabilities defined in clause 7.2 of TS 28.104 [2] can be supported.

4a AI/ML management functionality and service framework

4a.0 ML model lifecycle

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

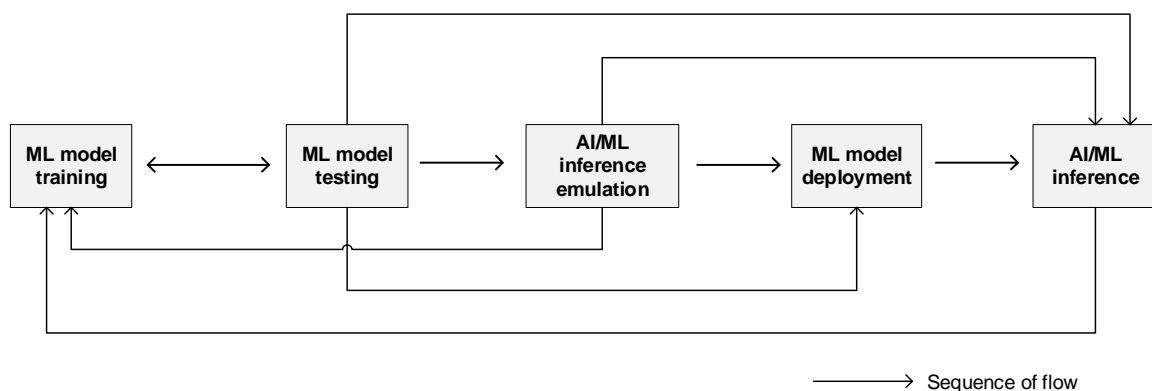


Figure 4a.0-1: ML model lifecycle

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

- **ML model training:** training, including initial training, re-training, pre-specialised training and fine-tuning. Training could be for a single ML model or a group of ML models. It may also include validation of the ML model(s) to evaluate the performance when the ML model(s) performs on the validation data. If the validation result does not meet the expectation (e.g., the variance is not acceptable), the MnS Producer may decide to re-train the ML model(s).
- **ML model testing:** testing of a validated ML model to evaluate the performance of the trained ML model when it performs on testing data. If the testing result meets the expectations, the ML model may proceed to the next step. If the testing result does not meet the expectations, the ML model needs to be re-trained.
- **AI/ML inference emulation:** running an ML model for inference in an emulation environment. The purpose is to evaluate the inference performance of the ML model in the emulation environment prior to applying it to the target network or system. If the emulation result does not meet the expectation (e.g., inference performance does not meet the target, or the ML model negatively impacts the performance of other existing functionalities), the ML model needs to be re-trained.

NOTE 1: The AI/ML inference emulation is considered optional and can be skipped in the ML model lifecycle.

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

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

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

NOTE 2: Depending on system implementation and AI/ML functionality arrangements, both AI/ML inference emulation and ML deployment steps may be skipped.

NOTE 3: The sequence of ML model lifecycle in Figure 4a.0-1 is the typical process in an E2E perspective. The actual sequence will depend on certain training techniques as described in their use cases, e.g., ML reinforcement learning, Distributed training or Federated learning, etc.

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

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

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

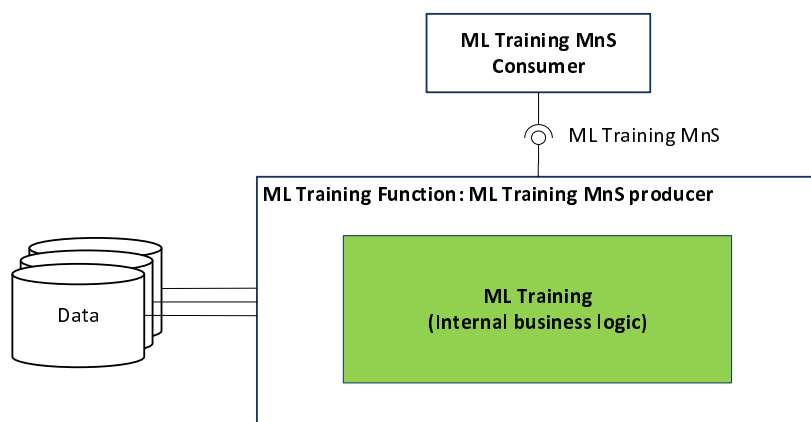


Figure 4a.1-1: Functional overview and service framework for ML model training

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

- Performance Measurements (PM) as per 3GPP TS 28.552 [4], 3GPP TS 32.425 [5] and Key Performance Indicators (KPIs) as per 3GPP TS 28.554 [6].
- Trace/MDT/RLF/RCEF/RRC data, as per 3GPP TS 32.422 [7].
- QoE and service experience data as per 3GPP TS 28.405 [9].
- Analytics data offered by NWDAF as per 3GPP TS 23.288 [3].
- Alarm information and notifications as per 3GPP TS 28.111 [22].
- 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 (relation with managed AI/ML features)

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

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

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

For NG-RAN, the ML training function and AI/ML inference function can both be located in the NG-RAN node, or the ML training function can be located in the management system and AI/ML inference function is located in the NG-RAN node. Where the ML training function corresponds to ML model training that stated in clause 16.20.2 in TS 38.300 [16] and AI/ML inference function can correspond to AI/ML inference stated in clause 16.20.2 in TS 38.300 [16].

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

Therefore, multiple location scenarios for ML training function and AI/ML inference functions are possible.

Scenario 1:

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

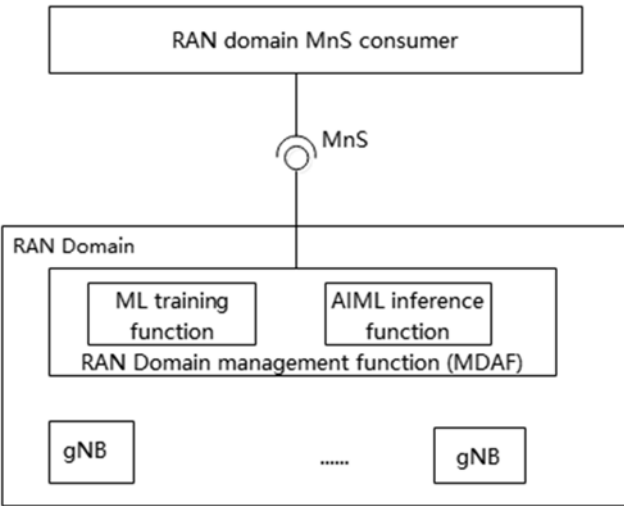


Figure 4a.2-1: Management for RAN domain specific MDAF

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

Scenario 2:

For AI/ML for NG-RAN, the ML model training is located in the 3GPP RAN domain-specific management function while the AI/ML inference is located in NG-RAN node. For AI/ML inference use case, refer to Network Energy Saving, Load Balancing, Mobility Optimization as defined in clause 16.20 in TS 38.300 [16]. See Figure 4a.2-2.

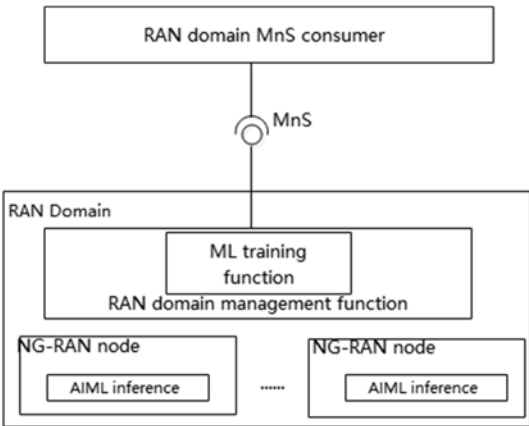


Figure 4a.2-2: Management where the ML model training is located in RAN domain management function and AI/ML inference is located in NG-RAN node

Scenario 3:

For AI/ML in NG-RAN, the ML model training and AI/ML inference are both located in the NG-RAN node. For ML model training and AI/ML inference use case, refer to Network Energy Saving, Load Balancing, Mobility Optimization as defined in clause 16.20 in TS 38.300 [16]. See Figure 4a.2-3.

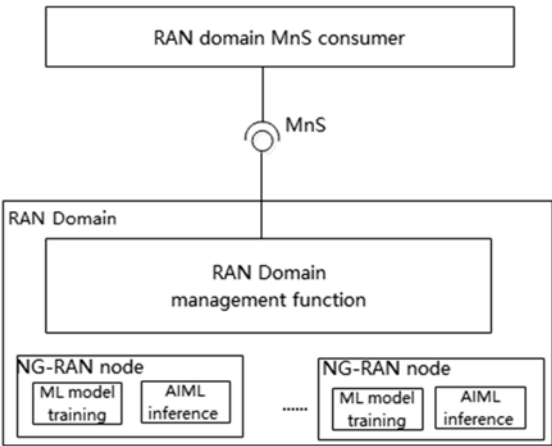


Figure 4a.2-3: Management where the ML model training and AI/ML inference are both located in NG-RAN node.

Scenario 4:

For NWDAF, both the MTLF and AnLF are located in the NWDAF. See Figure 4a.2-4.

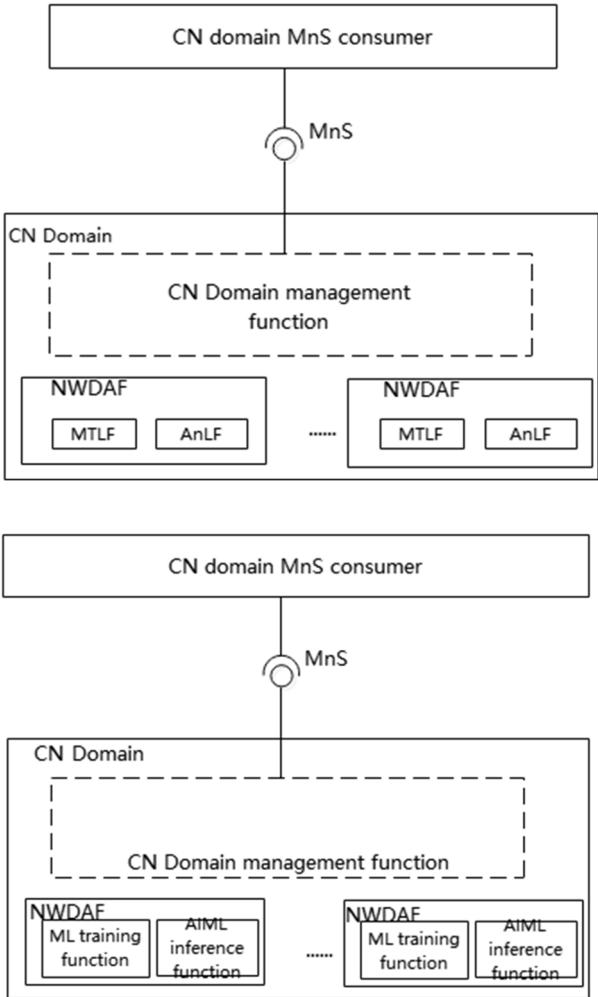


Figure 4a.2-4: Management where the MTLF and AnLF are both located in CN

5 Void

6 AI/ML management use cases and requirements

6.1 ML model lifecycle management capabilities

Each operational step in the ML model lifecycle (see clause 4a.0.1) is supported by one or more AI/ML management capabilities, which enable the MnS consumer (e.g. operator) to manage and control the ML model lifecycle as listed below.

Management capabilities for ML model training

- **ML model training management:** allowing the MnS consumer to request the ML model training, consume and control the producer-initiated training, and manage the ML model training/re-training process. The training management capability may include training performance management and setting a policy for the producer-initiated ML model training.
- ML model training capability also includes validation to evaluate the performance of the ML model 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 model would need to be re-trained before being made available for the next step in the ML model lifecycle (e.g., ML model testing).

Management capabilities for ML testing

- **ML model testing management:** allowing the MnS consumer to request the ML model testing, and to receive the testing results for a trained ML model. 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 model re-training based on the ML model testing performance results.

Management capabilities for AI/ML inference emulation:

- **AI/ML inference emulation:** a capability allowing an MnS consumer to request an ML inference emulation for a specific ML model or models (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 model deployment:

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

Management capabilities for AI/ML inference:

- **AI/ML inference management:** allowing an MnS consumer to control the inference, i.e., activate/deactivate the inference function and/or ML model/models, 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 model or an AI/ML inference function.

The use cases and corresponding requirements for AI/ML management capabilities are specified in the following clauses.

6.2 Void

6.2a Void

6.2b ML model training

6.2b.1 Description

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

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

The procedures in [21] for authentication and authorization are applicable for all ML model training use cases.

6.2b.2 Use cases

6.2b.2.1 ML model training requested by consumer

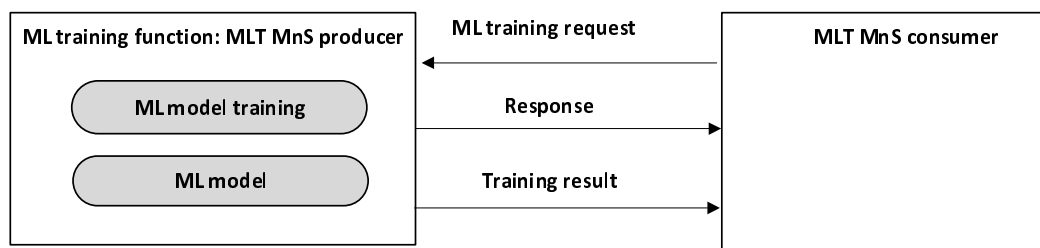


Figure 6.2b.2.1-1: ML model training requested by ML training MnS consumer

The ML model 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.

To trigger an initial ML model training, the MnS consumer needs to specify in the ML training request the inference type which indicates the function or purpose of the ML model, 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 model re-training, the MnS consumer needs to specify in the ML training request the identifier of the ML model 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 model 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 model degrades over time. The ML training MnS producer may re-train the ML model if the inference performance of the ML model falls below a certain threshold, which needs to be configurable by the MnS consumer.

Following the ML training request by the ML 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 model 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 model, 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;
- validate the trained model using validation set of the training data;
- provides the training results (including the identifier of the ML model generated from the initially trained ML model or the version number of the ML model associated with the re-trained model, training performance results, etc.) to the ML training MnS consumer(s).

6.2b.2.2 ML model training initiated by producer

The ML model training may be initiated by the ML training MnS producer, for instance as a result of performance evaluation of the ML model 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 model 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 model 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 model generated from the initially trained ML model or the version number of the ML model associated with the re-trained model, training performance, etc.) to the ML training MnS consumer(s) who have subscribed to receive the ML model training results.

6.2b.2.3 ML model 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 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 model, with the variants of trained ML models and to select the appropriate one for their respective conditions.

Besides, there is no guarantee that the available ML models 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 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.

6.2b.2.4 Managing ML model training processes

This relates to the management and controlling of the ML model 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.

6.2b.2.5 Handling errors in data and ML decisions

Ideally, the ML models 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 model 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 model 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 model is very likely to come across these inconsistencies. When this happens, the ML model shows high error in the inference outputs, even if redundant and uncorrupted data are available from other sources.

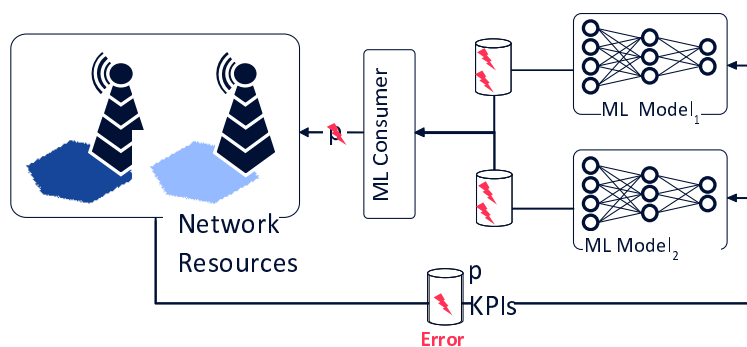


Figure 6.2b.2.5-1: The propagation of erroneous information

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

6.2b.2.6 ML model joint training

Each ML model supports a specific type of inference. An AI/ML inference function may use one or more ML models to perform the inference(s). When multiple ML models are employed, these ML models 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 model 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 models may coordinate. An example is the case where the output of one ML model can be used as input to another ML model forming a sequence of interlinked ML models. Another example is the case where multiple ML models 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 model. The group of ML models needs to be employed in a coordinated way to support an AI/ML inference function.

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

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

6.2b.2.7 ML model validation performance reporting

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

6.2b.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 model training based on insight of how the different portion of data contribute differently to the trained model accuracy.

6.2b.2.9 Performance management for ML model training

6.2b.2.9.1 Overview

In the ML model training, the performance of ML model needs to be evaluated on training data. The performance is the degree to which the ML models fulfil the objectives for which they were trained. The related performance indicators need to be collected and analyzed.

6.2b.2.9.2 Performance indicator selection for MLmodel training

The ML model training function may support training for a single or several ML model algorithm and may support the capability to evaluate each ML model 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 model. The performance indicators for training mainly include the following aspects: ML model training process monitors performance indicators: the performance indicators of the system that trains the ML model, including training duration indicator.

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

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

6.2b.2.9.3 ML model performance indicators query and selection for ML model training

ML model performance evaluation and management are needed during training. The related performance indicators need to be collected and analyzed. The MnS producer of ML model training 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 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 model training can be queried to provide the information on supported performance indicators referring to ML model training. Such performance indicators for training may be for example accuracy/precision/recall/F1-score/MSE/MAE. Based on supported performance indicators as well as based on consumer's requirements, the MnS consumer for ML model training 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 model training or to query the supported performance indicators and select a sub-set of performance indicators to be used for performance evaluation.

6.2b.2.9.4 MnS consumer policy-based selection of ML model performance indicators for ML model training

ML model performance evaluation and management is needed during ML model training. The related performance indicators need to be collected and analysed. The MnS producer for ML model 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 model training 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 by the MnS producer. These performance requirements do not need to indicate the technical performance indicators used for ML model 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 model training may not be capable enough to indicate the performance metrics to be used for training.

6.2b.2.10 ML-Knowledge-based Transfer Learning

6.2b.2.10.1 Description

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

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

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

For example, knowledge contained in an ML model deployed to perform mobility optimization during the day can be leveraged to produce a new ML model to perform mobility optimization at night. As illustrated in Figure 6.2b.2.10.1-1, the network or its management system needs to have the required management services to support ML Knowledge-based Transfer Learning (MLKTL). ML Knowledge-based Transfer Learning refers to the capability of enabling and managing the transfer learning between any two ML models or training functions.

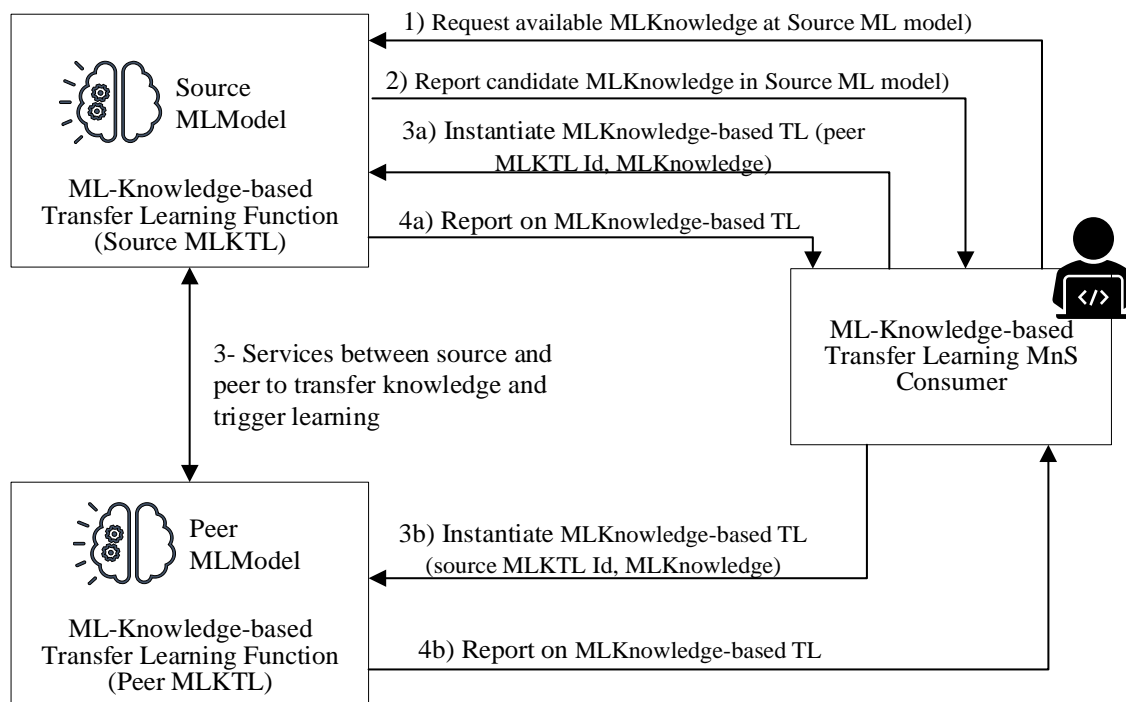


Figure 6.2b.2.10.1-1: ML Knowledge-based Transfer Learning (MLKTL) flow between the source MLKTL (which is the ML training function with the pre-trained ML model), the peer MLKTL (which is the ML training function that shall train a new ML model) and the MLKTL MnS consumer (which may be the operator or another management function that triggers or controls MLKTL)

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

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

6.2b.2.10.2 Use cases

6.2b.2.10.2.1 Discovering sharable Knowledge

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

6.2b.2.10.2.2 Knowledge sharing and transfer learning

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

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

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

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

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

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

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

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

6.2b.2.11 ML model training for multiple contexts

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

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

6.2b.2.12 ML Pre-specialised training

ML model pre-specialised training refers to the process of training an ML model using a dataset that is not specific to one single type of inference. This means that this type of ML model training is not intended to support only one type of inference but rather leverages commonalities among multiple use cases. ML model pre-specialised training can be applied to AI/ML-based use cases specified in [2], and [3]. For example, an ML model could be pre-specialised trained using dataset from SLS analysis capability group covering type of inference including ServiceExperienceAnalysis, NetworkSliceThroughputAnalysis, NetworkSliceTrafficAnalysis, NetworkSliceLoadAnalysis and E2ELatencyAnalysis (see TS 28.104 [4]).

A pre-specialised trained ML model supports more than one inference type (i.e., is not designed to conduct inference for a specific inference type), but this does not preclude the possibility for a pre-specialised model to conduct inference once it achieved performance requirement for a specific inference type.

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

6.2b.2.13 ML Fine-tuning of pre-specialised trained model

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

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

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

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

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

6.2b.2.14 Management of distributed training

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

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

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

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

When receiving an ML training request, the MLT MnS producer may evaluate whether distributed training is needed according to the training requirements provided by the ML training consumer, and it is up to the MLT MnS producer to determine, based on some information (e.g target inference location) provided by the consumer, appropriate training function(s) which may need to participate in the ML model training. The training requirement may further include (not

limited to) expected model performance. Collaboration, mutual agreement and authentication procedures are needed to be established between distributed ML training functions before sharing any information between these functions.

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

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

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

6.2b.2.15 Management of Federated learning

6.2b.2.15.1 Description

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

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

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

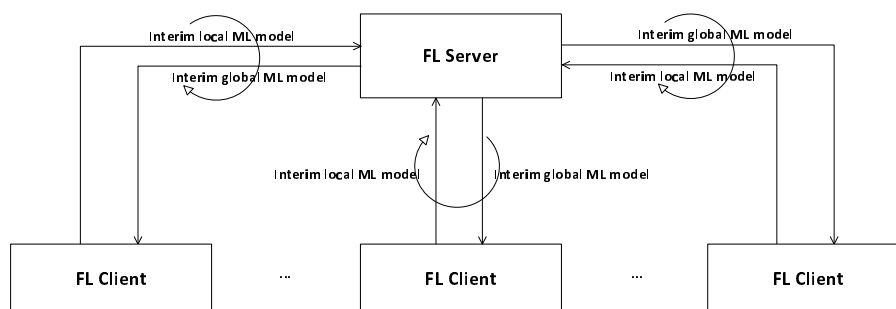


Figure 6.2b.2.15.1-1: ML model distribution and aggregation for HFL

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

6.2b.2.15.2 Use cases

6.2b.2.15.2.1 Management of different roles in Federated learning

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

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

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

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

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

6.2b.2.16 Management of Reinforcement Learning

6.2b.2.16.1 Description

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

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

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

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

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

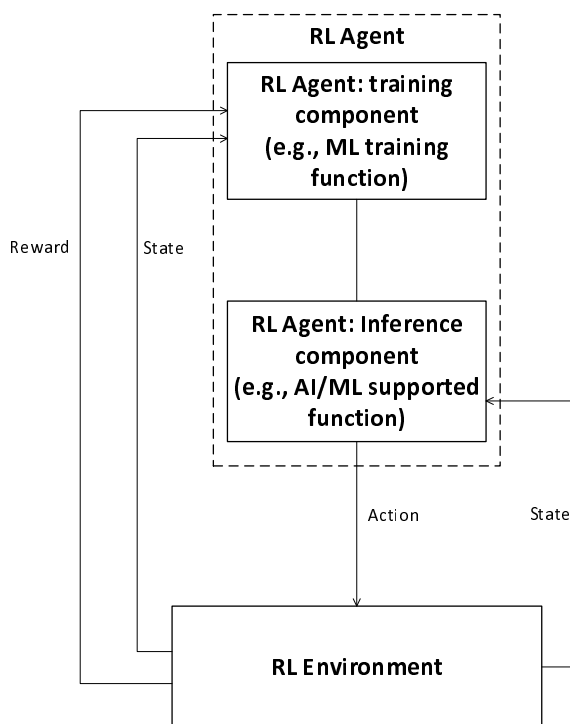


Figure 6.2b.2.16.1-1: RL in online mode

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

NOTE 1: RL in online mode is not supported in current release specifications.

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

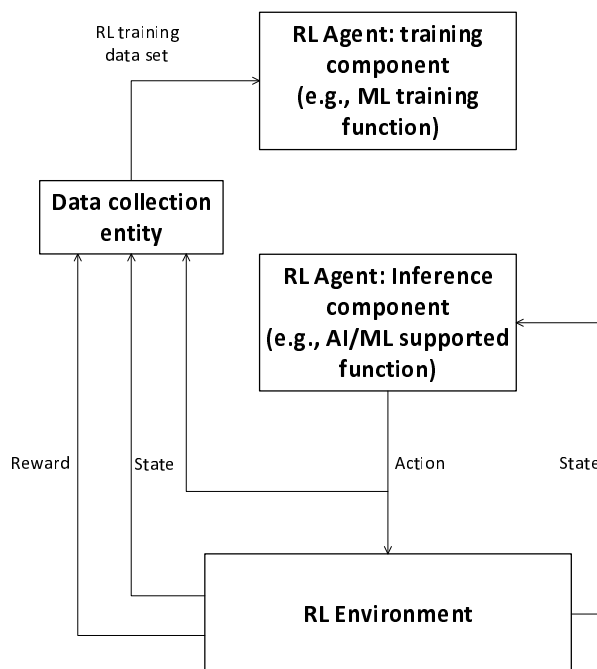


Figure 6.2b.2.16.1-2: RL in offline mode

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

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

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

NOTE 2: Rewards, states, and actions are not subject to standardization. While measurements and KPIs may be defined, the mapping of such data to states and rewards is implementation-specific.

NOTE 3: Figures 6.2b.2.16.1-1 and 6.2b.2.16.1-2 conceptually and logically illustrate how the RL process works in both online and offline modes, without restricting the implementation of RL agent training and inference components.

6.2b.2.16.2 Use cases

6.2b.2.16.2.1 Enabling Reinforcement Learning

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

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

- **RL agent training component:** collects the data for training, trains the RL model and reports the training result of the training component. The data for training includes state transitions, rewards from the RL environment for online RL, or the pre-processed data set for a period of time from the data collection entity for offline RL, as

well as the actions taken by the RL agent inference component. The trained RL model would be used by the inference component to determine actions for observed state conditions. Additionally, the RL agent training component needs to indicate the RL environment(s) (e.g., a live subnetwork or a simulation subnetwork) for which an RL model has been trained.

- **RL agent inference component:** activates the trained RL model (RL policy), receives state transitions from the RL environment, and determine actions based on the RL model. The RL agent inference component needs to be configured with the information about the RL environment(s) where a trained RL model can be used.
- **RL environment:** executes the actions determined by the RL agent inference component and reports rewards to the RL agent training component for online RL or to the data collection entity for offline RL.

6.2b.2.16.2.2 Exploration in Reinforcement Learning

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

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

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

6.2b.2.17 Training data statistics

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

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

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

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

6.2b.3 Requirements for ML model training

Table 6.2b.3-1

Requirement label	Description	Related use case(s)
REQ-ML_TRAIN-FUN-01	The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to request ML model training.	ML model training requested by consumer (clause 6.2b.2.1)
REQ- ML_TRAIN-FUN-02	The ML training MnS producer shall have a capability allowing the authorized ML training MnS consumer to specify the data sources containing the candidate training data for ML model training.	ML model training requested by consumer (clause 6.2b.2.1)
REQ- ML_TRAIN-FUN-03	The ML training MnS producer shall have a capability allowing the authorized ML training MnS consumer to specify the AI/ML inference name of the ML model to be trained.	ML model training requested by consumer (clause 6.2b.2.1)
REQ- ML_TRAIN-FUN-04	The ML training MnS producer shall have a capability to provide the training result to the ML training MnS consumer.	ML model training requested by consumer (clause 6.2b.2.1), ML model training initiated by MnS producer (clause 6.2b.2.2)
REQ- ML_TRAIN-FUN-05	The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to configure the thresholds of the performance measurements and/or KPIs to trigger the re-training of an ML model. (See Note)	ML model training initiated by MnS producer (clause 6.2b.2.2)
REQ- ML_TRAIN-FUN-06	The ML training MnS producer shall have a capability to provide the version number of the ML model when it is generated by ML model re-training to the authorized ML training MnS consumer.	ML model training requested by consumer (clause 6.2b.2.1), ML model training initiated by MnS producer (clause 6.2b.2.2)
REQ- ML_TRAIN-FUN-07	The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to manage the training process, including starting, suspending, or resuming the training process, and configuring the ML context for ML model training.	ML model training requested by consumer (clause 6.2b.2.1), ML model training initiated by MnS producer (clause 6.2b.2.2), ML model joint training (clause 6.2b.2.6)
REQ- ML_TRAIN-FUN-08	The ML training MnS producer should have a capability to provide the grouping of ML models to an authorized ML training MnS consumer to enable coordinated inference.	ML model joint training (clause 6.2b.1.2.6)
REQ- ML_TRAIN-FUN-09	The ML training MnS producer should have a capability to allow an authorized ML training MnS consumer to request joint training of a group of ML models.	ML model joint training (clause 6.2b.2.6)
REQ- ML_TRAIN-FUN-10	The ML training MnS producer should have a capability to jointly train a group of ML models and provide the training results to an authorized consumer.	ML model joint training (clause 6.2b.2.6)
REQ-ML_SELECT-01	3GPP management system shall have a capability to enable an authorized ML training MnS consumer to discover the properties of available ML models including the contexts under which each of the models were trained.	ML model and ML model selection (clause 6.2b.2.3)
REQ-ML_SELECT-02	3GPP management system shall have a capability to enable an authorized ML training MnS consumer to select an ML model to be used for inference.	ML models and ML model selection (clause 6.2b.2.3)
REQ-ML_SELECT-03	3GPP management system shall have a capability to enable an authorized ML training MnS consumer to request for information and be informed about the available alternative ML models of differing complexity and performance.	ML model and ML model selection (clause 6.2b.2.3)
REQ-ML_SELECT-04	The 3GPP management system shall have a capability to provide a selected ML model to the authorized ML training MnS consumer.	ML model and ML model selection (clause 6.2b.2.3)

Requirement label	Description	Related use case(s)
REQ-ML_TRAIN-MGT-01	The ML training MnS producer shall have a capability allowing an authorized consumer to manage and configure one or more requests for the specific ML model training, e.g. to modify the request or to delete the request.	ML model training requested by consumer (clause 6.2b.2.1), Managing ML model Training Processes (clause 6.2b.2.4)
REQ-ML_TRAIN-MGT-02	The ML training MnS producer shall have a capability allowing an authorized ML training MnS consumer to manage and configure one or more training processes, e.g. to start, suspend or restart the training.	ML model training requested by consumer (clause 6.2b.2.1), Managing ML model training processes (clause 6.2b.2.4)
REQ-ML_TRAIN-MGT-03	3GPP management system shall have a capability to enable an authorized ML training MnS consumer (e.g. the function/model different from the function that generated a request for ML model training) to request for a report on the outcomes of a specific training instance.	Managing ML model training processes (clause 6.2b.2.4)
REQ-ML_TRAIN-MGT-04	3GPP management system shall have a capability to enable an authorized ML training MnS consumer to define the reporting characteristics related to a specific training request or training instance.	Managing ML model training processes (clause 6.2b.2.4)
REQ-ML_TRAIN-MGT-05	3GPP management system shall have a capability to enable the ML training function to report to any authorized ML training MnS consumer about specific ML model training process and/or report about the outcomes of any such ML model training process.	Managing ML model training processes (clause 6.2b.2.4)
REQ-ML_ERROR-01	The 3GPP management system shall enable an authorized consumer of data services (e.g. an ML training function) to request from an MnS producer of data services a Value Quality Score of the data, which is the numerical value that represents the dependability/quality of a given observation and measurement type.	Handling errors in data and ML decisions (clause 6.2b.2.5)
REQ-ML_ERROR-02	The 3GPP management system shall enable an authorized consumer of AI/ML decisions (e.g. a controller) to request ML decision confidence score which is the numerical value that represents the dependability/quality of a given decision generated by an AI/ML inference function.	Handling errors in data and ML decisions (clause 6.2b.2.5)
REQ-ML_ERROR-03	The 3GPP management system shall have a capability to enable an authorized consumer to provide to the ML training MnS producer, a training data quality score, which is the numerical value that represents the dependability/quality of a given observation and measurement type.	Handling errors in data and ML decisions (clause 6.2b.2.5)
REQ-ML_ERROR-04	The 3GPP management system shall enable an MnS producer of ML decisions (e.g. an AI/ML inference function) to provide to an authorized consumer of ML decisions (e.g. a controller) an AI/ML decision confidence score which is the numerical value that represents the dependability/quality of a given decision generated by the AI/ML inference function.	Handling errors in data and ML decisions (clause 6.2b.2.5)
REQ-ML_VLD-01	The ML training MnS producer should have a capability to validate the ML models during the ML model training process and report the performance of the ML models on both the training data and validation data to the authorized consumer.	ML model validation performance reporting (clause 6.2b.2.7)
REQ-ML_VLD-02	The ML training MnS producer should have a capability to report the ratio (in terms of quantity of data samples) of the training data and validation data used during the ML model training and validation process.	ML model validation performance reporting (clause 6.2b.2.7)
REQ-TRAIN_EFF-01	The 3GPP management system should have the capability to allow an authorized consumer to configure an ML training function to report the effectiveness of data used for model training.	Training data effectiveness reporting (clause 6.2b.2.8)
REQ-ML_TRAIN-PM-1	The ML Training MnS producer should have a capability to allow an authorized consumer to get the capabilities about what kind of ML models the ML training function is able to train.	Performance indicator selection for ML model training (clause 6.2b.2.9.2)

Requirement label	Description	Related use case(s)
REQ-ML_TRAIN_PM-2	The ML Training MnS producer should have a capability to allow an authorized consumer to query what performance indicators are supported by the ML model training for each ML model.	ML model performance indicators query and selection for ML model training (clause 6.2b.2.9.3)
REQ-ML_TRAIN_PM-3	The ML Training MnS producer should have a capability to allow an authorized consumer to select the performance indicators from those supported by the ML training function for reporting the training performance for each ML model.	ML model performance indicators query and selection for ML model training (clause 6.2b.2.9.3)
REQ-ML_TRAIN_PM-4	The ML Training MnS producer should have a capability to allow an authorized consumer to provide the performance requirements for the ML model training using the selected the performance indicators from those supported by the ML training function.	MnS consumer policy-based selection of ML model performance indicators for ML model training (clause 6.2b.2.9.4)
REQ-MLKTL-1	The 3GPP management system should have a capability to enable an authorized MnS consumer to discover or request all or part of the available shareable knowledge at a given MLKTL MnS producer.	ML-Knowledge-based Transfer Learning (clause 6.2b.2.10.2.1)
REQ-MLKTL-2	The 3GPP management system should have a capability for an MLKTL MnS producer to provide to an authorized MnS consumer all or part of its available shareable knowledge	ML-Knowledge-based Transfer Learning (clause 6.2b.2.10.2.1)
REQ-MLKTL-3	The 3GPP management system should have a capability enabling an authorized MnS consumer to request a MLKTL MnS producer to initiate and execute a transfer learning instance to a specified ML model or ML-enabled function	ML-Knowledge-based Transfer Learning (clause 6.2b.2.10.2.2)
REQ-MLKTL-4	The 3GPP management system should have a capability to enable an authorized MnS consumer to manage or control the knowledge request or the knowledge process or transfer learning process, e.g. to suspend, re-activate or cancel the ML Knowledge Request; or to adjust the description of the desired knowledge NOTE: An example MnS consumers include an operator or the function that generated the request for available Knowledge	ML-Knowledge-based Transfer Learning (clause 6.2b.2.10.2.2)
REQ-MLKTL-5	The 3GPP management system should have a capability to enable an ML model or ML training function to register available knowledge to a shared knowledge repository, e.g. through a ML Knowledge Registration process	ML-Knowledge-based Transfer Learning (clause 6.2b.2.10.2.2)
REQ-MLKTL-6	The 3GPP management system should have a capability enabling an authorized MnS consumer to request the Knowledge Repository to provide some or all the knowledge available for sharing based on specific criteria	ML-Knowledge-based Transfer Learning (clause 6.2b.2.10.2.2)
REQ-ML_TRAIN_CLUSTER-01	The ML training MnS producer should have a capability for an authorized MnS consumer to request training of a cluster of ML models as per clustering criteria associated to a set of multiple contexts from a previously trained ML model.	ML model training for multiple contexts (clause 6.2b.2.11)
REQ-ML_TRAIN-PRE-01	The ML training MnS producer should have a capability allowing an authorized ML training MnS consumer to request pre-specialized training of a ML model.	ML Pre-specialised training (clause 6.2b.2.12)
REQ- ML_TRAIN-FT-x1	The ML training MnS producer should have a capability to enable an authorised consumer to request the fine-tuning of a pre-specialised trained ML model.	ML Pre-specialised training (clause 6.2b.2.12), ML fine-tuning (clause 6.2b.2.13)
REQ- ML_TRAIN-FT-x2	The ML training MnS producer should have a capability allowing the consumer to specify the training type of an ML model training request such as pre-specialised training, and fine-tuning.	ML Pre-specialised training (clause 6.2b.2.12), ML fine-tuning (clause 6.2b.2.13)
REQ-ML_DIST-TRNG-01	The ML training MnS producer should have a capability allowing and authorized consumer to provide distributed training requirements to the MnS Producer.	Management of Distributed training (clause 6.2b.2.14)

Requirement label	Description	Related use case(s)
REQ-ML_TRAIN_FL-1	The ML training MnS producer should have a capability allowing an authorized consumer to discover the FL roles (FL server or FL client) in Federated learning.	Management of different roles in Federated learning (Clause 6.2b.2.15.2.1)
REQ-ML_TRAIN_FL-2	The ML training MnS producer should have a capability allowing an authorized consumer to provide FL training requirements to the MnS Producer.	Management of different roles in Federated learning (Clause 6.2b.2.15.2.1)
REQ-ML_TRAIN_FL-3	The ML training MnS producer should have a capability allowing an authorized consumer to provide requirements for selecting (including adding and removing) FL clients in Federated learning to the MnS Producer.	Management of different roles in Federated learning (Clause 6.2b.2.15.2.1)
REQ-ML_TRAIN_FL-4	The ML training MnS producer should have a capability allowing an authorized consumer to get the performance of the global ML model on each participating FL Client.	Management of different roles in Federated learning (Clause 6.2b.2.15.2.1)
REQ-ML_TRAIN_FL-5	The ML training MnS producer should have a capability to report the information about the contribution of each FL client to the FL process to MnS consumer.	Management of different roles in Federated learning (Clause 6.2b.2.15.2.1)
REQ-ML_TRAIN_FL-6	The ML training MnS producer should have a capability to report the candidate FL Clients for the FL process.	Management of different roles in Federated learning (Clause 6.2b.2.15.2.1)
REQ-RL_TRAIN_01	The ML training MnS producer should have a capability allowing an authorized MnS consumer to query if RL training is supported.	Enabling Reinforcement learning (6.2b.2.16.2.1)
REQ-RL_TRAIN_02	The ML training MnS producer should have a capability to report RL types (i.e., online RL, offline RL) to an authorized consumer.	Enabling Reinforcement Learning (6.2b.2.16.2.1)
REQ-RL_TRAIN_03	The ML training MnS producer should have a capability to allow an authorized consumer to get the type and scope of the RL environment for which an RL model has been trained.	Exploration in Reinforcement learning (6.2b.2.16.2.2)
REQ-RL_TRAIN_04	The ML training MnS producer should have a capability to allow an authorized consumer to select the type of the RL environment for which an RL model is to be trained.	Exploration in Reinforcement learning (6.2b.2.16.2.2)
REQ-RL_TRAIN_05	The ML training MnS producer should have a capability to allow an authorized consumer to provide the scope of the RL environment for which an RL model is to be trained.	Exploration in Reinforcement learning (6.2b.2.16.2.2)
REQ-RL_TRAIN_06	The ML training MnS producer should have a capability allowing an authorized MnS consumer to provide network performance requirements of performing RL training.	Exploration in Reinforcement learning (6.2b.2.16.2.2)
REQ-RL_TRAIN_07	The ML training MnS producer should have a capability to allow an authorized MnS consumer to specify the configuration range that the RL agent is allowed to explore.	Exploration in Reinforcement learning (6.2b.2.16.2.2)
REQ-RL_TRAIN_08	The ML training MnS producer should have a capability to allow an authorized consumer to provide the allowed scope for the entities to be impacted by the RL actions.	Exploration in Reinforcement learning (6.2b.2.16.2)
REQ-ML_TRAIN_DST-01	The 3GPP management system should enable an authorized consumer to provide information on the training dataset distribution.	Training data statistics (clause 6.2b.2.17)
REQ-ML_TRAIN_DST-02	The 3GPP management system should enable an authorized consumer to provide information on the usage of outliers in the training dataset.	Training data statistics (clause 6.2b.2.17)
NOTE: The performance measurements and KPIs are specific to each type (i.e., the inference type that the ML model supports) of ML model.		

6.2c ML model testing

6.2c.1 Description

After the training and validation, the ML model needs to be tested to evaluate the performance of it when it conducts inference using testing data.

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

6.2c.2 Use cases

6.2c.2.1 Consumer-requested ML model testing

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

The ML model testing is to conduct inference on the tested ML model 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 model testing, the ML testing MnS producer provides the testing report indicating the success or failure of the ML model testing to the consumer. For a successful ML model testing, the testing report contains the testing results, i.e., the inference output for each testing dataset example.

6.2c.2.2 Producer-initiated ML model testing

The ML model testing may also be initiated by the MnS producer, after the ML model 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 model. The consumer may pre-define performance requirements for the ML model testing and allow the MnS producer to decide on whether ML model re-training/validation need to be triggered. ML model re-training may be triggered by the testing MnS producer itself based on the performance requirements supplied by the MnS consumer.

6.2c.2.3 Joint testing of multiple ML models

A group of ML models may work in a coordinated manner for complex use cases.

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

NOTE: This use case is about the ML models testing before deployment.

6.2c.2.4 Performance management for ML model testing

6.2c.2.4.1 Overview

During ML model testing, the performance of ML model needs to be evaluated on testing data. The performance is the degree to which the ML models fulfil the objectives for which they were trained. The related performance indicators need to be collected and analyzed.

6.2c.2.4.2 Performance indicator selection for ML model testing

The ML model testing function may support testing for a single or several ML model algorithms and may support the capability to evaluate each ML model 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 model. The performance indicators for testing mainly include the following aspects:

- ML model testing performance indicators: performance indicators of the ML model 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.

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

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

6.2c.2.4.3 ML model performance indicators query and selection for ML model testing

In a similar way as for training, the MnS producer of ML model 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 for ML model 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 procedure is the same as described in 6.2b.2.9.3 for training.

6.2c.2.4.4 MnS consumer policy-based selection of ML model performance indicators for ML model testing

In a similar way as for training, the MnS consumer for ML model 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 testing. The same description in 6.2b.2.9.4 applies for policy based selection of performance indicators for testing.

6.2c.3 Requirements for ML model testing

Table 6.2c.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 model.	Consumer-requested ML model testing (clause 6.2c.2.1)
REQ-ML_TEST-2	The ML testing MnS producer shall have a capability to trigger the testing of an ML model and allow the MnS consumer to set the policy for the testing.	Producer-initiated ML model testing (6.2c.2.2)
REQ-ML_TEST-3	The ML testing MnS producer shall have a capability to report the performance of the ML model when it performs inference on the testing data.	Consumer-requested ML model testing (clause 6.2c.2.1), and Producer-initiated ML model testing (clause 6.2c.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 models.	Joint testing of multiple ML models (clause 6.2c.2.3)

Requirement label	Description	Related use case(s)
REQ-ML_TEST_PM-1	The ML testing MnS producer should have a capability to allow an authorized consumer to get the capabilities about what kind of ML models the ML testing function is able to test.	Performance indicator selection for ML model testing (clause 6.2c.2.4.2)
REQ-ML_TEST_PM-2	The ML testing MnS producer should have a capability to allow an authorized consumer to query what performance indicators are supported by the ML testing function for each ML model.	Performance indicator selection for ML model testing (clause 6.2c.2.4.2)
REQ-ML_TEST_PM-3	The ML testing MnS producer should have a capability to allow an authorized consumer to select the performance indicators from those supported by the ML testing function for reporting the testing performance for each ML model.	Performance indicator selection for ML tra (clause 6.2c.2.4.2)

6.3 AI/ML inference emulation

6.3.1 Description

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

6.3.2 Use cases

6.3.2.1 AI/ML inference emulation

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

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

- request an inference emulation function to provide emulation reports; and
- to receive the results from running inference through an AI/ML inference emulation environment available at the emulation MnS producer.

6.3.2.2 ML inference emulation environment selection

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

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

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

6.3.3 Requirements for Managing AI/ML inference emulation

Table 6.3.3-1

Requirement label	Description	Related use case(s)
REQ-AI/ML_EMUL-1:	The MnS producer for AI/ML inference emulation should have a capability enabling an authorized MnS consumer to receive reporting about the ML inference emulation.	AI/ML Inference emulation (clause 6.3.2.1)
REQ-AI/ML_EMUL-2:	The MnS producer for AI/ML inference emulation should have a capability enabling an authorized MnS consumer to request an inference emulation function to provide inference emulation reports on an ML model or inference Function.	AI/ML Inference emulation (clause 6.3.2.1)
REQ-EMUL_SEL-1	The MnS producer for AI/ML inference emulation should have a capability enabling an authorized MnS consumer to select the emulation environment.	ML inference emulation environment selection (clause 6.3.2.2)

6.4 ML model deployment

6.4.1 ML model loading

6.4.1.1 Description

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

6.4.1.2 Use cases

6.4.1.2.1 Consumer requested ML model loading

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

6.4.1.2.2 Control of producer-initiated ML model loading

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

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

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

6.4.1.2.3 ML model registration

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

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

6.4.1.3 Requirements for ML model loading

Table 6.4.1.3-1

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

6.5 AI/ML inference

6.5.1 AI/ML inference performance management

6.5.1.1 Description

During AI/ML inference, the performance of the AI/ML inference function and ML model 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 model re-training, ML model testing, or re-deployment.

6.5.1.2 Use cases

6.5.1.2.1 AI/ML inference performance evaluation

During AI/ML inference, the AI/ML inference function (including e.g., MDAF, NWDAF or RAN functions) uses one or more ML models for inference to generate the AI/ML inference output. The performance of a running ML model

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 model 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 model, 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 model, start retraining, change the running model with a new one, etc) can be taken to avoid generating the inaccurate inference output.

To monitor the performance during AI/ML inference, 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 model.

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

Evaluation and management of the performance of an ML model is needed during AI/ML inference. 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 model 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 model. 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)

Requirement label	Description	Related use case(s)
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 model 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.	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 model(s) 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 model(s) is decreasing.

The MnS consumer may request the AI/ML Inference MnS producer to use an updated ML model(s) 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 model(s) updated, which may be loading the already trained ML model(s) or may lead to requesting to train/re-train the ML model(s) by utilizing the ML training MnS.

6.5.2.2 Use cases

6.5.2.2.1 Availability of new capabilities or ML models

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 model. 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 model update

When the inference capabilities of AI/ML inference functions degenerate, the typical action may be to trigger ML model re-training of the constituent ML models. 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 models. 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 models, i.e., to have the ML models 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 models 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 models, or it may trigger one or more remote or local AI/ML-related processes (including ML model 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 models 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 models (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 models of the respective network function are updated with/to the specific set of newly available AI/ML capabilities	Availability of new capabilities or ML models (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 models using a specific version of newly available AI/ML capabilities or ML models or using AI/ML capabilities or ML models with requirements (e.g., the minimum achievable performance after updating, the maximum time taken to complete the update, etc)	Triggering ML model 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 models	Triggering ML model 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 models or set of AI/ML capabilities	Triggering ML model 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/ model 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 model 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 models, 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 models 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 models and to map those capabilities to the desired logic.

6.5.3.2 Use cases

6.5.3.2.1 Identifying capabilities of ML models

Network functions, especially network automation functions, may need to rely on capabilities of ML models 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 models are used for inference per the corresponding MDA capability". 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 models that are employed by AI/ML inference functions.

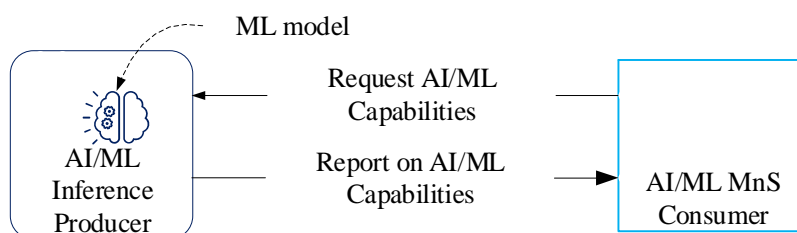


Figure 6.5.3.2.1-1: Request and reporting on AI/ML inference capabilities

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 models

Besides the discovery of the capabilities of ML models, services are needed for mapping the ML models and capabilities. In other words, instead of the consumer discovering specific capabilities, the consumer may want to know the ML models 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 models 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 model to complex intents that may require an intricate orchestration of multiple ML models. For simple intents, it may be easy to map the execution logic to one or multiple ML models. For complex intents, it may be required to employ multiple ML models along with a corresponding functionality that manages their interrelated execution. The usage of the ML models requires the awareness of their capabilities and interrelations.

Moreover, given the complexity of the required mapping to the multiple ML models, services should be supported to provide the mapping of ML models 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 models available within the AI/ML inference producer.	Identifying capabilities of ML models (clause 6.5.3.2.1)

Requirement label	Description	Related use case(s)
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 model 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 model can undertake optimization or control; the configuration parameters on the stated object or object types, which the ML model optimizes or controls to achieve the desired outcomes; and the network metrics which the ML model optimizes through its actions.	Identifying capabilities of ML models (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 model as an analysis described as a tuple <object(s), characteristics> with the entries respectively indicating: the object or object types for which the ML model can undertake analysis; and the network characteristics (related to the stated object or object types) for which the ML model produces analysis	Identifying capabilities of ML models (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 models.	Mapping of the capabilities of ML models (clause 6.5.3.2.2)

6.5.4 AI/ML inference capability configuration management

6.5.4.1 Description

The objective of AI/ML for NG-RAN is to improve network performance and user experience, which can yield further insights, e.g., for Network Energy Saving, Load Balancing, Mobility Optimization as defined in TS 38.300 [16]. According to the principles defined in clause 16.20.2, clause 15.4, clause 15.5 in TS 38.300 [16], either “AI/ML model training is located in the OAM and AI/ML model inference is located in the NG-RAN node;” or “AI/ML model training and AI/ML model inference are both located in the NG-RAN node”. To manage the network performance, the AI/ML inference and the associated ML model(s) may need to be managed and configured to conduct inference in the 5G system in alignment 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

The MnS consumer monitors the network performance and determines whether, and when, to activate or deactivate an AI/ML inference related to an AI/ML-based Distributed Network Energy Saving. The activation and deactivation actions for AI/ML inference related to an AI/ML-based Distributed Network Energy Saving performed by the MnS producer may also be triggered by policies provided by the consumer.

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

The MnS consumer monitors the network performance and determines whether activation or deactivation of an AI/ML inference related to an AI/ML-based Distributed Mobility Optimization is required. The activation and deactivation actions performed 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

The MnS consumer monitors the network performance and determines whether, and when, to activate or deactivate an AI/ML-based Distributed Load balancing. The activation and deactivation actions performed by the MnS producer may also be triggered by 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 manage the ML model and/or AI/ML Inference Function related to Distributed Energy Saving functions.	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 manage the ML model and/or AI/ML Inference Function related to Distributed Mobility Optimization functions.	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 ML model and/or AI/ML Inference Function related to Distributed the Load Balancing functions.	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 AI/ML Inference History

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 model 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 outputs derived by the ML model and the contexts, e.g., network resources, time periods, traffic conditions, etc. The inference history output should be reported by the MnS Producer to the MnS Consumer.

The inferences may need to be tracked for future reference, e.g., to evaluate the appropriateness/usefulness of the inference outcome for those contexts or to evaluate degradations in the ML model'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 model. 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.

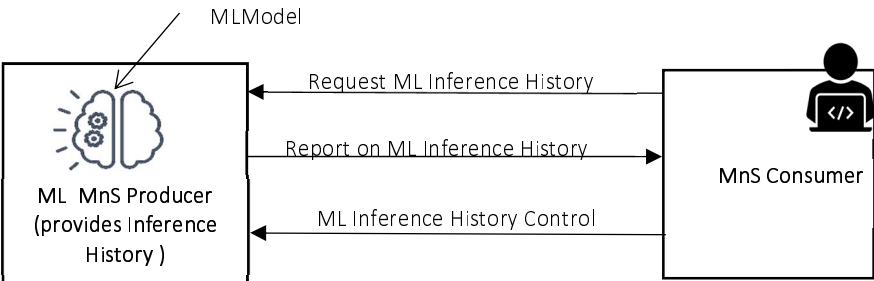


Figure 6.5.5.2.1-1: Example use and control of AI/ML inference history request and reporting.

6.5.5.3 Requirements for AI/ML Inference History

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 receive the inference history of a specific ML model.	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 of historical inference outputs related to a specific instance of an ML model.	AI/ML Inference History - tracking inferences and context (clause 6.5.5.2.1)

6.5.6 Managing ML models in use in a live network

6.5.6.1 Description

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

6.5.6.2 Use cases

6.5.6.2.1 Handling of underperforming ML trained models in live networks

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

6.5.6.2.2 Performance monitoring of Network Functions with ML trained models in live networks

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

AI/ML supported functions and NG-RAN AI/ML supported functions and would need to be collectable by the network operator.

6.5.6.3 Requirements for Managing ML models in use in a live network

Table 6.5.6.3-1

Requirement label	Description	Related use case(s)
REQ-AI/ML_INF-LIVES-01	The 3GPP management system should have a capability enabling an authorized consumer to identify ML model(s) that may cause performance measurement and/or KPI degradation.	Handling of underperforming ML trained models in live networks (6.5.6.2.1)
REQ-AI/ML_INF-LIVES-02	The 3GPP management system should be able to collect network related performance data pertaining to Network Functions actively utilizing ML models.	Performance monitoring of Network Functions with ML trained models in live networks (6.5.6.2.2)

6.5.7 AI/ML inference explainability

6.5.7.1 Description

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

6.5.7.2 Use cases

6.5.7.2.1 Management of explanation in AI/ML inference

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

6.5.7.3 Requirements for AI/ML inference explainability management

Table 6.5.7.3-1

Requirement label	Description	Related use case(s)
REQ-ML-INF-EXP-01	The 3GPP management system should have a capability to report the explanations for AI/ML inference to an authorized consumer.	Management of explanation in AI/ML inference (clause 6.5.7.2.1)

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
3GPP TS 28.541 [18], IOC, ANLFFunction	ANLFFunction
3GPP TS 28.541 [18], IOC, DESManagementFunction	DESManagementFunction
3GPP TS 28.541 [18], IOC, DMROFunction	DMROFunction
3GPP TS 28.541 [18], IOC, DLBOFunction	DLBOFunction
3GPP TS 28.541 [18], IOC, LMFFunction	LMFFunction

7.2 Void

7.2a Common information model definitions for AI/ML management

7.2a.1 Class diagram

7.2a.1.1 Relationships

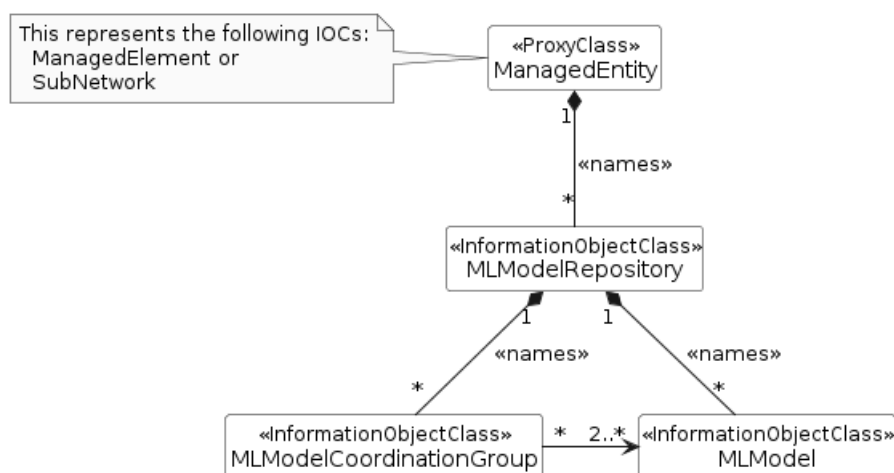


Figure 7.2a.1.1-1: Relations for common information models for AI/ML management

7.2a.1.2 Inheritance

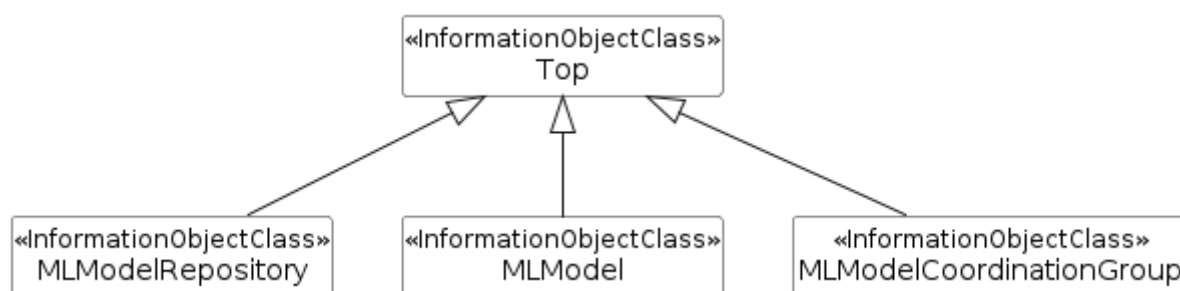


Figure 7.2a.1.2-1: Inheritance Hierarchy for common information models for AI/ML management

7.2a.2 Class definitions

7.2a.2.1 MLModel

7.2a.2.1.1 Definition

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

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

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

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

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

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

7.2a.2.1.2 Attributes

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

Table 7.2a.2.1.2-1

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

7.2a.2.1.3 Attribute constraints

Table Void

None.

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 MLModelRepository

7.2a.2.2.1 Definition

The IOC MLModelRepository represents the repository that contains the ML models. It is name-contained by SubNetwork or ManagedElement.

This MLModelRepository instance can be created by the system (MnS producer) or pre-installed.

The MLModelRepository MOI may contain one or more MLModel(s).

7.2a.2.2.2 Attributes

The MLModelRepository IOC inherited from Top IOC (defined in TS 28.622 [12]).

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 MLModelCoordinationGroup

7.2a.2.3.1 Definition

This IOC represents the group of ML models, which can be trained and tested jointly and used to perform inference in a coordinated way. It is name-contained by MLModelRepository.

This MLModelCoordinationGroup instance is created by the system (MnS producer) or pre-installed. The MnS consumer can request the System to delete the MLModelCoordinationGroup MOI.

One ML model may have dependencies on one or more of the other ML models of the same group.

One group is associated with at least two ML models.

7.2a.2.3.2 Attributes

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

Table 7.2a.2.3.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
Attribute related to role					
memberMLModelRefList	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 model 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].

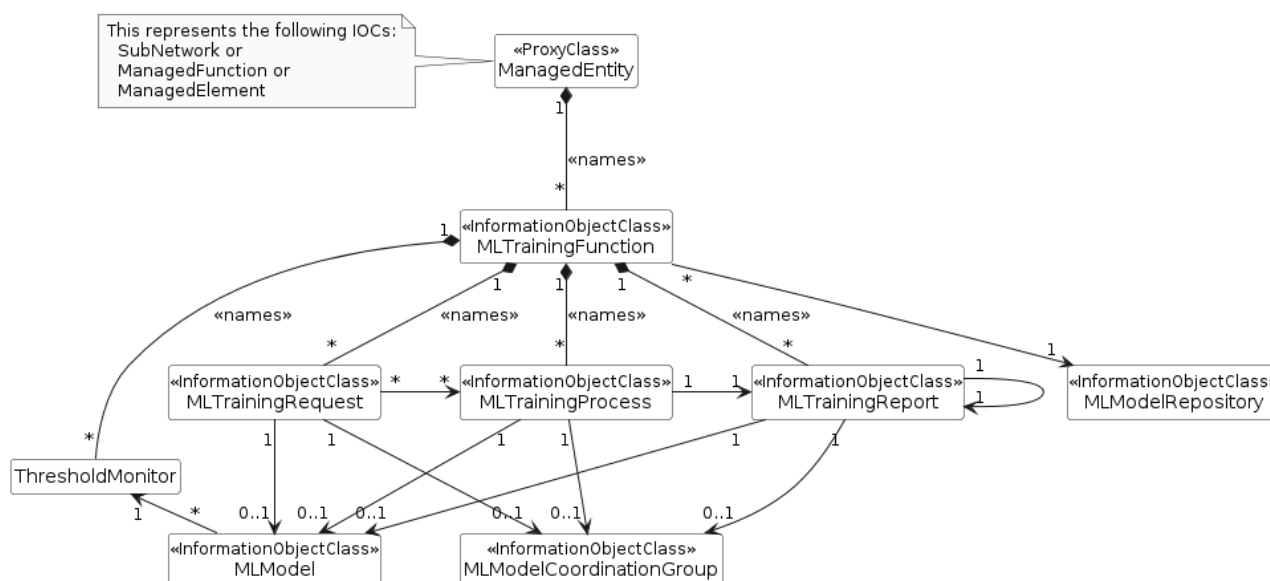


Figure 7.3a.1.1.1-1: NRM fragment for ML model training

7.3a.1.1.2 Inheritance

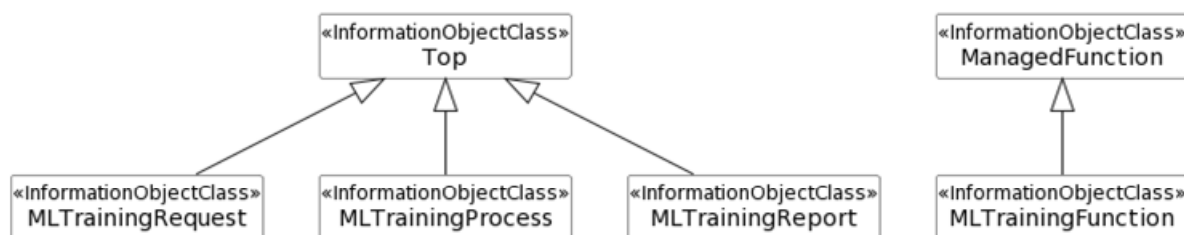


Figure 7.3a.1.1.2-1: Inheritance Hierarchy for ML model training related NRMs

7.3a.1.2 Class definitions

7.3a.1.2.1 MLTrainingFunction

7.3a.1.2.1.1 Definition

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

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

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

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

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

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

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

7.3a.1.2.1.2 Attributes

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

Table 7.3a.1.2.1.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
<code>supportedLearningTechnology</code>	M	T	F	F	T
<code>flParticipationInfo</code>	CM	T	F	F	T
<code>mLKnowledge</code>	O	T	F	F	T
<code>mLTrainingType</code>	M	T	F	F	T
Attribute related to role					
<code>mLModelRepositoryRef</code>	M	T	F	F	T

7.3a.1.2.1.3 Attribute constraints

Table 7.3a.1.2.1.3-1

Name	Definition
<code>flParticipationInfo</code>	Condition: FL is supported.

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 triggered by the ML training MnS consumer.

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

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

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

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

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

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

- collects (more) data for training, if the training data are not available or the data are available but not sufficient for the training;
- prepares and selects the required training data, with consideration of the MnS consumer's request provided candidate training data if any. The ML training MnS producer may examine the MnS consumer's provided candidate training data and select none, some or all of them for training. In addition, the ML training MnS producer may select some other training data that are available in order to meet the MnS consumer's requirements for the ML model training;
- trains the `MLModel` using the selected and prepared training data.

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

- The attribute values are "NOT_STARTED", "IN_PROGRESS", "SUSPENDED", "FINISHED", and "CANCELLED".
- When value turns to "IN_PROGRESS", the ML training MnS producer instantiates one or more `MLTrainingProcess` MOI(s) representing the training process(es) being performed per the request and notifies the MLT MnS consumer(s) who subscribed to the notification.

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

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

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

The attribute `FLRequirement` indicates the requirements for the `MLTrainingFunction` playing the role of FL server to coordinate the training of an `MLModel` using Federated learning.

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

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

7.3a.1.2.2.2 Attributes

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

Table 7.3a.1.2.2.1-1

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

7.3a.1.2.2.3 Attribute constraints

Table 7.3a.1.2.2.3-1

Name	Definition
aIMLInferenceName	Condition: Any of the following training types are supported: Initial training, pre-specialized training, fine-tuning.
mLModelCoordinationGroupRef	Condition: ML model joint training is supported.
mLKnowledgeName	Condition: ML-knowledge-based transfer learning is supported. Knowledge is indicated only if candidateTrainingDataSource is not indicated
rLRequirement	Condition: Reinforcement learning is supported.
fLRequirement	Condition: FL is supported
expectedInferenceScope	Condition: The MLTrainingRequest is for an ML model pre-specialised training.

7.3a.1.2.2.4 Notifications

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

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` is associated with one `MLModel` or one `MLModelCoordinationGroup`.

The `MLTrainingReport` instance is created by the training MnS producer automatically when creating an `MLTrainingRequest` instance.

The `MLTrainingReport` MOI is contained under one `MLTrainingFunction` MOI.

7.3a.1.2.3.2 Attributes

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

Table 7.3a.1.2.3.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
usedConsumerTrainingData	CM	T	F	F	T
modelConfidenceIndication	O	T	F	F	T
modelPerformanceTraining	M	T	F	F	T
areNewTrainingDataUsed	M	T	F	F	T
modelPerformanceValidation	O	T	F	F	T
fLReportPerClient	CM	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
mLModelGeneratedRef	M	T	F	F	T
mLModelCoordinationGroupGeneratedRef	CM	T	F	F	T
mLModelRef	M	T	F	F	T

7.3a.1.2.3.3 Attribute constraints

Table 7.3a.1.2.3.3-1

Name	Definition
trainingRequestRef	Condition: The MLTrainingReport MOI represents the report for the ML model training that was requested by the MnS consumer (via MLTrainingRequest MOI).
lastTrainingRef	Condition: The MLTrainingReport MOI represents the report for the ML model training that was not ML model initial training (i.e. the model has been trained before).
mLModelCoordinationGroupGeneratedRef	Condition: The MLTrainingReport MOI represents the report for ML model joint training.
fLReportPerClient	Condition: FL is supported.

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 model training process. When a ML model training process starts, an instance of the MLTrainingProcess is created by the MnS Producer and notification is sent to MnS consumer who has subscribed to it. The MnS producer can delete the MLTrainingProcess instance whose attribute status equals to "FINISHED" or "CANCELLED" automatically.

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

For each MLModel under training, a MLTrainingProcess is instantiated, i.e. an MLTrainingProcess is associated with one MLModel or one MLModelCoordinationGroup. 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 `MLModel`, although the `MLTrainingProcess` may be associated to only one `MLModel`. For example, the `MLTrainingProcess` may be triggered to start with a specific version of the `MLModel` and multiple `MLTrainingProcess` instances may be triggered for different versions of the `MLModel`. In either case the `MLTrainingProcess` instances are still associated with the same `MLModel` but are managed separately from the `MLModel`.

Each `MLTrainingProcess` has a `priority` that may be used to prioritize the execution of different `MLTrainingProcess` instances.

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 model 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

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

Table 7.3a.1.2.4.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
<code>priority</code>	M	T	T	F	T
<code>terminationConditions</code>	M	T	T	F	T
<code>progressStatus</code>	M	T	F	F	T
<code>cancelProcess</code>	O	T	T	F	T
<code>suspendProcess</code>	O	T	T	F	T
Attribute related to role					
<code>trainingRequestRef</code>	CM	T	F	F	T
<code>trainingReportRef</code>	M	T	F	F	T
<code>mLModelGeneratedRef</code>	CM	T	F	F	T
<code>mLModelRef</code>	M	T	F	F	T
<code>participatingFLClientRefList</code>	CM	T	F	F	T

7.3a.1.2.4.3 Attribute constraints

Table 7.3a.1.2.4.3-1

Name	Definition
trainingRequestRef	Condition: The MLTrainingReport MOI represents the report for the ML model training that was requested by the training MnS consumer (via MLTrainingRequest MOI).
mLModelGeneratedRef	Condition: The MLTrainingProcess MOI is instantiated to retrain an existing MLModel.
participatingFLClientRefList	Condition: FL is supported.

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.1b Information model definitions for ML model testing

7.3a.1b.1 Class diagram

7.3a.1b.1.1 Relationships

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

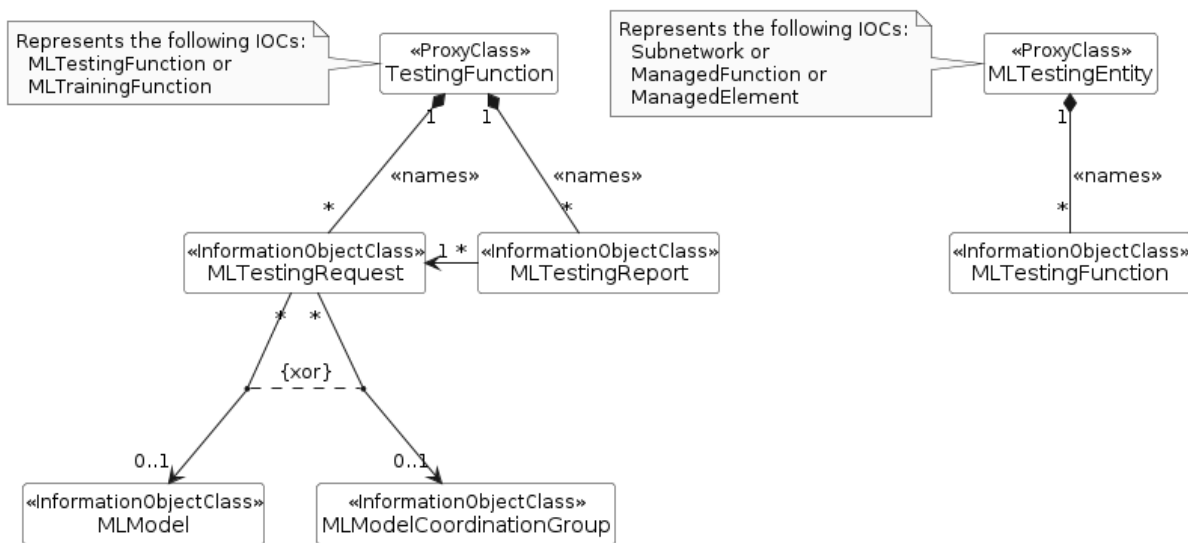


Figure 7.3a.1b.1.1-1: NRM fragment for ML model testing

7.3a.1b.1.2 Inheritance

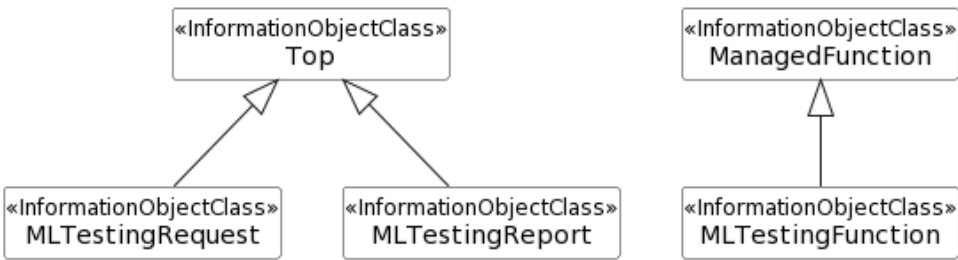


Figure 7.3a.1b.1.2-1: Inheritance Hierarchy for ML model testing related NRMs

7.3a.1b.2 Class definitions

7.3a.1b.2.1 MLTestingFunction

7.3a.1b.2.1.1 Definition

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

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

In case the ML model 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 model testing.

The model represented by MLTestingFunction MOI supports testing of one or more MLModel(s).

7.3a.1b.2.1.2 Attributes

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

Table 7.3a.1b.2.1.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
Attribute related to role					
mLModelRef	M	T	F	F	F

7.3a.1b.2.1.3 Attribute constraints

None.

7.3a.1b.2.1.4 Notifications

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

7.3a.1b.2.2 MLTestingRequest

7.3a.1b.2.2.1 Definition

The IOC MLTestingRequest represents the ML model testing request that is triggered by the ML testing MnS consumer.

To trigger the ML model testing process, ML testing MnS consumer needs to create MLTrainingRequest.

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

In case the request is accepted, the ML testing MnS producer decides when to start the ML model 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 `MLModel` by performing inference using the selected testing data, and
- reports the performance of the `MLModel` 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".

The ML testing MnS producer shall delete the corresponding `MLTestingRequest` instance in case of the status value turns to "FINISHED" or "CANCELLED". The MnS producer may notify the status of the request to MnS consumer before deleting `MLTestingRequest` instance.

7.3a.1b.2.2.2 Attributes

The `MLTestingRequest` IOC includes attributes inherited from Top IOC (defined in TS 28.622 [12]) and the following attributes: **Table 7.3a.1b.2.2.2-1**

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
<code>requestStatus</code>	M	T	F	F	T
<code>cancelRequest</code>	O	T	T	F	T
<code>suspendRequest</code>	O	T	T	F	T
Attribute related to role					
<code>mLModelRef</code>	CM	T	F	F	T
<code>mLModelCoordinationGroupRef</code>	CM	T	F	F	T

7.3a.1b.2.2.3 Attribute constraints

Table 7.3a.1b.2.2.3-1

Name	Definition
<code>mLModelRef</code>	Condition: Testing of a single ML model is supported.
<code>mLModelCoordinationGroupRef</code>	Condition: Joint testing is supported.

7.3a.1b.2.2.4 Notifications

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

7.3a.1b.2.3 MLTestingReport

7.3a.1b.2.3.1 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 model testing.

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

The `MLTestingReport` instance is created by the ML testing MnS producer and notification is sent to ML testing Consumer who has subscribed to it.

7.3a.1b.2.3.2 Attributes

The `MLTestingReport` represents the report for the ML model testing that was requested by the MnS consumer (via `MLTestingRequest` MOI). The IOC includes attributes inherited from Top IOC (defined in TS 28.622 [12]) and the following attributes:

Table 7.3a.1b.2.3.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
modelPerformanceTesting	M	T	F	F	T
mLTestingResult	M	T	F	F	T
Attribute related to role					
testingRequestRef	CM	T	F	F	T

7.3a.1b.2.3.3 Attribute constraints

Table 7.3a.1b.2. 3.3-1

Name	Definition
testingRequestRef	Condition: Report on testing requested by the MnS consumer is supported.

7.3a.1b.2.3.4 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 AI/ML inference emulation

7.3a.2.1 Class diagram

7.3a.2.1.1 Relationships

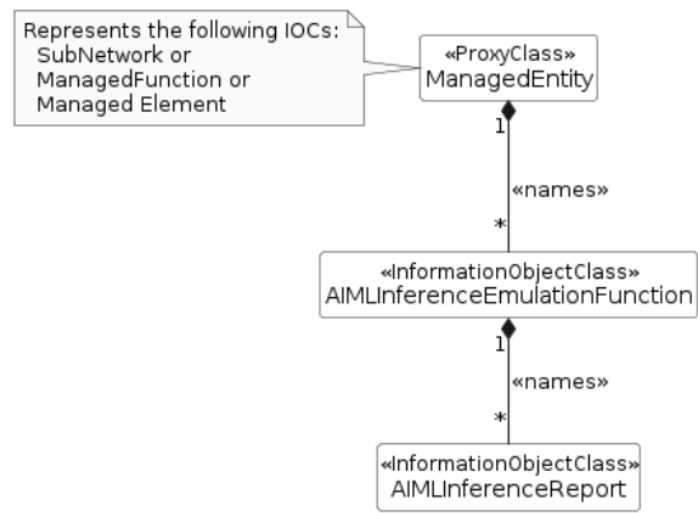


Figure 7.3a.2.1.1-1: NRM fragment for AI/ML inference emulation control

7.3a.2.1.2 Inheritance

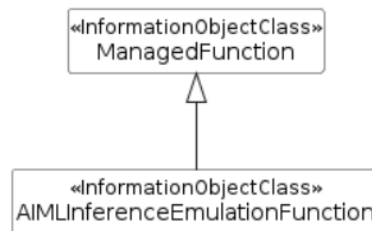


Figure 7.3a.2.1.2-1: AI/ML inference emulation Inheritance Relations

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.

This AIMLINferenceEmulationFunction instance is created by the system (AI/ML inference emulation MnS producer) or pre-installed, it can only be deleted by the system.

An AIMLINferenceEmulationFunction may be associated with one or more MLModel(s). AIMLINferenceFunction 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 inherited from ManagedFunction IOC (defined in TS 28.622 [12]).

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 model deployment

7.3a.3.1 Class diagram

7.3a.3.1.1 Relationships

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

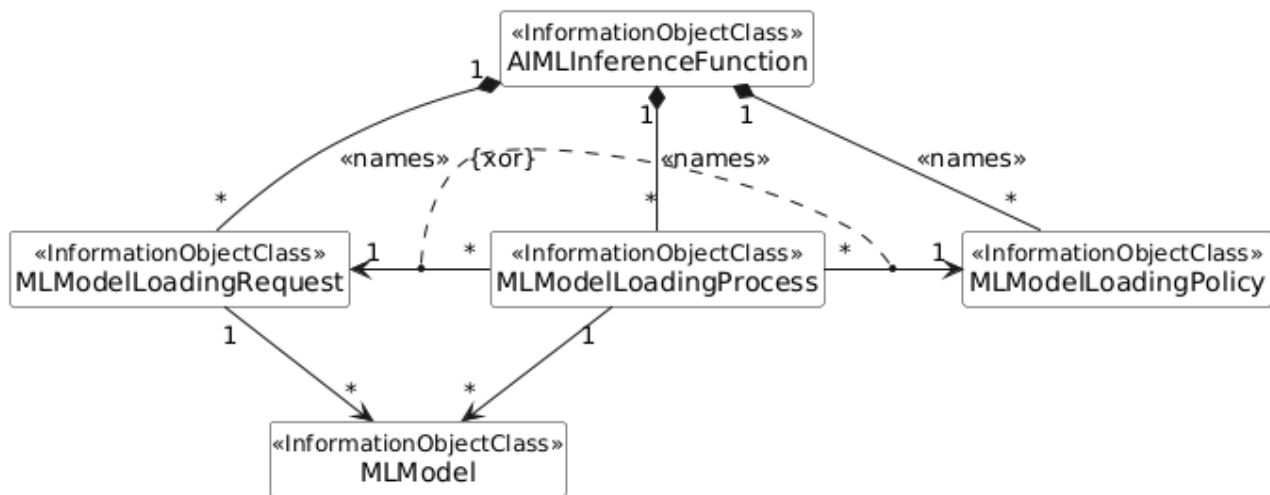


Figure 7.3a.3.1.1-1: NRM fragment for ML model loading

7.3a.3.1.2 Inheritance

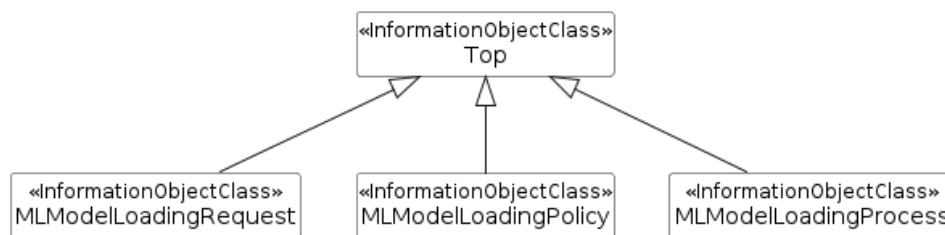


Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML model loading related NRMs

7.3a.3.2 Class definitions

7.3a.3.2.1 MLModelLoadingRequest

7.3a.3.2.1.1 Definition

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

To trigger the ML model loading process, MnS consumer has to create `MLModelLoadingRequest` object instances on the MnS producer.

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 `MLModelLoadingProcess` MOI(s) representing the loading process(es) being performed per the request and notifies the MnS consumer(s) who subscribed to the notification.

The MnS producer shall delete the corresponding `MLModelLoadingRequest` instance in case of the status value turns to "FINISHED" or "CANCELLED".

7.3a.3.2.1.2 Attributes

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

Table 7.3a.3.2.1.2-1

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					
mLModelToLoadRef	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 MLModelLoadingPolicy

7.3a.3.2.2.1 Definition

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

To specify ML model loading policy for one or multiply ML models, MnS consumer needs to create MLModelLoadingPolicy object instances.

To remove ML model loading policy for one or multiply ML models, MnS consumer needs to delete MLModelLoadingPolicy object instances.

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

7.3a.3.2.2.2 Attributes

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

Table 7.3a.3.2.2.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
aIMLInferenceName	CM	T	T	F	T
policyForLoading	M	T	T	F	T
Attribute related to role					
mLModelRef	CM	T	F	F	F

7.3a.3.2.2.3 Attribute constraints

Table 7.3a.3.2.2.3-1

Name	Definition
aIMLInferenceName	Condition: The ML model loading policy is related to an initially trained ML model.
mLModelRef	Condition: The ML model loading policy is related to a re-trained ML model.

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 MLModelLoadingProcess

7.3a.3.2.3.1 Definition

This IOC represents the ML model loading process.

For the consumer requested ML model loading, one or more MLModelLoadingProcess MOI(s) may be instantiated for each ML model loading request presented by the MLModelLoadingRequest MOI.

For the producer-initiated ML model loading, one or more MLModelLoadingProcess MOI(s) may be instantiated and associated with each MLModelLoadingPolicy MOI.

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

The "progressStatus" attribute represents the status of the ML model 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 model 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 MLModel under each MOI of the target inference function(s).

When a ML model loading process starts, an instance of the MLModelLoadingProcess is created by the MnS Producer and notification is sent to MnS consumers who have subscribed to it. The MnS producer can delete the MLModelLoadingProcess instance whose attribute status equals to "FINISHED" or or "CANCELLED" automatically.

7.3a.3.2.3.2 Attributes

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

Table 7.3a.3.2.3.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
progressStatus	M	T	F	F	T
cancelProcess	O	T	T	F	T
suspendProcess	O	T	T	F	T
Attribute related to role					
mLModelLoadingRequestRef	CM	T	F	F	T
mLModelLoadingPolicyRef	CM	T	F	F	T
loadedMLModelRef	M	T	F	F	T

7.3a.3.2.3.3 Attribute constraints

Table 7.3a.3.2.3.3-1

Name	Definition
mLModelLoadingRequestRef	Condition: The mLModelLoadingProcess MOI is corresponding to the ML model loading requested by the MnS consumer.
mLModelLoadingPolicyRef	Condition: The mLModelLoadingProcess MOI is corresponding to the ML model 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 AI/ML inference

7.3a.4.1 Class diagram

7.3a.4.1.1 Relationships

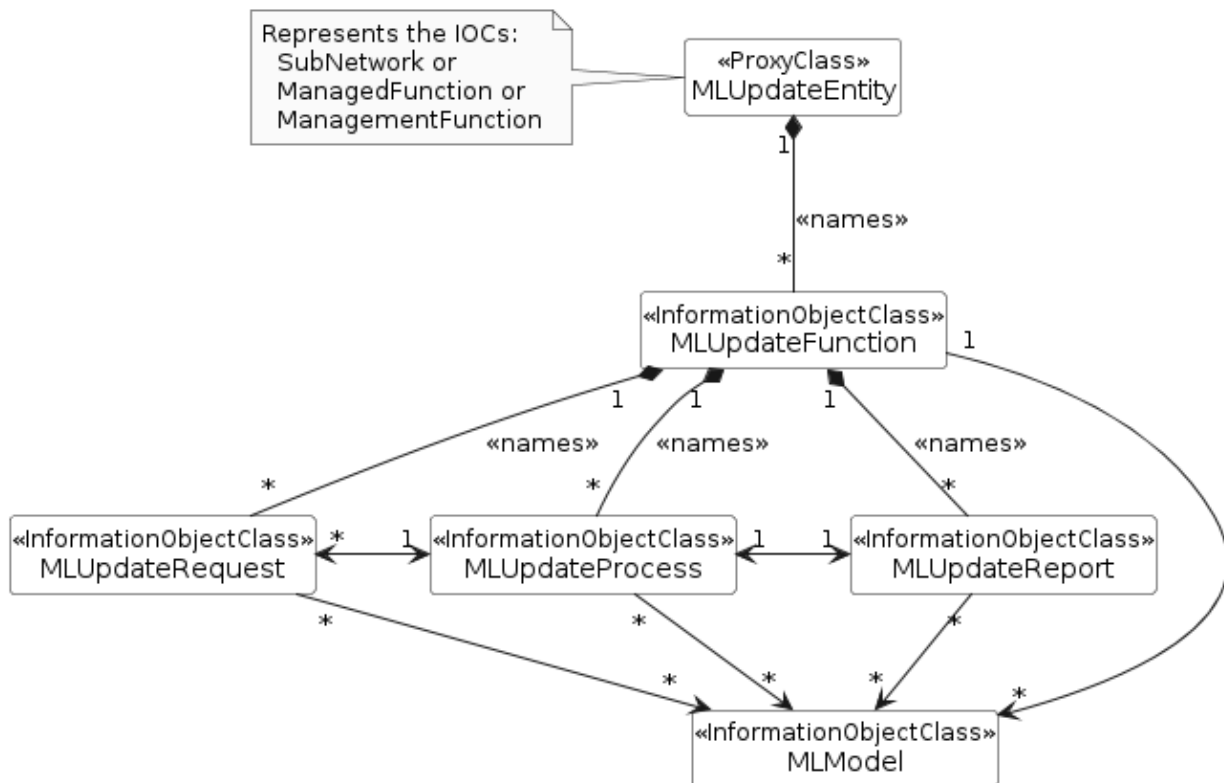
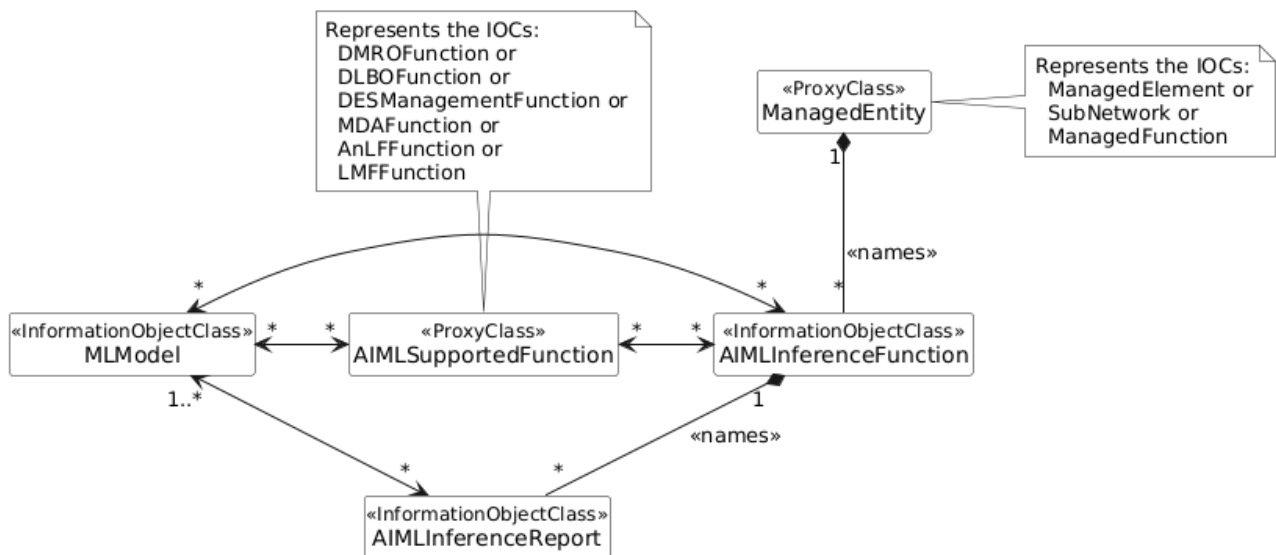


Figure 7.3a.4.1.1-1: NRM fragment for ML update



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

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

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

7.3a.4.1.2 Inheritance

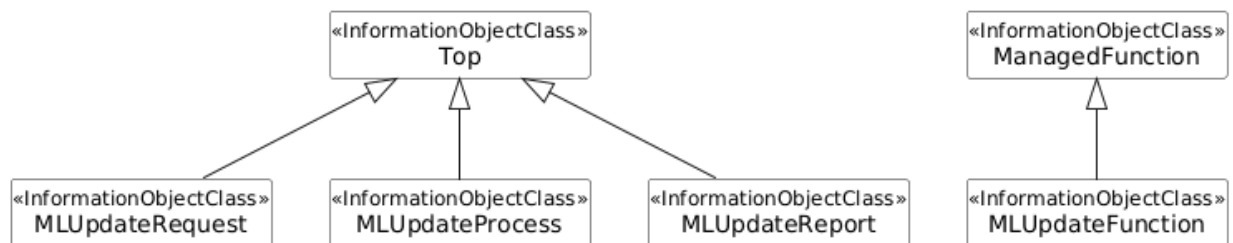


Figure 7.3a.4.1.2-1: Inheritance Hierarchy for ML update related NRMs

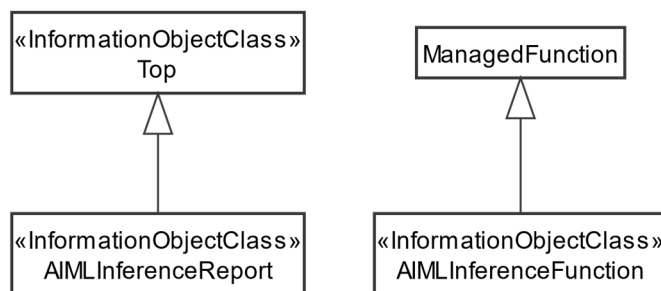


Figure 7.3a.4.1.2-2: Inheritance Hierarchy for AI/ML inference function

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.

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

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 models .

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					
mLModelRef	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 MLModel
- 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 indicate the maximum time that should be taken to complete the update.

To trigger the ML update process, MnS consumer needs to create MLUpdateRequest instances on the MnS producer.

The MnS producer shall delete the corresponding MLUpdateRequest instance in case of the status value turns to "FINISHED" or "CANCELLED".

7.3a.4.2.2.2 Attributes

The MLUpdateRequest IOC includes attributes inherited from Top IOC (defined in TS 28.622 [12]) 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	F	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
mLModelRefList	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 MLModel(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 MLModel.

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

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

When a ML update process starts, an instance of the `MLUpdateProcess` is created by the MnS Producer and informed to MnS consumer who has subscribed to it. The MnS producer can delete the `MLUpdateProcess` instance whose attribute status equals to "FINISHED" or "CANCELLED".

7.3a.4.2.3.2 Attributes

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

Table 7.3a.4.2.3.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
<code>cancelProcess</code>	O	T	T	F	T
<code>suspendProcess</code>	O	T	T	F	T
<code>progressStatus</code>	M	T	F	F	T
Attributes related to Role					
<code>mLModelRefList</code>	M	T	F	F	T
<code>mLUpdateRequestRefList</code>	M	T	F	F	T
<code>mLUpdateReportRef</code>	M	T	F	F	T

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 `MLUpdateReport`.

- The ML update process may generate one or more `MLUpdateReport(s)`.
- Each `MLUpdateReport` is associated to one or more `MLModel(s)` to indicate ML models 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 `MLModel(s)` or `MLUpdateProcess(s)` that themselves are associated with `MLModel(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`.

The `MLUpdateReport` instance can be created by the MnS producer when creating an `MLUpdateRequest` instance.

When the MnS producer delete a `MLUpdateRequest` instance, the corresponding `MLUpdateReport` instance is also deleted by MnS producer.

7.3a.4.2.4.2 Attributes

The MLUpdateReport IOC includes attributes inherited from Top IOC (defined in TS 28.622 [12]) and the following 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					
mLModelRefList	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.

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

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

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

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

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

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

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

7.3a.4.2.5.2 Attributes

The AIMLInferenceFunction IOC includes attributes inherited from ManagedFunction IOC (defined in TS 28.622 [12]) and the following 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
mLModelRefList	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 `MLModel`.

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

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

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

7.3a.4.2.6.2 Attributes

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

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

7.3a.4.2.6.3 Attribute constraints

None.

7.3a.4.2.6.4 Notifications

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

7.4 Data type definitions

7.4.1 ModelPerformance <<dataType>>

7.4.1.1 Definition

This data type specifies the performance of an ML model when performing training and 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 MLModel. 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
inferenceModelRef	Condition: The is supported for expectedRunTimeContext 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 model. 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	isNotifiable
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	Condition: if the performance indicator named performanceIndicatorName is applicable for training, the isSupportedforTraining must be stated
isSupportedForTesting	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 MLModel (s) and may indicate the one or more MLModel (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.

7.4.5.2 Attributes

The AvailMLCapabilityReport includes the following attributes:

Table 7.4.5.2-1 Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
availMLCapabilityReportID	M	T	F	F	T
mLCapabilityVersionId	M	T	F	F	T
expectedPerformanceGains	O	T	F	F	T
Attributes related to Role					
mLModelRef	M	T	F	F	T

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	isNotifiable
thresholdList	M	T	T	F	T
managedActivationScope	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	Condition: if the sub scope is per list of managed elements (e.g., DN list)
timeWindow	Condition: if the sub scope is per list of time window.
geoPolygon	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 model can make inference for. The capabilityName 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
aIMLInferenceName	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. The inference output may include inference explanation information of the ML model during Inference.

7.4.9.2 Attributes

The InferenceOutput includes the following attributes:

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
inferenceOutputId	M	T	F	F	T
aiMLInferenceName	M	T	F	F	T
inferenceOutputTime	M	T	F	F	T
inferencePerformance	O	T	F	F	T
inferenceExplanationInfo	O	T	F	F	T
outputResult	M	T	F	F	T

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

7.4.9.3 Attribute constraints

None.

7.4.9.4 Notifications

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

7.4.10 AIMLInferenceName <<choice>>

7.4.10.1 Definition

This <<choice>> represents the type of inference that the ML model supports.

7.4.10.2 Attributes

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
CHOICE_1.1 mDAType	M	T	F	F	T
CHOICE_2.1 nwdafAnalyticsType	M	T	F	F	T
CHOICE_3.1 ngRanInferenceType	M	T	F	F	T
CHOICE_4.1 vSExtensionType	O	T	F	F	T

7.4.10.3 Attribute constraints

Name	Definition
CHOICE_1.1 mDAType	This attribute shall be supported, when the MnS producer supports a management activity for an AIML inference for MDA
CHOICE_2.1 nwdafAnalyticsType	This attribute shall be supported, when the MnS producer supports a management activity for an AIML inference for NWDAF
CHOICE_3.1 ngRanInferenceType	This attribute shall be supported, when the MnS producer supports a management activity for an AIML inference for NG-RAN
CHOICE_4.1 vSExtensionType	This attribute shall be supported, when the MnS producer supports a management activity for an ML model with inference name is vendor's specific extensions

7.4.10.4 Notifications

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

7.4.11 DataStatisticalProperties <<dataType>>

7.4.11.1 Definition

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

7.4.11.2 Attributes

Table 7.4.11.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
uniformlyDistributedTrainingData	O	T	T	F	T
trainingDataWithOrWithoutOutliers	O	T	T	F	T

7.4.11.3 Attribute constraints

None.

7.4.11.4 Notifications

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

7.4.12 DistributedTrainingExpectation <<dataType>>

7.4.12.1 Definition

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

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

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

7.4.12.2 Attributes

Attribute name	S	isReadable	isWritable	isInvariant	isNotifiable
expectedTrainingTime	O	T	T	F	T
dataSplitIndication	O	T	T	F	T
suggestedTrainingNodeList	O	T	T	F	T

7.4.12.3 Attribute constraints

None

7.4.12.4 Notifications

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

7.4.13 PotentialImpactInfo <<dataType>>

7.4.13.1 Definition

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

7.4.13.2 Attributes

The `PotentialImpactInfo` includes the following attributes:

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
<code>impactedScope</code>	M	T	F	F	T
<code>impactedPM</code>	M	T	F	F	T

7.4.13.3 Attribute constraints

None.

7.4.13.4 Notifications

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

7.4.14 ImpactedPM <<dataType>>

7.4.14.1 Definition

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

7.4.14.2 Attributes

The `ImpactedPM` includes the following attributes:

Table 7.4.14.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
<code>pMIdentifier</code>	M	T	F	F	T

7.4.14.3 Attribute constraints

None.

7.4.14.4 Notifications

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

7.4.15 MLKnowledge <<dataType>>

7.4.15.1 Definition

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

The `MLKnowledge` is identified by a specific `aIMLInferenceName`.

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

NOTE: The `MLKnowledge` definition may be further refined or elaborated in future releases.

7.4.15.2 Attributes

Table 7.4.15.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
mLKnowledgeName	M	T	F	F	F
knowledgeType	M	T	F	F	F
predictorResponseArray	O	T	F	F	T

7.4.15.3 Attribute constraints

None

7.4.15.4 Notifications

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

7.4.16 EnvironmentScope <<choice>>

7.4.16.1 Definition

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

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

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

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

7.4.16.2 Attributes

Table 7.4.16.2-1

Attribute name	S	isReadable	isWritable	isInvariant	isNotifiable
Choice_1.1 <code>managedEntitiesScope</code>	CM	T	T	F	T
Choice_1.2 <code>areaScope</code>	CM	T	T	F	T
Choice_2 <code>timeWindow</code>	CM	T	T	F	T

7.4.16.3 Attribute constraints

Table 7.4.16.3-1

Name	Definition
Choice_1.1 <code>managedEntitiesScope</code>	Condition: the MnS producer supports to identify the scope by managed entities.
Choice_1.2 <code>areaScope</code>	Condition: the MnS producer supports to identify the scope by area scope.
Choice_2 <code>timeWindow</code>	Condition: the MnS producer supports to identify the scope by time window.

7.4.16.4 Notifications

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

7.4.17 SupportedLearningTechnology <<dataType>>

7.4.17.1 Definition

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

The SupportedLearningTechnology contains the following attributes:

`learningTechnologyName` indicates learning technologies including the name of Reinforcement learning, Federated learning and Distributed training which can be supported by the ML training function.

`supportedInferenceNameList` indicates the type of inference function that the learning technologies can be applied.

7.4.17.2 Attributes

The SupportedLearningTechnology includes the following attributes:

Table 7.4.17.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
<code>learningTechnologyName</code>	M	T	F	F	T
<code>supportedRLEnvironment</code>	CM	T	F	F	T
<code>supportedFLRole</code>	CM	T	F	F	T
<code>supportedInferenceNameList</code>	O	T	F	F	T

7.4.17.3 Attribute constraints

Table 7.4.17.3-1

Name	Definition
<code>supportedRLEnvironment</code>	Condition: This attribute shall be supported, when Reinforcement learning is supported.
<code>supportedFLRole</code>	Condition: This attribute shall be supported, when Federated learning is supported.

7.4.17.4 Notifications

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

7.4.18 RLRequirement <<dataType>>

7.4.18.1 Definition

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

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

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

`rlPerformanceRequirements` indicates the attribute of the network performance requirements for performing online ML training, which indicates the tolerable network performance degradation. When the network performance is within the range, the RL training process can be continued. Otherwise, fall back actions can be determined by the producer.

This information can be used to decide the appropriate rewards and state settings. ML training MnS consumer can use this attribute to provide performance requirements of the real operational network during RL training.

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

7.4.18.2 Attributes

The RLRequirement includes the following attributes:

Table 7.4.18.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifiable
rLEnvironmentType	O	T	T	F	T
rLEnvironmentScope	O	T	T	F	T
rLImpactedScope	O	T	T	F	T
rLPerformanceRequirements	O	T	T	F	T

7.4.18.3 Attribute constraints

None

7.4.18.4 Notifications

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

7.4.19 ClusteringCriteria <<dataType>>

7.4.19.1 Definition

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

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

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

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

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

7.4.19.2 Attributes

The `ClusteringCriteria` includes the following attributes:

Table 7.4.19.2-1

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

7.4.19.3 Attribute constraints

None.

7.4.19.4 Notifications

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

7.4.20 FLParticipationInfo <<dataType>>

7.4.20.1 Definition

This <<dataType>> defines the FL capability that an ML training function supports.

7.4.20.2 Attributes

Table 7.4.20.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
fLRole	M	T	F	F	T
isAvailableForFLTraining	O	T	F	F	T
candidateFLClientRefList	CO	T	F	F	T

NOTE: FL capability should be aligned with RL NRM

7.4.20.3 Attribute constraints

Table 7.4.20.3-1

Name	Definition
candidateFLClientRefList	Condition: The ML training function plays the role of "FL_server".

7.4.20.4 Notifications

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

7.4.21 FLRequirement <<dataType>>

7.4.21.1 Definition

The <<datatype>> FLRequirement represents the requirement for the ML training function that plays the role of FL server to train an ML model using FL.

7.4.21.2 Attributes

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

Table 7.4.21.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
FLClientSelectionCriteria	M	T	T	F	T

7.4.21.3 Attribute constraints

None

7.4.21.4 Notifications

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

7.4.22 FLClientSelectionCriteria <<dataType>>

7.4.22.1 Definition

This data type specifies the criteria for selecting the FL clients by the FL server. It defines the conditions that FL clients should meet to participate in FL.

7.4.22.2 Attributes

Table 7.4.22.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
minimumAvailableDataSamples	M	T	T	F	T
minimumAvailableTimeDuration	M	T	T	F	T
minimumInterimModelPerformance	O	T	T	F	T
servingGeoArea	O	T	T	F	T
clientRedundancy	O	T	T	F	T
trainingDataWithOrWithoutOutliers	O	T	T	F	T
uniformlyDistributedTrainingData	O	T	T	F	T

7.4.22.3 Attribute constraints

None.

7.4.22.4 Notifications

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

7.4.23 FLReportPerClient <<dataType>>

7.4.23.1 Definition

This data type specifies the detailed report for an FL from the FL server for each participating FL client.

7.4.23.2
Attributes

Table 7.4.23.2-1

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
clientRef	M	T	F	F	T
numberOfDataSamplesUsed	O	T	F	F	T
trainingTimeDuration	O	T	F	F	T
modelPerformanceOnClient	O	T	F	F	T

7.4.23.3
Attribute constraints

None.

7.4.23.4
Notifications

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

7.4a
Enumerations

7.4a.1
NgRanInferenceType <<enumeration>>

Table 7.4a.1-1: <<enumeration>> NgRanInferenceType

Enumeration value	Description
"NG_RAN_NETWORK_ENERGY_SAVING"	Indicates that the NgRanInferenceType for the Network Energy Saving defined in TS 38.300 [16]
"NG_RAN_LOAD_BALANCING"	Indicates that the NgRanInferenceType for Load Balancing defined in TS 38.300 [16]
"NG_RAN_MOBILITY_OPTIMIZATION"	Indicates that the NgRanInferenceType for the Mobility Optimization defined in TS 38.300 [16]

7.5 Attribute definitions

7.5.1 Attribute properties

Attribute Name	Documentation and Allowed Values	Properties
mLModelId	It identifies the ML model. It is unique in each MnS producer. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
candidateTrainingDataSource	It provides the address(es) of the candidate training data source provided by MnS consumer. The detailed training data format is vendor specific. allowedValues: N/A.	type: String multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
aIMLInferenceName	It indicates the type of inference that the ML model supports. allowedValues: see clause 7.4.10	type: AIMLInferenceName multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLTrainingRequest.aIMLInferenceName	It indicates the type of inference that the ML model conducting inference. allowedValues: see clause 7.4.10	type: AIMLInferenceName multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
mDAType	It indicates the type of inference that the ML model for MDA supports. The detailed definition and corresponding allowed values for mDAType see TS 28.104 [2].	type: MDAType (TS 28.104 [2]) multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
nwdafAnalyticsType	It indicates the type of inference that the ML model for NWDAF supports. The detailed definition and corresponding allowed values for nwdafAnalyticsID see NwdafEvent in TS 29.520 [20].	type: NwdafEvent (TS 29.520 [20]) multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
ngRanInferenceType	It indicates the type of inference that the ML model for NG-RAN supports. The detailed definition and corresponding allowed values for ngRanInferenceType see clause 7.4a.1	type: NgRanInferenceType multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
vSExtensionType	It indicates the type of inference that is vendor's specific extension. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
usedConsumerTrainingData	It provides the address(es) where lists of the MnS consumer-provided training data are located, which have been used for the ML model training. It may include the information about the effectiveness of training data, which indicates the MnS consumer-provided training data is useful or not. allowedValues: N/A.	type: String multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
trainingRequestRef	It is the DN(s) of the related MLTrainingRequest MOI(s).	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
trainingProcessRef	It is the DN(s) of the related MLTrainingProcess MOI(s) that produced the MLTrainingReport.	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
trainingReportRef	It is the DN of the MLTrainingReport MOI that represents the reports of the ML model training.	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
lastTrainingRef	It is the DN of the MLTrainingReport MOI that represents the reports for the last training of the ML model(s).	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
modelConfidenceIndication	It indicates the average confidence value (in unit of percentage) that the ML model would perform for inference on the data with the same distribution as training data. Essentially, this is a measure of degree of the convergence of the trained ML model. allowedValues: { 0..100 }.	type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
trainingRequestSource	It identifies the entity that requested to instantiate the MLTrainingRequest MOI. This attribute is the DN of a managed entity, otherwise, it is a String.	type: <<Choice>> multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLTrainingRequest.requestStatus	It describes the status of a particular ML model training request. allowedValues: NOT_STARTED, IN_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED.	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
mLTrainingProcessId	It identifies the training process. It is unique in each instantiated process in the MnS producer. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
priority	It indicates the priority of the training process. The priority may be used by the ML model training to schedule the training processes. Lower value indicates a higher priority. allowedValues: { 0..65535 }.	type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: 0 isNullable: False
terminationConditions	It indicates the conditions to be considered by the ML training MnS producer to terminate a specific training process. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
progressStatus	It indicates the status of the process. allowedValues: N/A.	type: ProcessMonitor multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLUpdateProcess.cancelProcesses	It allows the ML update MnS consumer to cancel the ML update process. Setting this attribute to "TRUE" cancels the ML update process. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
MLUpdateProcess.suspendProcesses	It allows the ML update MnS consumer to suspend the ML update process. Setting this attribute to "TRUE" suspends the ML update process. The process can be resumed by setting this attribute to "FALSE" when it is suspended. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
mLModelVersion	It indicates the version number of the ML model. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
performanceRequirements	It indicates the expected performance for a trained ML model when performing on the training data. allowedValues: N/A.	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
modelPerformanceTraining	It indicates the performance score of the ML model when performing on the training data. allowedValues: N/A.	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
MLTrainingProcess.progressStatus.progressStateInfo	It provides the following specialization for the "progressStateInfo" attribute of the "ProcessMonitor" data type for the "MLTrainingProcess.progressStatus". When the ML model training is in progress, and the "mLTrainingProcess.progressStatus.status" is equal to "RUNNING", it provides the more detailed progress information. allowedValues for "mLTrainingProcess.progressStatus.status" = "RUNNING": - "COLLECTING_DATA" - "PREPARING_TRAINING_DATA" - "TRAINING" + DN of the MLModel being trained The allowed values for "mLTrainingProcess.progressStatus.status" = "CANCELLING" are vendor specific. The allowed values for "mLTrainingProcess.progressStatus.status" = "NOT_STARTED" are vendor specific.	type: String multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
<code>inferenceOutputName</code>	It indicates the name of an inference output of an ML model. allowedValues: the name of the MDA output IEs (see 3GPP TS 28.104 [2]), name of analytics output IEs of NWDAF (see TS 23.288 [3]), RAN inference output IE name(s), and vendor's specific extensions.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>performanceMetric</code>	It indicates the performance metric used to evaluate the performance of an ML model, e.g. "accuracy", "precision", "F1 score", etc. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>performanceScore</code>	It indicates the performance score (in unit of percentage) of an ML model when performing inference on a specific data set (Note). The performance metrics may be different for different kinds of ML models depending on the nature of the model. For instance, for numeric prediction, the metric may be accuracy; for classification, the metric may be a combination of precision and recall, like the "F1 score". allowedValues: { 0..100 }.	type: Real multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>MLTrainingRequest.cancelRequest</code>	It allows the ML training MnS consumer to cancel the ML model training request. Setting this attribute to "TRUE" cancels the ML model training request. The request can be resumed by setting this attribute to "FALSE" when it is suspended. Cancellation is possible when the <code>requestStatus</code> is the "NOT_STARTED", "IN_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>MLTrainingRequest.suspendRequest</code>	It allows the ML training MnS consumer to suspend the ML model training request. Setting this attribute to "TRUE" suspends the ML model training process. Suspension is possible when the <code>requestStatus</code> is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>MLTrainingProcess.cancelProcess</code>	It allows the ML training MnS consumer to cancel the ML model training process. Setting this attribute to "TRUE" cancels the ML model training process. Cancellation is possible when the <code>"mLTrainingProcess.progressStatus.status"</code> is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>MLTrainingProcess.suspendProcess</code>	It allows the ML training MnS consumer to suspend the ML model training process. Setting this attribute to "TRUE" suspends the ML model training process. The process can be resumed by setting this attribute to "FALSE" when it is suspended. Suspension is possible when the <code>"mLTrainingProcess.progressStatus.status"</code> is not the "FINISHED", "CANCELLING" or "CANCELLED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
inferenceEntityRef	It describes the target entities that will use the ML model for inference.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
dataProviderRef	It describes the entities that have provided or should provide data needed by the ML model e.g. for training or inference	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
areNewTrainingDataUsed	It indicates whether new training data are used for the ML model training. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
trainingDataQualityScore	It indicates numerical value that represents the dependability/quality of a given observation and measurement type. The lowest value indicates the lowest level of dependability of the data, i.e. that the data is not usable at all. allowedValues: { 0..100 }.	type: Real multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
decisionConfidenceScore	It is the numerical value that represents the dependability/quality of a given decision generated by the AI/ML inference function. The lowest value indicates the lowest level of dependability of the decisions, i.e. that the data is not usable at all. allowedValues: { 0..100 }.	type: Real multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
expectedRuntimeContext	This describes the context where an MLModel is expected to be applied. allowedValues: N/A	type: MLContext multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
trainingContext	This specifies the context under which the MLModel has been trained. allowedValues: N/A	type: MLContext multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
runTimeContext	This specifies the context where the MLmodel or model is being applied. allowedValues: N/A	type: MLContext multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLTrainingRequest.mLModelRef	It identifies the DN of the MLModel requested to be trained.	type: DN multiplicity: 0..1 isOrdered: False isUnique: True defaultValue: None isNullable: False
MLTrainingReport.mLModelGeneratedRef	It identifies the DN of the MLModel generated by the ML model training.	type: DN multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
mLModelRepositoryRef	It identifies the DN of the mLModelRepository.	type: DN multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
mLRepositoryId	It indicates the unique ID of the ML repository.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
modelPerformanceValidation	It indicates the performance score of the ML model when performing on the validation data. allowedValues: N/A	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
dataRatioTrainingAndValidation	It indicates the ratio (in terms of quantity of data samples) of the training data and validation data used during the training and validation process. It is represented by the percentage of the validation data samples in the total training data set (including both training data samples and validation data samples). The value is an integer reflecting the rounded number of percent * 100. allowedValues: { 0 .. 100 }.	type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLTestingRequest.requestStatus	It describes the status of a particular ML testing request. allowedValues: NOT_STARTED, IN_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED.	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLTestingRequest.cancelRequest	It allows the ML testing MnS consumer to cancel the ML testing request. Setting this attribute to "TRUE" cancels the ML testing request. Cancellation is possible when the requestStatus is the "NOT_STARTED", "IN_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
MLTestingRequest.suspendRequest	It allows the ML testing MnS consumer to suspend the ML testing request. Setting this attribute to "TRUE" suspends the ML testing request. The request can be resumed by setting this attribute to "FALSE" when it is suspended. Suspension is possible when the requestStatus is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
MLTestingRequest.mLModelRef	It identifies the DN of the mLModel requested to be tested.	type: DN Multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
modelPerformanceTesting	It indicates the performance score of the ML model when performing on the testing data. allowedValues: N/A.	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
mLTestingResult	It provides the address where the testing result is provided. The detailed testing result format is vendor specific. allowedValues: N/A.	type: String multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
testingRequestRef	It identifies the DN of the mLTestingRequest MOI.	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
supportedPerformanceIndicators	This parameter lists specific PerformanceIndicator(s) of an ML model. allowedValues: N/A.	type: SupportedPerfIndicator multiplicity: 1..* isOrdered: False isUnique: True defaultValue: None isNullable: False
performanceIndicatorName	It indicates the identifier of the specific performance indicator. allowedValues: N/A	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
isSupportedForTraining	It indicates whether the specific performance indicator is supported a performance metric of ML model training for the ML model. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
isSupportedForTesting	It indicates whether the specific performance indicator is supported a performance metric of ML model testing for the ML model. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
mLUpdateProcessRef	It is the DN of the mLUpdateProcess MOI that represents the process of updating an ML model.	type: DN multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
mLUpdateRequestRefList	It is the list of DN of the mLUpdateRequest MOI that represents an ML update request.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
mLUpdateReportRef	It is the DN of the mLUpdateReport MOI that represents an ML update report.	type: DN multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
mLUpdateReportingPeriod	It specifies the time duration upon which the MnS consumer expects the ML update is reported.	type: TimeWindow multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
availMLCapabilityReport	It represents the available ML capabilities. allowedValues: N/A.	type: AvailMLCapabilityReport multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
updatedMLCapability	It represents the updated ML capabilities. allowedValues: N/A.	type: AvailMLCapabilityReport multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
availMLCapabilityReportID	It identifies the available ML capability report. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
newCapabilityVersionId	It indicates the specific version of AI/ML capabilities to be applied for the update. It is typically the one indicated by the MLCapabilityVersionID in a newCapabilityVersion	type: String multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
mlCapabilityVersionId	It indicates the version of ML capabilities that is available for the update.	type: String multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
performanceGainThreshold	It defines the minimum performance gain as a percentage that shall be achieved with the capability update, i.e., the difference in the performances between the existing capabilities and the new capabilities should be at least performanceGainThreshold otherwise the new capabilities should not be applied. Allowed value: float between 0.0 and 100.0	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
expectedPerformanceGains	It indicates the expected performance gain if/when the AI/ML capabilities of the respective network function are updated with/to the specific set of newly available AI/ML capabilities.	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
updateTimeDeadline	It indicates the maximum as stated in the MLUpdate request that should be taken to complete the update	type: TimeWindow multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
MLUpdateReport.mLModelRefList	It indicates the DN of MLModel instances that can be updated.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
MLUpdateRequest.requestStatus	It describes the status of a particular ML update request. allowedValues: NOT_STARTED, IN_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED.	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
<code>MLUpdateRequest.cancelRequest</code>	It allows the MnS consumer to cancel the ML update request. Setting this attribute to "TRUE" cancels the ML update request. Cancellation is possible when the <code>requestStatus</code> is the "NOT_STARTED", "IN_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>MLUpdateRequest.suspendRequest</code>	It allows the MnS consumer to suspend the ML update request. Setting this attribute to "TRUE" suspends the ML update request. The request can be resumed by setting this attribute to "FALSE" when it is suspended. Suspension is possible when the <code>requestStatus</code> is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>memberMLModelRefList</code>	It identifies the list of member ML models within an ML model coordination group.	type: DN multiplicity: 2..* isOrdered: True isUnique: True defaultValue: None isNullable: False
<code>MLTrainingRequest.mLModelCoordinationGroupRef</code>	It identifies the DN of the <code>MLModelCoordinationGroup</code> requested to be trained.	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>MLTrainingReport.mLModelCoordinationGroupGeneratedRef</code>	It identifies the DN of the <code>MLModelCoordinationGroup</code> generated by ML model joint training.	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>MLTestingRequest.mLModelCoordinationGroupRef</code>	It identifies the DN of the <code>MLModelCoordinationGroup</code> requested to be tested.	type: DN multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>retrainingEventsMonitorRef</code>	It indicates the DN of the <code>ThresholdMonitor</code> MOI that indicates the performance measurements and its corresponding thresholds to be used by MnS producer to initiate the re-training of the <code>MLModel</code> .	type: DN multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>MLModelLoadingRequest.requestStatus</code>	It describes the status of a particular ML model loading request. allowedValues: NOT_STARTED, IN_PROGRESS, CANCELLING, SUSPENDED, FINISHED, and CANCELLED.	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>MLModelLoadingRequest.cancelRequest</code>	It allows the MnS consumer to cancel the ML model loading request. Setting this attribute to "TRUE" cancels the ML model loading. Cancellation is possible when the <code>requestStatus</code> is the "NOT_STARTED", "IN_PROGRESS", and "SUSPENDED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
<code>MLModelLoadingRequest.suspendedRequest</code>	It allows the MnS consumer to suspend the ML model loading request. Setting this attribute to "TRUE" suspends the ML model loading request. The request can be resumed by setting this attribute to "FALSE" when it is suspended. Suspension is possible when the <code>requestStatus</code> is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>mLModelToLoadRef</code>	It identifies the DN of a trained <code>MLModel</code> requested to be loaded to the target inference function(s).	type: DN multiplicity: 0..* isOrdered: False isUnique: True defaultValue: None isNullable: False
<code>policyForLoading</code>	It provides the policy for controlling ML model loading triggered by the MnS producer. This policy contains two thresholds in the <code>thresholdList</code> attribute. The first threshold is related to the ML model to be loaded, and the second threshold is related to the existing ML model being used for inference.	type: AIMLManagementPolicy multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>thresholdList</code>	It provides the list of threshold.	type: ThresholdInfo multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
<code>MLModelLoadingProcess.progressStatus.progressStateInfo</code>	It provides the following specialization for the "progressStateInfo" attribute of the "ProcessMonitor" data type for the " <code>MLModelLoadingProcess.progressStatus</code> ". When the ML model loading is in progress, and the " <code>MLModelLoadingProcess.progressStatus.status</code> " is equal to "RUNNING", it provides the more detailed progress information. allowedValues for " <code>MLModelLoadingProcess.progressStatus.status</code> " = "RUNNING": The allowed values for " <code>MLModelLoadingProcess.progressStatus.status</code> " = "CANCELLING" are vendor specific. The allowed values for " <code>MLModelLoadingProcess.progressStatus.status</code> " = "NOT_STARTED" are vendor specific.	type: String multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>MLModelLoadingProcess.cancelProcess</code>	It allows the MnS consumer to cancel the ML model loading process. Setting this attribute to "TRUE" cancels the process. Cancellation is possible when the " <code>MLModelLoadingProcess.progressStatus.status</code> " is not the "FINISHED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
<code>MLModelLoadingProcess.suspendedProcess</code>	It allows the MnS consumer to suspend the ML model loading process. Setting this attribute to "TRUE" suspends the process. The process can be resumed by setting this attribute to "FALSE" when it is suspended. Suspension is possible when the "MLModelLoadingProcess.progressStatus.status" is not the "FINISHED", "CANCELLING" or "CANCELLED" state. Setting the attribute to "FALSE" has no observable result. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
<code>mLModelLoadingRequestRef</code>	It identifies the DN of the associated <code>MLModelLoadingRequest</code> .	type: DN multiplicity: 0..* isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>mLModelLoadingPolicyRef</code>	It identifies the DN of the associated <code>MLModelLoadingPolicyRef</code> .	type: DN multiplicity: 0..* isOrdered: False isUnique: True defaultValue: None isNullable: False
<code>loadedMLModelRef</code>	It identifies the DN of the <code>MLModel</code> that has been loaded to the inference function.	type: DN multiplicity: 0..* isOrdered: False isUnique: True defaultValue: None isNullable: False
<code>activationStatus</code>	It describes the activation status. allowedValues: ACTIVATED, DEACTIVATED.	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>AIMLManagementPolicy.managedActivationScope</code>	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: <code>ManagedActivationScope</code> multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>AIMLInferenceFunction.managedActivationScope</code>	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: <code>AIMLManagementPolicy</code> multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
<code>ManagedActivationScope.dnList</code>	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

Attribute Name	Documentation and Allowed Values	Properties
ManagedActivationScope.timeWindow	It indicates the list of time window; the list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope. allowedValues: N/A	type: TimeWindow multiplicity: * isOrdered: True isUnique: True defaultValue: None isNullable: False
ManagedActivationScope.geoPolygon	It indicates the list of GeoArea, the list is an ordered list indicating the inference is activated for the first sub scope and gradually extended to the next sub scope. allowedValues: N/A	type: GeoArea multiplicity: * isOrdered: True isUnique: True defaultValue: None isNullable: False
usedByFunctionRefList	It provides the DNs of the functions supported by the AIMLINferenceFunction. allowedValues: N/A	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
inferenceOutputId	It identifies an inference output within an AIMLINferenceReport.	type: String multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
inferenceOutputs	It indicates the Outputs that have been derived by the AIMLINferenceFunction instance from a specific ML model. Each ML model, inferenceOutputs may be a set of values. allowedValues: N/A.	type: InferenceOutput multiplicity: 1..* isOrdered: False isUnique: True defaultValue: None isNullable: False
inferencePerformance	It indicates the performance score of the ML model during Inference. allowedValues: N/A.	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
inferenceOutputTime	It indicates the time at which the inference output is generated. allowedValues: N/A	type: DateTime multiplicity: * isOrdered: True isUnique: True defaultValue: None isNullable: False
outputResult	It indicates the result of an inference.	type: AttributeValuePair multiplicity: * isOrdered: False isUnique: True defaultValue: Null isNullable: False
mLCapabilitiesInfoList	It indicates information about what an ML model can generate inference for. allowedValues: N/A.	type: MLCapabilityInfo multiplicity: 1..* isOrdered: False isUnique: True defaultValue: None isNullable: False
capabilityName	It indicates the name of a capability for which an ML model can generate inference. The capability is defined by Mns producer which can be traffic analysis capability, coverage analysis capability, mobility analysis capability or vendor specific extensions. allowedValues: N/A.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
mLCapabilityParameters	It indicates a set of optional parameters that apply for an aIMLInferenceName capabilityName. allowedValues: N/A	type: AttributeValuePair multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
aIMLInferenceReportRefList	It indicates a list of DN of the aIMLInferenceReport MOI that represents an AIML inference report.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
mLModelRefList	It identifies the list of MLModel DN.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
mLKnowledge	It indicates an instance of ML Knowledge available at the ML training function.	type: MLKnowledge multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
mLKnowledgeName	It identifies the ML Knowledge. It is unique in each MnS producer.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
knowledgeType	It identifies the type of ML Knowledge as either a Statistic, a regression or a Table of input-output value(s) Allowed values: TABLE , STATISTIC, REGRESSION	type: ENUM multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
predictorResponseArray	It identifies the predictor and corresponding response data for a piece of ML Knowledge. For exapme, it represents one of the following: - the input and output data for a table - the predictor and response for a statistic, - the input and output data for a regression NOTE: The nature of the data is not scope of this specification	type: pair<String, String> multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
inferenceExplanationInfo	It indicates the inference explanation information of the ML model Inference results. E.g. the critical features in the training or inference data.	type: String multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
mLTrainingType	It indicates the type of ML training (e.g., initial-training, re-training, pre-specialised training, fine-tuning) requested by the MnS consumer. allowed values: INITIAL_TRAINING, PRE_SPECIALISED_TRAINING, RE_TRAINING, FINE_TUNING	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
expectedInferenceScope	It indicates the inference capabilities that the ML model is expected to support, where the inference scope contains a list of aIMLInferenceName that the ML model can be potential adapted to support.	type: AIMLInferenceName multiplicity: * isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
inferenceScope	It indicates the inference capabilities that the ML model after pre-specialized training can be fine-tuned to support, where the inference scope contains a list of aIMLInferenceName that the ML model can be potentially adapted to support.	type: AIMLInferenceName multiplicity: * isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
distributedTrainingExpectation	It indicates distributed training expectations provided by MnS consumer. allowedValues: N/A.	type: DistributedTrainingExpectation multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
expectedTrainingTime	It indicates the expected training duration provided by MnS consumer, in unit of minutes. allowedValues: Integer	type: Integer multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
dataSplitIndication	This is a Boolean attribute specifying whether the provided training data should be split or not. The value FALSE specifies that the training data shall not be split. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: False isNullable: False
suggestedTrainingNodeList	It indicates a list of suggested training nodes provided by MnS consumer. allowedValues: Not applicable.	type: DN multiplicity: 0..* isOrdered: N/A isUnique: True defaultValue: None isNullable: False
trainingDataStatisticalProperties	It indicates the training data statistical properties to be considered by the MnS producer when training an ML model.	type: DataStatisticalProperties multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
uniformlyDistributedTrainingData	It indicates the need for using training data that are uniformly distributed according to the different aspects (e.g., equivalent data samples for each UE in the training data, equivalent data samples for each type of slice in the training data, equivalent data samples from each GeoArea in the training data) of the aIMLInferenceName. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
trainingDataWithOrWithoutOutliers	It indicates that the training data samples should consider or disregard data samples that are at the extreme boundaries of the value range. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
potentialImpactInfo	This datatype defines the potential network impacts due to the inference output results	type: PotentialImpactInfo multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
impactedScope	<p>This will specify the scope of affect, the inference output may have on the network including entities performing the recommended actions in the inference output and entities impacted due to implementation of the recommended actions</p> <p>The choice attribute <code>dNList</code> defines Identifier of the network functions that may be affected by the output result of the inference function.</p> <p>The choice attribute <code>timeWindow</code> defines a time duration indicating that the related network function(s) may be affected during this time duration by the inference output result.</p> <p>The choice attribute <code>geoPolygon</code> defines a Geographical location indicating that the network function(s) in that location may be affected by the inference output result.</p>	<p>type: ManagedActivationScope multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False</p>
impactedPM	This will identify the potential performance metrics that may be degraded/improved due to the implementation of recommendations provided as part of inference output.	<p>type: ImpactedPM multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False</p>
pMIdentifier	This indicates the performance measurement or the KPI that may be impacted by the ML model. This will be the name of PM and KPI as defined in 3GPP TS 28.552 and 28.554 respectively (e.g. for Managing NG-RAN AI/ML-based distributed Load Balancing function, the PM can be measurements related to MLB, UE throughput and Radio resource utilization etc).	<p>type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False</p>
supportedLearningTechnology	It identifies the learning technologies including Reinforcement learning, Federated learning and Distributed training which supported by the ML training function.	<p>type: SupportedLearningTechnology multiplicity: 1 isOrdered: False isUnique: True defaultValue: None isNullable: False</p>
rLRequirement	It identifies the expected performance and performed scope for the ML model training when Reinforcement learning is supported.	<p>type: RLRequirement multiplicity: 1 isOrdered: False isUnique: True defaultValue: None isNullable: False</p>
learningTechnologyName	<p>It indicates a list of learning technology names used to represent the learning technics supported by the ML training function.</p> <p>allowedValues: RL, FL, DL where RL indicates Reinforcement learning, FL indicates Federated learning and DL indicates of Distributed training.</p>	<p>type: Enum multiplicity: 1..* isOrdered: False isUnique: True defaultValue: None isNullable: False</p>
supportedRLEnvironment	<p>It indicates the supported RL environments. When the ML training function supports RL, this attribute is included in the <code>SupportedLearningTechnology</code> datatype, which indicates the supported environment of the ML training function for ML model training by RL.</p> <p>allowedValues: SIMULATION_ENVIRONMENTS, REAL_NETWORK_ENVIRONMENTS.</p>	<p>type: Enum multiplicity: 1..* isOrdered: False isUnique: True defaultValue: None isNullable: False</p>

Attribute Name	Documentation and Allowed Values	Properties
supportedFLRole	It indicates the role that the ML training function supports to play in the FL. allowedValues: FL_SERVER, FL_CLIENT,	type: Enum multiplicity: 1..2 isOrdered: False isUnique: True defaultValue: None isNullable: False
FLParticipationInfo	It indicates the information of the ML training function participating in the FL process.	type: FLParticipationInfo multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLParticipationInfo.fLRole	It indicates the role that an ML training function plays in FL. allowedValues: FL_Server, FL_Client.	type: Enum multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLParticipationInfo.isAvailableForFLTraining	This attribute defines the FL state of the MLTrainingFunction for training a given ML model. allowedValues: "TRUE" indicates that the MLTrainingFunction is available for starting or joining a new FL process for the ML model; "FALSE" indicates that the MLTrainingFunction is unavailable for a starting or joining new FL process for the ML model.	type: Boolean multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: False isNullable: False
FLParticipationInfo.candidateFLClientRefList	It identifies the DNs of the MLTrainingFunction instances that are capable of acting as the FL client.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
FLRequirement	It indicates the requirements of FL training.	type: FLRequirement multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLClientSelectionCriteria	It provides the criteria for selecting the FL clients for an FL.	type: FLClientSelectionCriteria multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
FLClientSelectionCriteria.minimumAvailableDataSamples	It indicates the minimum number of data samples can be used for training.	type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLClientSelectionCriteria.minimumAvailableTimeDuration	It indicates the minimum time length that the FL client is available to participate into an FL, in unit of minutes.	type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLClientSelectionCriteria.minimumInterimModelPerformance	It indicates the minimum training performance score for an interim model on an FL client.	type: ModelPerformance multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
FLClientSelectionCriteria.uniformlyDistributedTrainingData	It indicates the need for using training data that are uniformly distributed according to the different aspects (e.g., equivalent data samples for each UE in the training data, equivalent data samples for each type of slice in the training data, equivalent data samples from each GeoArea in the training data) of the aMLInferenceName. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
FLClientSelectionCriteria.trainingDataWithOrWithoutOutliers	It indicates that the training data samples should consider or disregard data samples that are at the extreme boundaries of the value range. allowedValues: TRUE, FALSE.	type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False
FLClientSelectionCriteria.servingGeoArea	It indicates the serving geographical area of an FL client.	type: GeoArea multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
FLClientSelectionCriteria.clientRedundancy	This defines that if the FL client needs to have some type of redundancy to handle client dropouts gracefully in order to be selected by FL server to train a ML model. Its values can be TRUE or FALSE, where TRUE means the FL client must have some type of redundancy and FALSE means the redundancy of FL client does not matter for its selection	type: Boolean multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLReportPerClient	This report is provided by the server to the consumer.	type: FLReportPerClient multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
FLReportPerClient.clientRef	It identifies the DN of MLTrainingFunction that plays the role of FL client.	type: DN multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLReportPerClient.numberOfDataSamplesUsed	It indicates the number of data samples that have been used in the ML training.	type: Integer multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
FLReportPerClient.trainingTimeDuration	It indicates the time window that the FL client/FL server can participate into an FL process.	type: TimeWindow multiplicity: 1..* isOrdered: False isUnique: True defaultValue: None isNullable: False
FLReportPerClient.modelPerformanceOnClient	It indicates the performance score of the final global ML model running on the local data set of the FL client.	type: ModelPerformance multiplicity: 2..* isOrdered: False isUnique: False defaultValue: None isNullable: False
participatingFLClientRefList	It identifies the list of DN of the MLTrainingFunction that participated the FL process as FL clients.	type: DN multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False

Attribute Name	Documentation and Allowed Values	Properties
supportedInferenceNameList	It indicates a list of inference name that the learning technologies can be applied. allowedValues: see clause 7.4.10	type: AIMLInferenceName multiplicity: 1..* isOrdered: False isUnique: N/A defaultValue: None isNullable: False
rLEnvironmentType	It indicates the simulated environment or real network where the ML model should be trained. allowedValues: SIMULATION_ENVIRONMENTS, REAL_NETWORK_ENVIRONMENTS	type: Enum multiplicity: 0..* isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
rLEnvironmentScope	It indicates the specific environment scope for the entities that the RL process should be performed, i.e, where the RL agent is located.	type: EnvironmentScope multiplicity: 1..* isOrdered: False isUnique: N/A defaultValue: None isNullable: False
rLImpactedScope	It indicates the specific environment scope for the entities that may be impacted by the RL process, i.e., scope may be impacted by actions of the RL agent.	type: EnvironmentScope multiplicity: 1..* isOrdered: False isUnique: N/A defaultValue: None isNullable: False
rLPerformanceRequirements	It indicates a list of thresholds for the network performance requirements, when the RL training process(es) is performed.	type: ThresholdInfo multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
clusteringInfo	It contains information that indicates the clustering criteria for the ML models that can be grouped together for training	type: ClusteringCriteria multiplicity: * isOrdered: False isUnique: True defaultValue: None isNullable: False
ClusteringCriteria.performanceMetric	This defines clustering criteria based on the performance metric for which the ML model is mainly evaluated. That is, the models, which intend to achieve same performance characteristic (e.g. accuracy, precision, F1 score etc) can be clustered together for training. It indicates the performance metric used to evaluate the performance of an ML model allowedValues: N/A	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
taskType	This defines grouping criteria based on the task the ML model is trained for. For example, this can be aIMLInferenceName or capabilityName as defined in 3GPP TS 28.105. Note: Whether the taskType can be aIMLInferenceName here is FFS.	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
allowedClusterTrainingTime	This defines the combined time limit within which the training of ML models cluster shall be completed. A cluster of ML models takes more time to train together as compared to time taken for training an individual ML model. The criteria allows accommodating only those ML models whose training time does not exceed the set combined time limit	type: TimeWindow multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: True

Attribute Name	Documentation and Allowed Values	Properties
preferredModelDiversity	This defines the MnS consumer preferred model diversity types that is to be considered for models clustering. For example, decision trees, neural networks, linear regression and like so	type: String multiplicity: 1 isOrdered: N/A isUnique: N/A defaultValue: None isNullable: False
NOTE: When the performanceScore is to indicate the performance score for ML model training, the data set is the training data set. When the performanceScore is to indicate the performance score for ML validation, the data set is the validation data set. When the performanceScore is to indicate the performance score for ML model testing, the data set is the testing data set.		

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	--
notifyMOIChanges	O	--

8 Service components

8.0 General

The operations for generic provisioning management service refer to clause 11.1.1 of TS 28.532 [11]. For notifications, see clause 7.6.

8.1 Lifecycle management operations for AI/ML management MnS

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

Table 8.1-1: Components for ML model training

ML model training management capability	Management service component type A	Management service component type B	Management service component type C
Create an ML model training request	createMOI operation	MLTrainingRequest	N/A
Modify an ML model training request	modifyMOIAttributes operation		
Query an ML model training report	getMOIAttributes operation	MLTrainingReport	
Modify an ML model training process	modifyMOIAttributes operation	MLTrainingProcess	
Query an ML model training process	getMOIAttributes operation		
Create, Delete, and Update ML model training	changeMOIs operation	MLTrainingRequest	

The components for ML model testing are listed in table 8.1-2.

Table 8.1-2: Components for ML model testing

ML model testing management capability	Management service component type A	Management service component type B	Management service component type C
Create an ML model testing request	createMOI operation	MLTestingRequest	N/A
Modify an ML model testing request	modifyMOIAttributes operation		
Query an ML model testing report	getMOIAttributes operation	MLTestingReport	
Subscribe an ML model testing report	createMOI operation		
Unsubscribe an ML model testing report	deleteMOI operation		
Query an ML model testing report subscription	getMOIAttributes operation		
Create, Delete, and Update ML model testing	changeMOIs operation	MLTestingRequest	

The components for ML model deployment are listed in table 8.1-3

Table 8.1-3: Components for ML model deployment

ML model deployment management capability	Management service component type A	Management service component type B	Management service component type C
Create an ML model loading request	createMOI operation	MLModelLoadingRequest	N/A
Modify an ML model loading request	modifyMOIAttributes operation		
Create an ML model loading policy	createMOI operation	MLModelLoadingPolicy	
Delete an ML model loading policy	deleteMOI operation		
Modify an ML model loading policy	modifyMOIAttributes operation		
Query an ML model loading policy	getMOIAttributes operation		
Modify an ML model loading process	modifyMOIAttributes operation	MLModelLoadingProcess	
Query an ML model loading process	getMOIAttributes operation		
Create, Delete, and Update ML model Loading	changeMOIs operation	MLModelLoadingRequest MLModelLoadingPolicy	

The components for ML model inference are listed in table 8.1-4

Table 8.1-4: Components for AI/ML inference

AI/ML Inference management capability	Management service component type A	Management service component type B	Management service component type C
Create an ML model update request	createMOI operation	MLUpdateRequest	N/A
Modify an ML model update request	modifyMOIAttributes operation		
Modify an ML model update process	modifyMOIAttributes operation	MLUpdateProcess	
Query an ML model update process	getMOIAttributes operation		
Query an ML model update report	getMOIAttributes operation	MLUpdateReport	
Query an AI/ML inference report	getMOIAttributes operation	AIMLInferenceReport	
Create, Delete, and Update ML model Update and ML update request	changeMOIs operation	MLUpdateRequest	

The components for AI/ML inference emulation are listed in table 8.1-5.

Table 8.1-5: Components for AI/ML inference emulation MnS

ML model emulation management capability	Management service component type A	Management service component type B	Management service component type C
Query an AI/ML inference emulation report	getMOIAttributes operation	AIMLInferenceReport	N/A

9 Solution Set (SS)

9.1 OpenAPI document for provisioning MnS

The OpenAPI/YAML definitions for provisioning MnS are specified in 3GPP Forge, refer to clause 4.3 (OpenAPI Definitions) of TS 28.623 [19] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag_Rel19_SA105/".

Directory: OpenAPI

File: TS28532_ProvMnS.yaml

9.2 OpenAPI document for AI/ML management

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

The OpenAPI/YAML definitions are specified in 3GPP Forge, refer to clause 4.3 of TS 28.623 [19] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag_Rel19_SA105/".

Directory: OpenAPI

File: TS28105_AiMINrm.yaml

10 Stage 3 definition for AI/ML Management

10.1 RESTful HTTP-based solution set

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

10.1.1 ML model training

Table 10.1.1-1 describes the SS to support ML model training request management based on Table 12.1.1.1-1 in TS 28.532 [11].

Table 10.1.1-1: SS to support ML model training request management

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

Table 10.1.1-2 describes the SS to support ML model training report management based on Table 12.1.1.1-1 in TS 28.532 [11].

Table 10.1.1-2: SS to support ML model training report management

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

Table 10.1.1-3 describes the SS to support ML model training process based on Table 12.1.1.1-1 in TS 28.532 [11].

Table 10.1.1-3: SS to support ML model training process management

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

10.1.2 ML model testing

Table 10.1.2-1 describes the SS to support ML model testing management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.2-1: SS to support ML model testing management

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

Table 10.1.2-2 describes the SS to support ML model testing report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.2-2: SS to support ML model testing report management

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

10.1.3 AI/ML inference emulation

Table 10.1.3-1 describes the SS to support AI/ML inference emulation report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.3-1: SS to support AI/ML inference emulation report management

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

10.1.4 ML model deployment

Table 10.1.4-1 describes the SS to support ML model loading request management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.4-1: SS to support ML model loading request management

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

Table 10.1.4-2 describes the SS to support ML model loading policy based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.4-2: SS to support ML model loading policy management

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

Table 10.1.4-3 describes the SS to support ML model loading process management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.4-3: SS to support ML model loading process management

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

10.1.5 AI/ML inference

Table 10.1.5-1 describes the SS to support ML model update request management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.5-1: SS to support ML model update request management

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

Table 10.1.5-2 describes the SS to support ML model update report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.5-2: SS to support ML model update report management

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

Table 10.1.5-3 describes the SS to support ML model update process management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.5-3: SS to support ML model update process management

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

Table 10.1.5-4 describes the SS to support AI/ML inference report management based on Table 12.1.1.1.1-1 in TS 28.532 [11].

Table 10.1.5-4: SS to support AI/ML inference report management

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

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 MLModel <<InformationObjectClass>>
class MLModelCoordinationGroup <<InformationObjectClass>>
class MLTrainingFunction <<InformationObjectClass>>
class MLTrainingRequest <<InformationObjectClass>>
class MLTrainingReport <<InformationObjectClass>>
class MLTrainingProcess <<InformationObjectClass>>
class MLModelRepository <<InformationObjectClass>>

ManagedEntity "1" *-- "*" MLTrainingFunction: <<names>>
MLTrainingFunction "1" *-- "*" MLTrainingProcess: <<names>>
MLTrainingFunction "1" *-- "*" MLTrainingRequest: <<names>>
MLTrainingFunction "1" *-- "*" MLTrainingReport: <<names>>
MLTrainingFunction "1" *-- "*" ThresholdMonitor : <<names>>

MLTrainingFunction "*" --> "1" MLModelRepository
MLTrainingProcess "1" -r-> "1" MLTrainingReport
MLTrainingProcess "*" <-l- "*" MLTrainingRequest
MLTrainingRequest "1" --> "0..1" MLModel
MLTrainingRequest "1" -r-> "0..1" MLModelCoordinationGroup
MLTrainingReport "1" --> "0..1" MLModel
MLTrainingReport "1" --> "0..1" MLModelCoordinationGroup
MLTrainingProcess "1" --> "0..1" MLModel
MLTrainingProcess "1" --> "0..1" MLModelCoordinationGroup
MLModel "*" -l-> "1" ThresholdMonitor
MLTrainingReport "1" -r-> "1" MLTrainingReport

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 MLModelRepository <<InformationObjectClass>>
class MLModel <<InformationObjectClass>>
class MLModelCoordinationGroup <<InformationObjectClass>>

Top <|-- MLModelRepository
Top <|-- MLModel
Top <|-- MLModelCoordinationGroup

@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 ManagedEntity <<ProxyClass>>
class MLModelRepository <<InformationObjectClass>>
class MLModel <<InformationObjectClass>>
class MLModelCoordinationGroup <<InformationObjectClass>>

ManagedEntity "1" *-- "*" MLModelRepository : <<names>>
MLModelRepository "1" *-- "*" MLModel : <<names>>
MLModelRepository "1" *-- "*" MLModelCoordinationGroup : <<names>>

MLModelCoordinationGroup "*" -r-> "2..*" MLModel

note left of ManagedEntity
    This represents the following IOCs:
    ManagedElement or
    SubNetwork
end note

@enduml
```

A.6 PlantUML code for Figure 7.3a.1.1.1-2: NRM fragment for ML model 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 MLModel <<InformationObjectClass>>
class MLModelCoordinationGroup <<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" MLModel
MLTestingRequest "*" --> "0..1" MLModelCoordinationGroup
MLTestingReport "*" -l-> "1" MLTestingRequest

(MLTestingRequest, MLModel) ... (MLTestingRequest, MLModelCoordinationGroup) : {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 model 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 MLModel <<InformationObjectClass>>

MLUpdateEntity "1" *-- "*" MLUpdateFunction:<<names>>
MLUpdateFunction "1" *-- "*" MLUpdateRequest:<<names>>
MLUpdateFunction "1" *-- "*" MLUpdateProcess:<<names>>
MLUpdateFunction "1" *-- "*" MLUpdateReport:<<names>>

MLUpdateFunction "1" --> "*" "MLModel"
MLUpdateRequest "*" <-r-> "1" "MLUpdateProcess"
MLUpdateProcess "1" <-r-> "1" "MLUpdateReport"
MLUpdateProcess "*" --> "*" "MLModel"
MLUpdateReport "*" --> "*" "MLModel"
MLUpdateRequest "*" --> "*" "MLModel"

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 model 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 MLModel <<InformationObjectClass>>
class MLModelLoadingRequest <<InformationObjectClass>>
class MLModelLoadingPolicy <<InformationObjectClass>>
class MLModelLoadingProcess <<InformationObjectClass>>

AIMLINferenceFunction "1" *-- "*" MLModelLoadingRequest : <<names>>
AIMLINferenceFunction "1" *-- "*" MLModelLoadingPolicy : <<names>>
AIMLINferenceFunction "1" *-- "*" MLModelLoadingProcess : <<names>>

MLModelLoadingRequest "1" <-r- "*" MLModelLoadingProcess
MLModelLoadingProcess "*" -r-> "1" MLModelLoadingPolicy

MLModelLoadingRequest "1" --> "*" MLModel
MLModelLoadingProcess "1" --> "*" MLModel

(MLModelLoadingProcess, MLModelLoadingRequest) ... (MLModelLoadingProcess, MLModelLoadingPolicy) :
{xor}

@enduml

```

A.11 PlantUML code for Figure 7.3a.3.1.2-1: Inheritance Hierarchy for ML model 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 MLModelLoadingRequest <<InformationObjectClass>>
class MLModelLoadingPolicy <<InformationObjectClass>>
class MLModelLoadingProcess <<InformationObjectClass>>

Top <|-- MLModelLoadingRequest
Top <|-- MLModelLoadingPolicy
Top <|-- MLModelLoadingProcess

@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 MLModel <<InformationObjectClass>>

```

```

class ManagedEntity <<ProxyClass>>
class AIMLSupportedFunction <<ProxyClass>>

ManagedEntity "1" *-- "*" AIMLINferenceFunction : <<names>>
AIMLINferenceFunction "*" <-l-> "*" AIMLSupportedFunction
MLModel "*" <-r-> "*" AIMLSupportedFunction
MLModel "*" <-r-> "*" AIMLINferenceFunction
MLModel "1..*" <-r-> "*" AIMLINferenceReport
AIMLINferenceFunction "1" *-- "*" AIMLINferenceReport : <<names>>

note right of ManagedEntity #white
  Represents the IOCs:
    ManagedElement or
    SubNetwork or
    ManagedFunction
end note

note top of AIMLSupportedFunction #white
  Represents the IOCs:
    DMROFunction or
    DLBOFunction or
    DESManagementFunction or
    MDAFunction or
    AnLFFunction or
    LMFFunction
end note

@enduml

```

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 AIMLINferenceReport << InformationObjectClass >>

ManagedEntity "1" *-- "*" AIMLINferenceEmulationFunction: <<names>>
AIMLINferenceEmulationFunction "1" *-- "*" AIMLINferenceReport : <<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 ManagedFunction <<InformationObjectClass>>
class AIMLINferenceEmulationFunction << InformationObjectClass >>

ManagedFunction <|-- AIMLINferenceEmulationFunction
@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"

Note that clause 9 includes the location of TS28105_AiMlNrm.yaml.

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 (1 st 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 (1 st 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
2024-06	SA#104	SP-240830	0126	-	F	TS28.105 Rel18 correction to Schema definition Issues for SubNetwork of OpenAPI SS	18.4.0
2024-06	SA#104	SP-240808	0140	-	F	TS28.105 Rel18 Moving normative stage 3 to Forge	18.4.0
2024-06	SA#104	SP-240804	0142	1	A	Rel-18 CR TS 28.105 correct the AI/ML technique overview	18.4.0
2024-06	SA#104	SP-240830	0146	-	F	Rel 18 Input to draftCR TS 28.105 correct terminationConditions attribute	18.4.0
2024-06	SA#104	SP-240830	0147	-	F	Rel18 TS 28.105 stage 3 changes for stage 2 corrections	18.4.0
2024-06	SA#104	SP-240830	0151	-	F	CR Rel-18 TS28.105 AI/ML management	18.4.0
2024-09	SA#105	SP-241181	0152	1	F	TS28.105 Rel18 correction to attribute name of attribute related to role for different cardinality	18.5.0
2024-09	SA#105	SP-241181	0153	1	F	Rel18 TS 28.105 corrections to misalignment between stage 3 and stage 2	18.5.0
2024-09	SA#105	SP-241181	0154	1	F	Rel-18 CR TS 28.105 correct the ML model related attributes	18.5.0
2024-09	SA#105	SP-241181	0155	1	F	Rel-18 CR TS 28.105 correct the description of ML model lifecycle and ML entity	18.5.0
2024-09	SA#105	SP-241181	0156	-	F	Rel-18 CR TS 28.105 Correct InferenceType to aiMLInferenceName	18.5.0

Change history							
Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
2024-09	SA#105	SP-241181	0157	1	F	Rel-18 CR TS 28.105 Correct MLEntity to MLModel	18.5.0
2024-09	SA#105	SP-241181	0160	2	F	Rel-18 CR TS 28.105 Clarification of scope	18.5.0
2024-09	SA#105	SP-241181	0161	1	F	Rel-18 CR TS28.105 corrections to ML model lifecycle figure and corresponding description	18.5.0
2024-09	SA#105	SP-241181	0163	-	F	Rel 18 CR TS 28.105 Fix training definitions	18.5.0
2024-09	SA#105	SP-241181	0164	-	F	Rel 18 CR TS 28.105 Fix remaining entity to model	18.5.0
2024-09	SA#105	SP-241181	0168	1	A	Rel-18 CR TS 28.105 Correction to using data types	18.5.0
2024-09	SA#105	SP-241181	0176	1	F	Rel-18 CR TS 28.105 Corrections including editorial fixes	18.5.0
2024-09	SA#105	SP-241181	0165	1	C	Rel 19 CR TS 28.105 Remove Support Qualifier from attribute constraints	19.0.0
2024-12	SA#106	SP-241639	0181	1	C	Rel-19 CR TS 28.105 Implement readonly attributes for openAPI SS	19.1.0
2024-12	SA#106	SP-241644	0183	1	A	Rel-19 CR TS 28.105 correct the areNewTrainingDataUsed attributes	19.1.0
2024-12	SA#106	SP-241639	0184	-	B	Enhance the isUnique property for stage 3 OpenAPI	19.1.0
2024-12	SA#106	SP-241634	0189	1	A	Rel 19 CR TS 28.105 Make decisionConfidenceScore attr applicable for Training	19.1.0
2024-12	SA#106	SP-241644	0191	2	A	Rel 19 CR TS 28.105 Fix modelPerformanceValidation attribute	19.1.0
2024-12	SA#106	SP-241644	0193	1	A	CR Rel-19 TS 28.105 add missing terms	19.1.0
2024-12	SA#106	SP-241644	0195	-	A	Rel-19 CR TS 28.105 AIML Terminology Alignment	19.1.0
2024-12	SA#106	SP-241639	0197		C	Rel-19 CR TS 28.105 Implement default value for openAPI SS	19.1.0
2024-12	SA#106	SP-241644	0199	1	A	Rel-19 CR TS 28.105 Clarify the Usage of MLModelRef and MLModelCoordinationGroupRef	19.1.0
2024-12	SA#106	SP-241644	0200	1	A	Rel-19 CR TS28.105 Correct stage2 and stage3 definition for aiMLInferenceName	19.1.0
2024-12	SA#106	SP-241639	0201	1	B	Rel19 CR 28.105 Enhance the OpenAPI related to multiplicity	19.1.0
2024-12	SA#106	SP-241634	0204	1	A	Rel-19 CR TS 28.105 Fix trainingRequestSource attribute	19.1.0
2024-12	SA#106	SP-241639	0206		C	Rel-19 CR TS 28.105 add missing inheritance statement for IOC definition	19.1.0
2024-12	SA#106	SP-241644	0211	1	A	Rel 19 CR TS 28.105 Correct information model definitions for ML model deployment	19.1.0
2024-12	SA#106	SP-241644	0214	1	A	Rel 19 CR TS 28.105 Corrections for aiml inference history	19.1.0
2024-12	SA#106	SP-241644	0216		A	Rel 19 CR TS 28.105 Remove allowedValues from attribute definitions where value is same as type	19.1.0
2024-12	SA#106	SP-241643	0217	1	A	Rel-19 CR TS 28.105 Correct description of MLModelCoordinationGroup	19.1.0
2024-12	SA#106	SP-241644	0219	1	A	Rel-19 CR TS 28.105 Add missing attribute to MLModelRepository	19.1.0
2024-12	SA#106	SP-241644	0221	1	A	Rel-19 CR TS 28.105 correction of attribute descriptions and legal values	19.1.0
2024-12	SA#106	SP-241644	0223	1	A	Rel-19 CR TS 28.105 Correction of not used attribute and attribute properties	19.1.0
2025-03	SA#107	SP-250148	0229	1	F	Rel-19 CR TS 28.105 Correction of PlanUML code	19.2.0
2025-03	SA#107	SP-250148	0233	-	F	Rel19 CR TS28.105 correction to data type definitions related to AIMLInferenceName	19.2.0
2025-03	SA#107	SP-250156	0241	1	A	Rel-19 CR TS 28.105 correct the components for ML model deployment and AI/ML inference	19.2.0
2025-03	SA#107	SP-250156	0243	1	A	Rel-19 CR TS 28.105 correct the the requirements for ML model training	19.2.0
2025-03	SA#107	SP-250156	0250	-	A	Rel-19 CR TS 28.105 Correct MLTrainingFunction definition	19.2.0
2025-03	SA#107	SP-250156	0254	-	A	Rel 19 CR TS 28.105 Correct isWritable property of requestStatus for MLLoadingRequest and MLUpdateRequest	19.2.0
2025-03	SA#107	SP-250156	0259	1	A	Rel-19 CR TS 28.105 Correct MLContext attributes	19.2.0
2025-03	SA#107	SP-250156	0266	1	A	Rel-19 CR TS 28.105 Correction of MLTestingRequest attributes	19.2.0
2025-03	SA#107	SP-250156	0268	1	A	Rel-19 CR TS 28.105 Correction of MLModel attribute properties	19.2.0
2025-03	SA#107	SP-250156	0273	1	A	Rel-19 CR TS 28.105 Correct MLTrainingReport attributes	19.2.0
2025-03	SA#107	SP-250156	0275	1	A	Rel-19 CR TS 28.105 Correct MLTrainingRequest attributes	19.2.0
2025-03	SA#107	SP-250156	0276	1	A	Rel-19 CR TS 28.105 Correct MLTrainingProcess attributes	19.2.0
2025-09	SA#109	SP-251108	0290	1	C	Rel-19 TS 28.105 stage 3 update to align with Rel-19 stage 2 specifications	19.3.0

Change history							
Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
2025-09	SA#109	SP-251087	0294	1	A	Rel 19 TS 28.105 Correct missing Associated information entities	19.3.0
2025-09	SA#109	SP-251087	0298	1	A	Rel 19 TS 28.105 Correct Common notifications definition	19.3.0
2025-09	SA#109	SP-251108	0301		B	REL-19 CR to TS 28.105 for AIML_MGT_Ph2	19.3.0

History

Document history		
V19.3.0	October 2025	Publication