ETSI TR 128 908 V18.0.0 (2024-05)

5G;
Study on Artificial Intelligence/Machine Learning (AI/ ML) management
(3GPP TR 28.908 version 18.0.0 Release 18)

Reference

DTR/TSGS-0528908vi00

Keywords

5G

Contents

Foreword

This Technical Report 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 studies 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]), 5GC (e.g. NWDAF, see 3GPP TS 23.288 [3], and NG-RAN (e.g. RAN intelligence defined in 3GPP TS 38.300 [16] and 3GPP TS 38.401 [19]).

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 (MDA)".

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

[4] 3GPP TS 28.105: "Management and orchestration; Artificial Intelligence/Machine Learning (AI/ML) management".

[5] IBM Watson Studio: "Model Drift" [Online].

NOTE: Available at: https://www.ibm.com/cloud/watson-studio/drift.

[6] 3GPP TR 28.864: "Study on Enhancement of the management aspects related to NetWork Data Analytics Functions (NWDAF)".

NOTE: Available at https://www.3gpp.org/dynareport/28864.htm.

[7] 3GPP TS 28.310: "Management and orchestration; Energy efficiency of 5G".

[8] 3GPP TS 28.552: "Management and orchestration; 5G performance measurements".

[9] 3GPP TS 28.313: "Management and orchestration; Self-Organizing Networks (SON) for 5G networks".

[10] European Commission (21.04.2021): "Proposal for a Regulation laying down harmonized rules on artificial intelligence".

[11] High-level Expert Group on Artificial Intelligence setup by the European Commission (08.04.2019): "Ethical Guidelines for Trustworthy AI".

[12] ISO/IEC TR 24028:202: "Information technology -- Artificial intelligence -- Overview of trustworthiness in artificial intelligence".

[13] 3GPP TS 28.622: "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)".

[14] 3GPP TS 28.554: "Management and orchestration;5G end to end Key Performance Indicators (KPI)".

[15] 3GPP TR 37.817: "Study on enhancement for data collection for NR and ENDC".

[16]	3GPP TS 38.300: "NR; NR and NG-RAN Overall description; Stage-2".
[17]	3GPP TS 28.533: "Management and orchestration; Architecture framework".
[18]	3GPP TR 28.813: "Management and orchestration; Study on new aspects of Energy Efficiency (EE) for 5G".
[19]	3GPP TS 38.401: "NG-RAN; Architecture description".
[20]	3GPP TS 28.541: "Management and orchestration of 5G networks; Network Resource Model (NRM); Stage 2 and stage 3".

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in 3GPP TR 21.905 [1], TS 28.105 [4] 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].

3.2 Symbols

Void.

3.3 Abbreviations

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

MAE	Mean Absolute Error
MDCA	Management Data Correlation Analytics
MSE	Mean Squared Error

4 Concepts and overview

4.1 Concepts and terminologies

Biased data: Biased data in machine learning occurs when certain data samples of a training dataset are more heavily weighted and/or overrepresented in comparison to others. Biased data may lead to lower quality predictions and/or reduced accuracy of the trained ML model.

F1 score: (also known as F-measure, or balanced F-score) is a metric used to measure the training performance of classification ML models.

4.2 Overview

Artificial Intelligence/Machine Learning (AI/ML) techniques are being embraced by telecommunication service providers around the world to facilitate enabling the existing and the new challenging use cases that 5G offers. AI/ML capabilities are being increasingly adopted in mobile networks as a key enabler for wide range of features and functionalities that maximise efficiency and bring intelligence and automation in various domains of the 5GS. For example, these include the Management Data Analytics (MDA) in the management and orchestration [1], the Network Data Analytics Function (NWDAF) in the 5G core network domain [3], and NG-RAN (e.g. RAN intelligence) defined in 3GPP TS 38.300 [16] and 3GPP TS 38.401 [19].

The AI/ML inference functions in the 5GS use the ML model for inference and in order to enable and facilitate the AI/ML adoption, the ML model needs to be created, trained and then managed during its entire lifecycle.

To enable, facilitate and support AI/ML-capabilities in the 5GS, the following management capabilities are studied in the present document:

- Validation of ML model or entity.
- Testing of ML model or entity (before deployment).
- Deployment of ML model or entity (new or updated model/entity).
- Configuration of ML training and AI/ML inference.
- Performance evaluation of ML training and AI/ML inference.

NOTE: The ML model training capability is specified in 3GPP TS 28.105 [4].

4.3 AI/ML workflow for 5GS

4.3.1 AI/ML operational workflow

AI/ML techniques are widely used in 5GS (including 5GC, NG-RAN and management system), and the generic workflow of the operational steps in the lifecycle of an ML model or entity, is depicted in the figure 4.3.1-1.

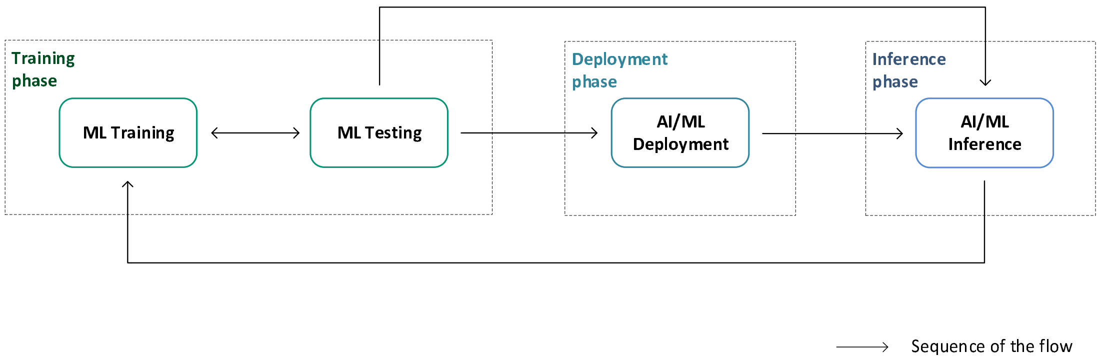


Figure 4.3.1-1: AI/ML operational workflow

The workflow involves 3 main phases; the training, deployment and inference phase, including the main operational tasks for each phase. These are briefly described below:

Training phase:

- **ML Training:** Learning by the Machine from the training data to generate the (new or updated) ML entity (see 3GPP TS 28.105 [4]) that could be used for inference. The ML Training may also include the validation of the generated ML entity to evaluate the performance variance of the ML entity when performing on the training data and validation data. If the validation result does not meet the expectation (e.g. the variance is not acceptable), the ML entity needs to be re-trained. This is the initial step of the workflow. The ML Training MnS is specified in 3GPP TS 28.105 [4].

- **ML Testing:** Testing of the validated ML entity with testing data to evaluate the performance of the trained ML entity for selection for inference. When the performance of the trained ML entity meets the expectations on both training data and validation data, the ML entity is finally tested to evaluate the performance on testing data. If the testing result meets the expectation, the ML entity may be counted as a candidate for use towards the intended use case or task, otherwise the ML entity may need to be further (re)trained. In some cases, the ML entity may need to be verified which is the special case of testing to check whether it works in the AI/ML inference function or the target node. In other cases, the verification step may be skipped, for instance in case the input and output data, data types and formats, have been unchanged from the last ML entity.

Deployment phase:

- **ML Deployment:** Deployment of the trained and tested ML entity to the target inference function which will use the subject ML entity for inference.

NOTE: The deployment phase may not be needed in some cases, for example when the training function and inference function are in the same entity.

Inference phase:

- **AI/ML Inference:** Performing inference using the ML entity by the inference function.

In telco-grade environments, it is worth noting that the selected learning method (see examples of learning methods captured in table 4.1-1 in 3GPP TS 28.105 [4]) can influence on how AI/ML operational workflow executes. In some cases (e.g. when using supervised learning methods), the inference phase cannot start until training phase gets ended. In other cases (e.g. when using reinforcement learning methods), the inference phase can start while training phase is still in progress.

4.3.2 AI/ML management capabilities

Each operational step in the workflow (as depicted in clause 4.3.1) may be related to one or more AI/ML management capabilities, including:

Management capabilities for training and testing phase

- **ML training data management:** This involves management capabilities for managing the data needed for training the ML entities. It may also include capabilities for processing of data as requested by a training function, by another management function or by the MLT MnS consumer.
- **ML training management:** allowing the MnS consumer to request and/or manage the model training/retraining. For example, activating/deactivating, training performance management and setting policy for the producer-initiated ML training (e.g. the conditions to trigger the ML (re-)training based on the AI/ML inference performance or AI/ML inference trustworthiness).
- **ML testing management:** allowing the MnS consumer to request the ML entity testing, and to receive the testing results for a trained ML model. It may also include capabilities for selecting the specific performance and trustworthiness metrics to be used or reported by the ML testing function.
- **ML validation:** ML training capability may also include validation to evaluate the performance and trustworthiness of the ML entity when performing on the validation data, and to identify the variance of the performance and trustworthiness on the training data and the validation data. If the variance is not acceptable, the entity would need to be tuned (re-trained) before being made available to the consumer and used for inference.

Management capabilities for deployment phase

- **AI/ML deployment control and monitoring:** This involves capabilities for loading the ML entity to the target inference function. It includes providing information to the consumer when new entities are available, enabling the consumer to request the loading of the ML entity or to set the policy for such deployment and to monitor the deployment process.

Management capabilities for inference phase

- **ML entity activation/deactivation:** allowing the MnS consumer to activate/deactivate the inference function and/or ML entity/entities, including instant activation, partial activation, schedule-based or policy-based activations.
- **AI/ML inference function control:** allowing the MnS consumer to control the inference function including the activation and deactivation of the function.
- **AI/ML inference performance management:** allowing the MnS consumer to monitor and evaluate the inference performance of an ML entity when used by an AI/ML inference function.
- **AI/ML trustworthiness management:** allowing the MnS consumer to monitor and evaluate the inference trustworthiness of an ML entity when used by an AI/ML inference function.

- **AI/ML inference orchestration:** enabling MnS consumer to orchestrate the AI/ML inference functions (e.g. by setting the conditions to trigger the specific inferences) with the knowledge of AI/ML capabilities, the expected and actual running context of ML entity, AI/ML inference performance, AI/ML inference trustworthiness, etc.

5 Use cases, potential requirements and possible solutions

5.1 Management Capabilities for ML training phase

5.1.1 Event data for ML training

5.1.1.1 Description

In analytics solutions, Performance Measurements (PMs) and alarm data from various network functions and/or management functions are collected and analysed (e.g. by employing MDA) to derive events (statistical insights and predictions). For most algorithms, the prediction accuracy depends upon the amount of available relevant historical data, motivating the need to store ever more data, which correspondingly increases the storage and processing resource requirements. However, it is possible that not all recorded data is useful as the derived events, e.g. captured through analytics processes, may have loss of information OR misinformation e.g. with respect to time of the event. As such the collected raw data as well as the derived events may be pre-processed both in terms of volume and quality in order to produce optimised/accurate training data.

5.1.1.2 Use cases

5.1.1.2.1 Pre-processed event data for ML training

For AI/ML algorithms, a large amount of data points does not necessarily add value, e.g. if most of it includes biased data which ends up getting discarded during the pre-processing stages. Instead, the AI/ML algorithms need to have information-rich events data that is condensed but with most of it useful for the required training. For example, one could train an interference optimization solution that learns the best way to combat interference by looking at correlated events which are derived from PM counters of handover failures correlated with signal quality.

It is as such necessary to provide means to isolate and store the information rich network events in the network, to ensure that minimizing storage and processing costs by discarding the unnecessary/redundant raw data does not compromise the ability to and still avails adequate historical information to adequately train ML entities. In other words it is necessary for network functions and their management system to generate data about the observed network events, e.g. based on the criteria set by the MnS consumer (e.g. operator). The network events can then be stored to be used later to train ML entities.

Here, a network event represents a specific combination of activities executed on the network or occurrences in the network identified say by a set of instances of threshold crossings.

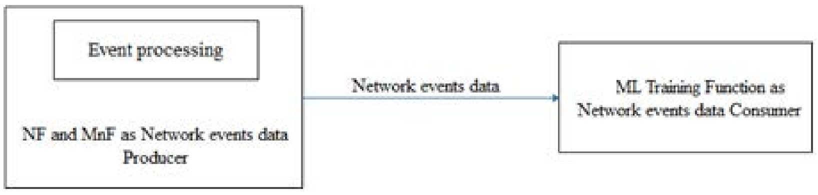

Event processing
NF and MnF as Network events data Producer
Network events data
ML Training Function as Network events data Consumer

Figure 5.1.1.2.1-1: Exposing and storing network events data

As maybe seen in figure 5.1.1.2.1-1, there could be multiple sources for the same event data e.g. either the source network function or a related management function. Correspondingly, it may be necessary to eliminate duplications of events or event data that is exposed to the AI/ML inference functions. It may in such cases make sense to have a single events data exposure entity, say called a network events aggregator that exposes data from multiple sources but without duplications.

Relatedly, when an MnS consumer for ML training producer wishes to trigger a training based on such events and not on specific raw data, such a consumer should be enabled to request for such training and clarify that they have training executed based on network events.

5.1.1.3 Potential requirements

REQ-EVENT-DATA-1: The 3GPP management system should enable an authorized consumer to request from the network data producer for network events corresponding to the data produced by that network data producer.

REQ-EVENT-DATA-2: The 3GPP management system should enable a network data producer to generate network events.

REQ-EVENT-DATA-3: The 3GPP management system should enable a network events aggregator to take the events from different network entities and re-expose them in an aggregated way that eliminates duplications.

REQ-EVENT-DATA-4: The 3GPP management system should enable an authorized MnS consumer to configure the ML Training MnS producer to provide training based on a set of network events.

REQ-EVENT-DATA-5: The 3GPP management system should have a capability to inform an authorized MnS consumer when training is triggered based on an MnS consumer's request for training based on a set of network events.

5.1.1.4 Possible solutions

1) Introduce an IOC for processing data to identify network events. The IOC may be called a Network Events Processing Agent (named e.g. NetworkEventsProcessor or NEPA). The NEPA may be associated with data collection and exposure services and may be contained in any ManagedFunction, ManagementFunction or subnetwork. This NEPA is responsible for generating the events data which can be availed to the ML training function, i.e. the training function may configure the NEPA to compile and generate network events with certain characteristics:

    a. The ML training functions may configure the events on any one or more data sources according to the kinds of events needed by the ML training functions(s).

    b. The data source provides the events to the ML training functions(s) as configured, ML training functions(s) executes the training using those events.

2) Introduce an information Model for a Network Event as the datatype that can be exposed by an NEPA. The model may distinguish between Metric-Threshold Crossing events and Object-Status Change Events:

    a. Metric Threshold Crossing events derived from a threshold or a combination of multiple thresholds, i.e. the Metric Threshold Crossing events are the events captured when one or more metrics cross the one or more configured thresholds. The definition of the Metric Threshold Crossing events may reuse the threshold monitoring MnS defined in 3GPP TS 28.550 as well as the related NRMs defined in 3GPP TS 28.622 [13]. Thereby, new parameters and metrics for which thresholds can be set may have to be defined.

    b. Object Status Change Events are the events captured when the status of a managed object instance changes be it due to internal processes within the MOI or due to external actors doing something on the MOI.

3) Introduce a Status-Change-monitoring IOC contained by any ManagedElement, ManagedFunction, or ManagementFunction, to enable an MnS consumer to define Object Status Change Events.

4) Define/specify a set of network events for different managed objects. Example network events include.

Table 5.1.1.4-1: Examples of potential Metric Threshold Crossing events

Event-source Function	Event Name	Event description	Input/Metric	MOI	Condition	Threshold	Monitoring Period
BTSs; NBs; eNBs; gNBs; BSC; RNC	High Call Drop Rate event	Call Drop Rate more than a configurable threshold	Call Drop Rate	Cell	greater than	2 %	15 minutes
	Low Availability KPIs event	Availability KPIs dropping below a configurable threshold	Availability	Cell	less than	99 %	30 minutes
	Low Retainability KPIs event	Retainability KPIs dropping below a configurable threshold	Retainability	Cell	less than	98 %	30 minutes
	High Traffic event	Traffic greater than a configurable threshold	Cell load / PRB Utilization	Cell	greater than	80 %	15 minutes
	Interference event	User has experienced interference	SINR	Cell	less than	X dB	15 minutes
			Serving cell RSRP	Cell	greater than	Y dBm	15 minutes

Note that some events are the results of combinations of multiple thresholds (derived from the success of multiple thresholds monitoring).

Table 5.1.1.4-2: Examples of potential Object-Status Change events

Event-source Function	Event Name	Event description	MOI	Affected unit /parameter	Change/ value
BTSs; NBs; eNBs; gNBs; BSC; RNC; NMS	HW Upgrade event	System Module HW version upgraded	BTS	System Module, Radio Module, …	HW version
	SW Upgrade event	System Software Upgraded	BTS	System Module, Radio Module, …	SW version
	Capability Enablement event	A specific Capability Enabled on the MOI	BTS	Spectrum Sharing	Spectrum Range for affected RATs
	New Sector Addition Event	A new Sector getting added to a Site	Cell	Capacity and Coverage	Number of Sectors / Cells
Network Management	Parameter change event	CM Parameter changes applied for specific network element	Cell	Configuration Parameter	Parameter value
	Home status event	MOI (e.g. site) Re-homing	Site	BSC/RNC, OSS	New BSC/ RNC/ OSS
SON, Analytics function	New Site event	New Site Integrated	Site/ gNB	C-SON Functions	Optimization Parameters in C-SON
	Predicted Congestion	Trigger for Load Balancing detected	Cell	C-SON LBO	Mobility Parameter changes
	Frequent Handover Failures	Trigger for MRO detected	Cell	C-SON MRO	Handover Parameter Changes
	PCI Conflict	PCI conflict detected	Cell	C-SON PCI	PCI Changes
	PRACH Conflict	PRACH conflict detected	Cell	C-SON PRACH	PRACH related parameter Changes
	NCR Change	New First tier neighbour getting added	Cell	C-SON ANR	NCR Changes
	Frequency Layer Change	New Frequency Layer added onto a site	BTS	C-SON	Frequency Layer Addition

5.1.1.5 Evaluation

The solution described in clause 5.1.1.4 reuses the existing provisioning MnS operations and notifications in combination with extensions of the NRM. This solution is also consistent with the approach used by ML training MnS in 3GPP TS 28.105 [4] where the training MnS producer is configured with the needs to train using events data based on the MnS consumer's request for the data services that provide events data. The solution provides the flexibility to allow any function to be the MnS producer for the network events data and for any training producer to consume that data for its training.

Therefore, the solution described in clause 5.1.1.4 is a feasible solution to be developed further in the normative specifications.

5.1.2 ML entity validation

5.1.2.1 Description

During the ML training process, the generated ML entity (see 3GPP TS 28.105 [4]) needs to be validated. The purpose of ML validation is to evaluate the performance of the ML entity when performing on the validation data, and to identify the variance of the performance on the training data and the validation data. If the variance is not acceptable, the entity would need to be tuned (re-trained) before being made available to the consumer and used for inference.

The training data and validation data are normally split from the same data set with a certain ratio in terms of the quantity of the data examples, therefore they have the same pattern. The training data set is used to create (fine-tune) the ML entity, while the validation data set is used to qualify performance of the trained entity.

5.1.2.2 Use cases

5.1.2.2.1 ML entity validation performance reporting

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

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

5.1.2.3 Potential requirements

REQ-MODEL_VLD-CON-1: The MLT MnS producer should have a capability to validate the ML entities during the training process and report the performance of the ML entities on both the training data and validation data to the authorized consumer.

REQ-MODEL_VLD-CON-2: The MLT MnS producer should have a capability to report the ratio (in terms of the quantity of the data examples) of the training data and validation data used for training of an ML entity during the training process.

5.1.2.4 Possible solutions

5.1.2.4.1 Validation performance reporting by enhancing the existing IOC

In 3GPP TS 28.105 [4], the ML entity training report is provided by `MLTrainingReport` IOC, which includes the attribute indicating the performance of the ML entity when performing on the training data.

To support the ML entity validation performance reporting, a new optional attribute can be defined in the `MLTrainingReport` IOC to indicate the performance of the ML entity when performing on the validation data.

5.1.2.5 Evaluation

The possible solution described in clause 5.1.2.4.1 enhances the existing `MLTrainingReport` IOC with a new optional attribute indicating the performance of the ML entity when performing on the validation data, the change is lightweight and it is backward compatible, therefore it is a feasible solution.

5.1.3 ML entity testing

5.1.3.1 Description

After an ML entity is trained, validation is done to ensure the training process is completed successfully. However, even when validation is conducted successfully during ML entity development, it is necessary to test and check if the ML entity is working correctly under certain runtime contexts or using certain testing data set. Testing may involve interaction with third parties (besides the ML training MnS producer (MLT function), e.g. the operator may use the ML training function or third-party systems/functions that may rely on the results computed by the ML entity for testing.

After completing the ML entity training, and when the performance of the trained ML entity meets the expectations on both training and validation data, the ML entity is made available to the consumer(s) via the ML training report (see MLTrainingReport IOC in 3GPP TS 28.105 [4]). Before applying the ML entity to the target AI/ML inference function, the ML training MnS producer may need to allow the consumer to evaluate the performance of the ML entity via the ML testing process using the consumer's provided testing data. The testing data have the same pattern as the input part of the training data.

For these reasons, provision of ML entity testing, and its control need to be standardized to enable the multi-vendor interaction among the different systems. If the testing performance is not acceptable or does not meet the pre-defined requirements, the consumer may request the ML training producer to re-train the ML entity with specific training data and/or performance requirements.

5.1.3.2 Use cases

5.1.3.2.1 Consumer-requested ML entity testing

After receiving an ML training report about a trained ML entity from the ML training MnS producer, the consumer may request the testing MnS producer to test the ML entity before applying it to the target inference function. In the ML testing request, the consumer provides the testing data which have the same pattern as the input part of the training data.

Any ML entity needs to be tested with specific inputs and features that are applicable to the use case and the applicable deployment environment.

The ML testing MnS producer performs the ML testing using the consumer's provided testing data. The ML testing is to conduct inference on the tested ML entity using the testing data as the inference inputs and produce the inference output for each testing dataset example.

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

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

The ML testing MnS producer needs to have the capabilities to provide the services needed to enable the consumer to request testing and receive results on the testing of a specific ML entity or of an application or function that contains an ML entity.

To achieve the desired outcomes, any ML entity needs to be tested with the appropriate testing data (e.g. batch data or continuous data streams), which can reflect the current status of the network where the ML entity is expected to be deployed. Correspondingly, the ML testing MnS producer needs to support the required management services to test the ML entities.

5.1.3.2.2 Control of ML entity testing

Given a testing capability as provided by a given ML testing MnS producer, a consumer (e.g. an operator) may wish to control and manage that testing process capability. For example, the operator may wish to define policies on how frequent testing for a given ML entity may be executed. Correspondingly, the 3GPP management system needs to provide the capability to allow the ML entity testing to be configured.

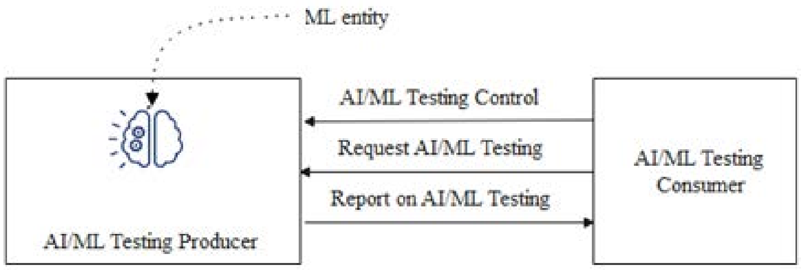


Figure 5.1.3.2.2-1: ML entity testing and control

5.1.3.2.3 Multiple ML entities joint testing

For a given use case, different entities apply the respective ML model to fulfil different inference requirements and capabilities. However, in some cases, multiple ML entities may be worked in a synergic manner for complex use cases, in the case an ML entity is just one step in the production process to implement some specific function, with the analytics outputs of ML entity as the inputs to the next ML entity. For example, the output of Inter-gNB beam selection optimization could be used as input for Handover optimization analysis.

After the joint training of ML entities, to test and check if the ML entities can work correctly under certain runtime contexts, the consumer may request the joint testing to verify whether multiple ML entities can synergically work before applying it to the target inference function. Hence, the 3GPP management system needs to provide the capability to allow multiple ML entities joint testing.

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

5.1.3.2.4 Model evaluation for ML testing

In the ML entity training phase, the ML entity is generated based on the learning from the training data, while performance and trustworthiness will be evaluated on validation data. When the performance and trustworthiness of the trained ML entity meets the expectations on both training and validation data, the ML entity is made available to the consumer(s).

However, it does not mean the ML entity could have good performance and trustworthiness on completely unseen real-world data. The ML entity should be finally tested and evaluated on testing data.

After the ML testing MnS producer performs the ML testing on the testing data, the performance and trustworthiness of the ML entity/entities needs to be evaluated.

The ML testing MnS producer uses one or more ML entities for testing and generates the inference output. In order to understand the behaviours and performance of the ML entity/entities, the ML testing function may support reporting the testing results with related performance and trustworthiness metrics and may support to evaluate each kind of ML model by one or more specific corresponding performance and trustworthiness indicators.

5.1.3.3 Potential requirements

REQ-AI/ML_TEST-1: The ML testing MnS producer should have a capability to enable an authorized consumer to request the testing of a specific ML entity.

REQ-AI/ML_TEST-2: The ML testing MnS producer should have a capability to create a testing process instance per the testing request for an authorized consumer.

REQ-AI/ML_TEST-3: The ML testing MnS producer should have a capability to report to an authorized consumer the results of a specific instance of ML testing process with the result of a successful ML entity testing containing the inference output for each testing data set example.

REQ-AI/ML_TEST-4: The ML testing MnS producer should have a capability to enable an authorized consumer (e.g. the operator) to configure or modify an instance of ML testing process.

REQ-AI/ML_TEST-5: The ML testing MnS producer should have a capability to test a specific ML entity using specific data set specified by the consumer, using data set at a location address specified by the consumer, using data set with specific characteristics defined by the consumer, or using continuous data streams provided from the consumer.

REQ-AI/ML_TEST-6: The ML testing MnS producer should have a capability to test a specific ML entity for a specified expected runtime context as may be stated by the consumer.

REQ-AI/ML_TEST-8: The ML testing MnS producer should support a capability to enable an authorized consumer to define the reporting characteristics related to a specific instance of ML testing request.

REQ-AI/ML_TEST-9: The ML testing MnS producer should support a capability for an authorized consumer to manage the ML testing request, including suspending, resuming, cancelling the request, or adjusting the desired runtime context of the testing.

REQ-AI/ML_TEST-10: The ML testing MnS producer may have a capability for an authorized consumer to request the joint testing of multiple ML entities.

REQ-AI/ML_TEST-11: The ML testing MnS producer should have a capability to evaluate the ML entities during the testing process and report the performance metrics and trustworthiness metrics of the ML entities to the authorized consumer.

5.1.3.4 Possible solutions

5.1.3.4.1 NRM based solution

This solution uses the instances of following IOCs for interaction between ML testing MnS producer and consumer to support the ML entity/entities testing:

1) The IOC representing the ML entity testing request, for example named as `MLTestingRequest`.

   This IOC is created by the ML entity testing MnS consumer on the producer, and it contains the following attributes:

   - identifier of the ML entity to be tested;

   - testing environment requirements, e.g. expected runtime context;

   - testing data.

2) The IOC representing the ML entity testing policy, for example named as `MLTestingPolicy`.

   This IOC is created by the ML entity testing MnS consumer on the producer to control the testing initiated by the producer, and it contains the following attributes:

   - identifier or inference type of the ML entity to be tested;

   - testing environment requirements, e.g. expected runtime context;

   - testing triggers, i.e. the conditions that would trigger the testing of an ML entity;

   - relationship between multiple ML entities, e.g. ML entity sequence, ML entity hierarchical order.

3) The IOC representing the ML entity testing process, for example named as `MLTestingProcess`. This MOI is created for the ML entity testing process corresponding to the testing requested by the consumer per the IOC described in 1), or the testing initiated by the producer based on the given testing policy per the IOC described in 2).

This IOC is created by the ML entity testing MnS producer and reported to the consumer, and it contains the following attributes:

- identifier(s) of the ML entity/entities being tested;
- the associated ML entity testing request;
- the associated ML entity testing policy;
- testing progress;
- testing environment, e.g. the testing runtime context;
- testing data to be used;
- control of the process, like cancel, suspend and resume.

4) The IOC representing the ML testing report, for example named as `MLTestingReport`.

This IOC is created by the ML testing MnS producer and reported to the consumer, and it contains the following attributes:

- identifier of the tested ML entity;
- the associated ML entity testing request;
- the associated ML entity testing process;
- testing result indicating the success or failure and containing the inference output for each testing data example for successful case, and the failure reason for the failed case;
- performance metrics (e.g. "accuracy", "precision", "F1 score") and trustworthiness metrics (see AI/ML trustworthiness indicators defined in clause 5.3.1.2.1) of the ML entity when performing on the testing data.

The examples of IOCs and their relations between the IOCs are depicted in figure 5.1.3.4.1-1.

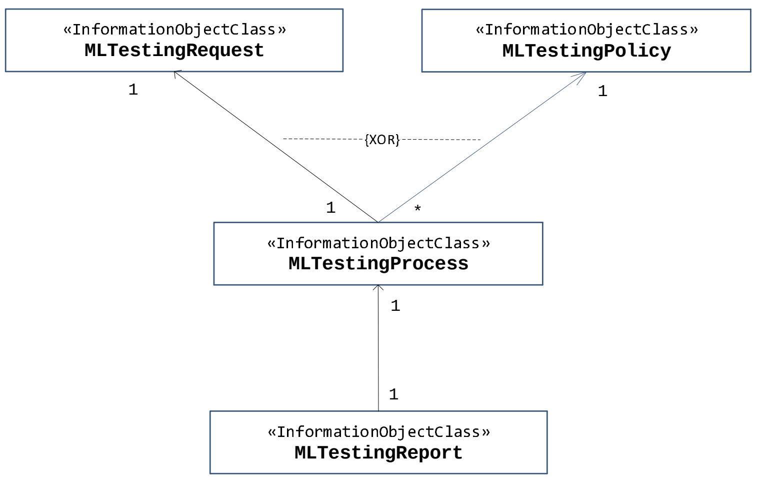


Figure 5.1.3.4.1-1: Example of ML entity testing related NRMs

NOTE: The name of the IOCs and attributes are to be decided in normative phase.

5.1.3.5 Evaluation

The solution described in clause 5.1.3.4.1 adopts the NRM-based approach, which reuses the existing provisioning MnS operations and notifications. This solution is also consistent with the approach used by ML training MnS defined in 3GPP TS 28.105 [4]. It does not only reuse the existing capabilities (provisioning MnS operations and notifications) to a greater extent, but also provides the flexibility to facilitate both co-located and distributed implementation and deployment of ML training MnS and ML testing MnS producers by using the consistent NRM-based approach.

Therefore, the solution described in clause 5.1.3.4.1 is a feasible solution.

5.1.4 ML entity re-training

5.1.4.1 Description

A trained ML entity is to support a specific type of inference. To improve the performance of the ML entity for conducting the same type of inference, the ML entity needs to be re-trained in the situation when e.g. the inference performance degrades or when the ML entity running context changes.

The ML entity re-training refers to the process of re-training the ML entity using new training data to make the ML entity to be more adaptive to the new data pattern, without changing the type of inference (i.e. the types of inference input and output).After a successful re-training, a new version of ML entity is generated with the ability to make the same type of inference as the old ML entity.

5.1.4.2 Use cases

5.1.4.2.1 Producer-initiated threshold-based ML entity re-training

The performance of the ML entity depends on the degree of commonality of the distribution of the data used for training in comparison to the distribution of the data used for inference. Typically, the model performance would be good only for a limited period after deployment. This is because the chances of the distributions of the data used for training and the samples picked for inference are the same. As the time progresses, the distribution of the network data might change as compared to the distribution of the training data. In such scenarios, the performance of the ML entity degrades over time. Hence there is a need for monitoring the performance of the network using counters and thresholds, such as PMs, KPIs alarms etc., and use this information in the ML training producer to decide on the re-training.

5.1.4.2.2 Efficient ML entity re-training

During inference phase of ML entity, a lot of potentially new data samples are processed and some of them are useful for a re-training and should therefore be labelled and added to the training set. However, using all inference data samples for re-training generates a need for high effort and resources for data labelling, data provision (signalling) and model training, and this effort is not feasible in environments with limited resources.

In the case that re-training/model adaptation is performed at the entity under the conditions of low processing power and/or limited energy consumption, then the amount of data used for re-training and the time needed for model to converge towards maximum performance is critical and therefore need to be minimized.

In order to optimize the re-training, it is necessary to reduce the number of training samples, by extracting the most supporting data samples for re-training from all available data samples (pool samples) that have been used for inference. For example, the training data can be further enriched/optimized by deriving key events with events processing technique instead of using volume data (see clause 5.1.1).

5.1.4.3 Potential requirements

REQ-AIML_RETRAIN-1: The ML training MnS producer should have a capability allowing an authorized AI/ML MnS consumer to provide the counters and thresholds to be monitored to trigger the re-training of an ML entity.

REQ-AIML_RETRAIN-2: The ML training MnS producer should have a capability allowing an authorized AI/ML MnS consumer to update the ML entity with counters data and thresholds to be monitored to trigger the re-training of that ML entity.

REQ-AIML_RETRAIN-3: The 3GPP management system should have a capability for the authorized MnS consumer to request and receive from the AI/ML inference producer the most supporting data samples/events for re-training from all data samples (pool samples) that have been used for AI/ML inference.

REQ-AIML_RETRAIN-4: The 3GPP management system should have a capability for the AI/ML MnS inference producer to provide to the ML training MnS producer the most supporting data samples and/or monitored events for re-training.

5.1.4.4 Possible solutions

5.1.4.4.1 Producer Initiated Retraining

Following is the proposed solution based on information model defined in 3GPP TS 28.105 [4]:

- Extend the existing `MLTrainingRequest` IOC with an optional `<<datatype>>` attribute on monitored data events. The attribute may be called "monitoredDataEvents" and is a list of monitored data events each of which may be of `<<datatype>>` "monitoredDataEvent" that contains the following information:
    - An attribute called "`ThresholdInfoList`" as a list of threshold information with each entry a "`ThresholdInfo`" <<`datatype>>` as defined in 3GPP TS 28.622 [13]. The `ThresholdInfo` is an array containing:
        1) the performance metrices to be monitored and collected by the AIML inference producer;
        2) the threshold value;
        3) threshold directions indicating the direction for which a threshold crossing triggers a threshold; and
        4) the threshold hysteresis indicating hysteresis of a threshold, if configured, the PM is not compared only against the threshold value but also considering the hysteresis value.
    - An attribute called "`MonitoredkPIList`" as a list of KPIs t be monitored for the particular data event. Each entry of the "`MonitoredkPIList`" is a "`kPIName`" indicating the name of the KPI as defined in 3GPP TS 28.554 [14] to be monitored for this ML training.
- Existing `MLEntity <<datatype>>` is extended with the same information mentioned above. This is needed to ensure an MnS consumer can configure the ML entity and by doing so trigger the ML retraining. ML training producer may monitor the information available at `MLEntity <<datatype>>` and when any of the thresholds is crossed, retraining may be performed by the ML training producer. The threshold crossing may be identified via direct monitoring of the ML entity by the retraining producer e.g. via data monitoring IOC or via a notification to the retraining producer.

5.1.4.4.2 Efficient ML entity re-training

This solution uses the instances of following IOCs for interaction between ML inference MnS producer and MnS consumer (e.g. the ML training function) to support efficient re-training of ML entity:

- `MLDataSamplesRequest` - this IOC represents the request for obtaining the data samples that are likely to have more value for re-training among all data samples that have been used for inference. This IOC allows an MOI to be created on the ML inference MnS Producer and may contain the following attributes:
    - definition of one or more data samples/events features;
    - minimum number of data samples/events to be obtained;
    - criteria for obtaining the most supporting data samples/events.

  All data samples/events that have been used for inference are filtered in accordance with the one or more requested features and other provided criteria in order to obtain the most supporting data samples/events to allow efficient ML entity re-training.
- `MLDataSamplesResponse` - this IOC represents the response indicating the data obtained according to the `MLDataSamplesRequest`. This IOC is created by the AI/ML MnS inference producer towards the MnS consumer and includes at least the requested minimum number of data samples/events or pointers to them that satisfy criteria specified in `MLDataSamplesRequest`. The response may further include additional information quantifying the supportiveness for each collected data sample/event.

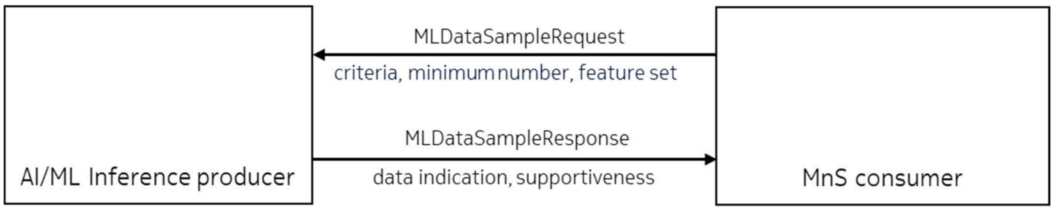


Figure 5.1.4.4.2-1: Interaction between AI/ML inference MnS producer and MnS consumer (e.g. the ML training function) to support efficient re-training of ML entity

5.1.4.5 Evaluation

The solution described in clause 5.1.4.4.1 adopts the NRM-based approach, which reuses the existing provisioning MnS operations and notifications. This solution is also consistent with the approach used by ML training MnS defined in 3GPP TS 28.105 [4]. Moreover, it also enables the MnS consumer to configure existing management data relevant for ML training like PMs and KPIs. Therefore, the solution described in clause 5.1.4.4.1 is a feasible solution.

The solution described in clause 5.1.4.4.2 is consistent with NRM- based approach and reuses existing provisioning MnS operations. The solution is also consistent with the approach used by ML training MnS described in 3GPP TS 28.105 [4]. It provides the means for obtaining the data samples that are likely to have more value for re-training using the consistent NRM-based approach.

5.1.5 ML entity joint training

5.1.5.1 Description

An AI/ML inference function may use one or more ML entities to perform the inference(s). When multiple ML entities are employed, these ML entities may operate together in a coordinated way, such as in a sequence, or even a more complicated structure. In this case, any change in the performance of one ML entity may impact another, and consequently impact the overall performance of the whole AI/ML inference function. Therefore, it is desirable that these coordinated ML entities can be trained or re-trained jointly, so that the group of these ML entities can complete a more complex task jointly with better performance.

5.1.5.2 Use cases

5.1.5.2.1 Support for ML entity modularity - joint training of ML entities

Besides the discovery of the capabilities of ML entities, services are needed for identifying which AI/ML capabilities are used in specific use case and how. 3GPP TS 28.105 [4] defines the `inferenceType` which indicates the type of inference, i.e. the use case that the ML model supports. This indicator may be represented by the MDA type (see 3GPP TS 28.104 [2]), Analytics ID(s) of NWDAF (see 3GPP TS 23.288 [3]), types of inference for RAN-intelligence, and vendor's specific extensions.

In order to address complex use cases, applying multiple, cooperative ML entities might be necessary. There are different ways in which the ML entities may cooperate. An example is the case where the output of one ML entity can be used as input to another ML entity forming a sequence of interlinked ML entities. Another example is the case where multiple ML entities provide the output in parallel (either the same output type where outputs may be merged (e.g. using weights), or their outputs are needed in parallel as input in the automation process or as input to another ML entity. Such modular approach in building a single AI/ML inference function for a given use case facilitates the reusability of ML entities in different use cases. Furthermore, it facilitates the replacement, changes and improvements of individual ML entities ties within complex use cases.

Based on the use case complexity, a single or multiple ML entities may be needed in order to provide a complete solution for a given use case. For simple use cases, it may be sufficient to apply only a single ML entity. For complex use cases or vendor specific extensions, multiple ML entities need to be employed in a potentially coordinated way, e.g. ML entities employed in sequence, parallel or any structure thereof. Given the complexity of the required mapping between the use cases and potentially multiple ML entities, management services should be supported to facilitate such mapping, e.g. determination of whether the specific use case can be realized by a single ML entity or a group of ML entities, configuration of individual ML entities based on their interdependencies in the group, enabling joint training (e.g. re-training) of interdependent ML entities, etc.

5.1.5.3 Potential requirements

REQ-ML_MOD-1: The 3GPP Management system should have a capability for an authorized MnS consumer to request the training of a group of ML entities working together to address specific use case.

REQ-ML_MOD-2: 3GPP management system should have the capability to enable an authorized MnS consumer to manage and configure multiple training processes, e.g. to start, suspend or restart the training; or to adjust the training conditions and/or characteristics based on the ML entity group.

REQ-ML_MOD-3: The MLT MnS producer should have a capability to provide the ML entities group training result (including the location of the trained ML entities in the group) to the authorized MnS consumer.

5.1.5.4 Possible solutions

5.1.5.4.1 Support for ML entity modularity - joint training of ML entities

The IOC `MLTrainingRequest` represents the ML model training request that is created by the ML training MnS consumer. In order to support joint training of a group of ML entities this IOC needs to capture the information on the ML entities group and the relation among the ML entities, i.e. `MLEntityGroupProfile <<datatype>>`. Such data type may contain following attributes:

- `MLEntityGroupID` - Unique identity value identifies the ML entity group instance.
- `JointTrainingIndicator` - which indicates if the ML entity group instance needs to be trained (perform joint training of ML entities in the group).
- Levels - An integer range (1, n) that indicate the ML entity group has n levels, where an integer in the range indicates a specific level, starting from 1. Each level would consist of an ML entity. The output data of one level are used as the input data of the next level:
    - Parallels - a sub integer range (1, m) may be added to indicate the series or the parallel arrangement of ML entities inside a given level, starting from 1. The output data of parallel ML entities inside a given level are used as the input data of the next level.
- `expectedRunTimeContext` - This may include information related to specific extraction, transformation, and load of data as input to a specific ML entity inside a given level and parallel.
- `MLEntitiyID` inside a given level and parallel.

If multiple ML models need to be trained jointly (in relation which each other) the MnS producer needs to start the training based on the information obtained in the `MLEntityGroupProfile`. The ML training MnS producer instantiates multiple `MLTrainingProcess` MOI(s) that are responsible to perform the following:

- collects data for training, taking into account the inter-relation among ML entities in the MLEntityGroupProfile, e.g. if output of first ML entity is used as input to the second ML entity, the data for training of the second model needs to be collected accordingly;
- prepares the training data for each ML entity in the group, based on the information in `MLEntityGroupProfile`. I.e. based on `expectedRunTimeContext` contained in `MLEntityGroupProfile` the specific extraction, transformation, and load of data as input to a specific ML entity inside a given level and parallel needs to be performed;
- trains the ML entities based on the `JointTrainingIndicator`.

5.1.5.5 Evaluation

The solution described in clause 5.1.5.4.1 is consistent with the `MLTrainingRequest` IOC and enhances it in order to support joint training of a group of ML entities. The new attributes added to the `MLTrainingRequest` IOC are to configure the joint training of a group of ML entities. It is a fully NRM-based approach and reuses the existing provisioning MnS operations. It provides the means to facilitate both capturing the information on the group of ML entities working together to address specific use case, as well as enabling the configuration of joint training of such group of ML entities using the consistent NRM-based approach.

5.1.6 Training data effectiveness reporting and analytics

5.1.6.1 Description

For ML model training, a large amount of data instances does not necessarily add value, e.g. if only a subset of the data contributes to actual model training, the rest will be discarded by some well-designed algorithms. During ML model training the information can be provided on whether a specific sample is useful for the training or not and on how much such a sample is useful for the training.

5.1.6.2 Use cases

5.1.6.2.1 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 particular measurement type among all types of input training data) to ML model training process.

To train a ML model, high quality and large volume of training data instances are mandatory. The general practice is to collect as much data as possible and feed them to the ML model for pre-processing and training, in the hope to get high quality of trained model. This is usually done without considering the possibly different contributions of the different portions of input data samples to the accuracy of the trained model. However, this open use of all available data can be costly, both in terms of data collection process and also from a computational resources perspective since the data also contains the unnecessary data samples that are computed through the ML model. One solution is to leave the challenge to the specific ML model training function to optimize the training data usage during model training, or before every training or retraining, e.g. by simply resampling the training data to only use part of the collected training data. However, this method can be inappropriate, especially that in mobile networks where the amount of data can be quite large, resources are constrained while ML models need to still be very accurate and optimally trained,. Instead, it is better that the training function evaluates the usefulness of different data samples or features and indicates that usefulness to the consumer so that the data used for re-training can be further optimized.

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

5.1.6.2.2 Training data effectiveness analytics

A single/independent observation on whether a certain sample or feature at a given timestamp contributed to model gradients cannot provide understanding on whether using such a sample or feature will contribute to model accuracy of the same ML model in further training/re-trainings, or if it will contribute to efficiency of training of further models related to the same use case.

In order to have such understanding, further analysis of the data related to the importance of the data instances during training is needed. The patterns of the most effective training data generated from the analytics would be very helpful to improve data collection in order to optimize the quantity and quality of the data to be used for training.

5.1.6.2.3 Measurement data correlation analytics for ML training

For ML model training, a large amount of measurement data points may be collected and does not necessarily add value, e.g. due to the complexity and time-varying nature of network when the data is collected. Based on the fact that the collected measurement data can be highly correlated (linear or non-linear), using all measurement data for model training (and inference) can be a waste of computing resources and some means are therefore necessary to optimise the data based on the correlation. Hence there is a need to correlate the measurement data for ML training, such as:

- For a given task (e.g. analytics, model training), automatically analyses the correlation among the given set of all measurement data, the output can be a much smaller set of measurement data, with which ML model training could be much more efficient with limited (or managed) impact to model training performance (compared to using full set of data).

- Regularly renew the correlation analytics as time progresses, since the correlation relationship might change; this is especially useful when there is a need to regularly re-train the ML model re-training.

5.1.6.3 Potential requirements

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.

REQ-TRAIN_EFF-02: The 3GPP management system should have the capability to report the effectiveness of data used in the training, including providing an indication or label of which data instance is useful or not useful and an indication of which data feature is useful or not useful, or is contributing negatively to the model training.

REQ-TRAIN_EFF-03: The 3GPP management system should have the capability to allow the authorized consumer to activate the ML training function /Entity to label the effectiveness of data used for training.

REQ-TRAIN_EFF-04: The 3GPP management system should have the capability to allow authorized consumer to request analytics for data used for model training with respect to effectiveness of such data during model training.

REQ-TRAIN_EFF-05: The 3GPP management system should have the capability to generate and to provide to consumer a pattern of highly effective data for training of either specific version of a model, all versions of a single model, or all models related to certain use case.

REQ-MEAS-DATA-1: the 3GPP management system should have a capability to enable an authorized MnS consumer (e.g. an MLT function) to request the analysis of the correlation of measurement data used for training an ML entity.

REQ-MEAS-DATA-2: The ML MnS producer should have the capability enabling an authorized AI/ML MnS consumer to configure scheduling of the analysis of the correlation of measurement data.

REQ-MEAS-DATA-3: The ML MnS producer should have the capability enabling an authorized AI/ML MnS consumer to manage the scheduled analysis of the correlation of measurement data, e.g. to suspend an ongoing scheduled measurement data correlation analytics (MDCA) activity, to resume a suspended scheduled MDCA activity, to cancel a scheduled MDCA activity, to configure the cycle of MDCA activity.

5.1.6.4 Possible solutions

5.1.6.4.1 Possible solution for training data effectiveness reporting

Introduce an information Element (e.g. instance of IOC or a dataType) for interaction between ML training MnS producer and consumer: TrainingDataEffectivenessReport - this information element may represent model training effectiveness information of the related training data:

1) Introduce a control information element (e.g. an indication flag) in the MLTrainingRequest IOC (see 3GPP TS 28.105 [4]) to enable consumer to request training data effectiveness report. When the training data effectiveness report is enabled, the training data effectiveness report may be included as part of MOI of MLTrainingReport.

2) The information Element (TrainingDataEffectivenessReport) may contain the following attributes:

   - An attribute to indicate the overall status of report regarding training data instance effectiveness request, it can be a list of enumerations:

     * SUCCESSFUL indicates the request is successful, and report is generated.

     * UNSUCCESSFUL indicates the request is failed.

     * PARTIAL_SUCCESSFUL indicates request is partially successful, and report is generated.

     * NOTSUPPORT indicates the training data effectiveness request is not supported.

   - An attribute to represent the overall effectiveness information.

   - An attribute to represent the type of effectiveness is modelled, it can be a list of enumerations:

     * BINARY indicates the training data instance is useful or not useful.

     * WEIGHT indicates the training data instance contribution measured by weights (e.g. 0 to 1).

- A "_linkToReportFiles" attribute may optionally be supported.

5.1.6.4.2 Possible solution for training data effectiveness analytics

The solution may use a capability (Effective Training Data Pattern Analytics) in ML training MnS producer:

1) An MnS consumer may request the MnS producer to identify a pattern of the data used for training which contribute the most to the training process (e.g. input data that results in significant gradient changes)

   - The consumer may specify in the request if the most effective training data may be derived for:

     * specific version of an ML model;

     * for all versions of an ML model; or

     * for all ML models related to a use case/problem.

2) The Effective Training Data Pattern Analytics capability may derive and provide the pattern of the data used for training which contribute the most to the training process, i.e. effectiveTrainingDataPattern. Such pattern may comprise the following information (or any combination of the following information):

   - Set of data features which when used simultaneously (in combination) as input to model training have the significant effectiveness on the training process. This can be expressed by the list of e.g. DN (distinguished names) of performance metric or KPI that is the most significant for model training.

   - The list of DN (distinguished names) of the network objects from which the most effective data features have been collected.

   - The description of area from which the most effective data features have been collected, this can be expressed by e.g. list of cells (E-UTRAN-CGI or NG-RAN CGI), list of tracking area (identified by TAC - Tracking Area Code).

   - The information on geographical location of the network objects from which the most effective data features have been collected (e.g. latitude and longitude) or the larger geographical area info specified by convex polygon. See 3GPP TS 28.622 [13].

   - The time window(s) in which the most effective data features have been collected.

   - The effectiveness information of data as per data source (e.g. producer provided, or consumer provided).

   The effectiveTrainingDataPattern is derived by learning the associations between the data instance importance during ML model training and the context (e.g. time or geo-location) in which the given data instance has been collected.

3) The output of the analytics may be represented by an information element (IOC or DataType), e.g. effectiveTrainingDataPattern, which may be part of the MLEntity <<datatype>>, see 3GPP TS 28.105 [4]. This information element may include the following attributes:

   - combination of the data features with the most effect on the ML training;

   - list of network objects from which the effective data has been collected;

   - description of geographical location or area from which the effective data has been collected;

   - time instance or period in which the effective data has been collected;

   - any combination of the attribute listed above (e.g. effective Network Objects and effective Time indicating network objects and time from which the effective data has been collected).

5.1.6.4.3 Possible solution for measurement data correlation analytics

The solution may enable the ML MnS producer to support (with or without scheduled) measurement data correlation analytics via configuration, request from consumer. The high-level description of the proposed solution is illustrated by figure 5.1.6.4.3-1.

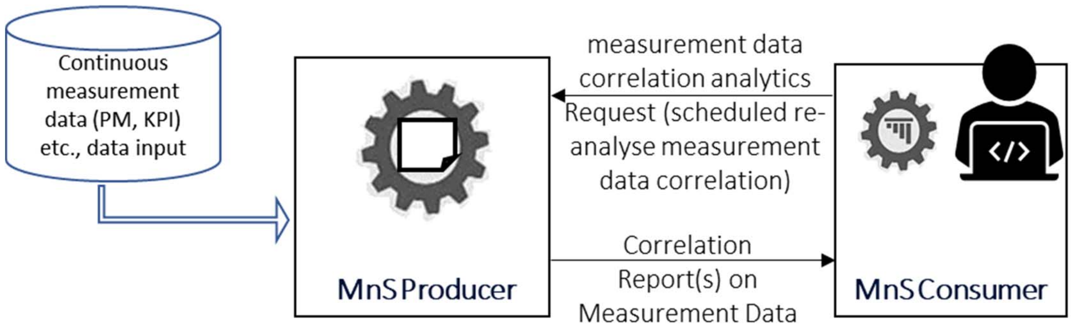

Continuous measurement data (PM, KPI) etc., data input
MnS Producer
measurement data correlation analytics Request (scheduled re-analyse measurement data correlation)
Correlation Report(s) on Measurement Data
MnS Consumer

Figure 5.1.6.4.3-1: High level description of the solution for measurement data correlation analytics

The steps in the figure are explained below:

1) MnS consumer may request the producer to initiate the measurement data correlation analytics. The request includes an indication may be an attribute (e.g. an information element, it may be named as MDCA-MeasurementDataCorrelationAnalytics, which may include necessary configuration for measurement data correlation analytics).

2) MnS producer upon receiving the request with configuration for measurement data correlation analytics, instantiates the MOI.

3) The instantiated MOI takes care of the input measurement data correlation analytics as part of request handling. E.g. prepare the pipeline of data input, cleansing, measurement data correlation analytics, and report the results with correlation results, etc.

4) If the MnS may support regular renew the measurement data analytics, the MnS consumer may:

    - Suspend the MDCA by update the configuration of scheduling information in MnS Producer.

    - Cancel the scheduled MDCA by stopping ongoing correlation analytics activity and delete related configuration in MnS Producer.

    - Update the scheduled MDCA cycle by re-configure the ongoing scheduled correlation analytics activity in MnS Producer.

The configuration information Element (e.g. MeasurementDataCorrelationAnalytics) may contain the following attributes:

- An attribute may indicate the address(es) of the candidate correlated measurement data generated from MDCA activity.

- An attribute may indicate the MDCA results, it may be SUCCESSFUL WITH MDCA GENERATED, FAILED DUE TO PERFORMANCE IMPACT, or other failure results.

- An attribute may indicate the MDCA performance requirement. It can be a percentage which requires the performance impact of trained MLEntity with generated measurement data within the range of the performance trained with full measurement data. E.g. 5 % means the model performance for the MLEntity trained with generated measurement data shall be no worse than 5 % of the performance trained with full measurement data.

- An attribute may indicate the actual MDCA performance impact. It can be a percentage which indicate the loss the model performance from trained MLEntity with generated measurement data comparison to the performance trained with full measurement data.

- If MnS support producer initiated regular MDCA, a scheduled MDCA activity may be enabled, a scheduling attribute may include the following attributes:

    - An attribute may indicate how frequent the MDCA activity shall be performed, e.g. weekly, or monthly.

    - Optionally MnS consumer may indicate when to start the scheduled MDCA.

    - An attribute may indicate the status of current scheduledMDCA as: RUNNING, SUSPENDED.

- The scheduling Flag may indicate if a scheduled MDCA is enabled or not.

To be noted, the solution may be implemented as part of MDA.

5.1.6.5 Evaluation

The solution described in clause 5.1.6.4.1 adopts the NRM-based approach, proposing two new information elements (class or dataType) with clear relationship to existing information element "MLTrainingRequest" and " MLTrainingReport ". It fully reuses the existing provisioning MnS Operations and notifications for control of Training Data Effectiveness reporting. The implementation of this NRM-based solution is straightforward. Therefore, the solution described in clause 5.1.6.4.1 is a feasible solution for training data effectiveness reporting.

The solution described in clause 5.1.6.4.2 is consistent with the ML training procedures and enhances the existing information element `MLEntity` with list of attributes. It is a fully NRM-based approach and reuses the existing provisioning MnS operations and notifications for Effective Training Data Pattern configuration and monitoring. It introduces the effectiveTrainingDataPattern information class to enable a versatile solution for training data effectiveness pattern. It provides the means to facilitate both capturing the information on the context of the `MLEntity`, as well enabling the notifications on the context change using the consistent NRM-based approach. Therefore, the solution described in clause 5.1.6.4.2 is a feasible solution for training data effectiveness analytics.

The solution described in clause 5.1.6.4.3 adopts the NRM-based approach, proposing new information elements (class or dataType) with clear relationship to existing information element like "MLTrainingRequest", "MDAReqeust" and " MLTrainingReport". It reuses the existing provisioning MnS Operations and notifications for control of measurement data correlation analytics and the implementation of this NRM-based solution is straightforward. Therefore, the solution described in clause 5.1.6.4.3 is a feasible solution to be developed further in the normative specifications.

5.1.7 ML context

5.1.7.1 Description

MLContext attribute (see 3GPP TS 28.105 [4]) represents the status and conditions related to the ML entity (see 3GPP TS 28.105 [4]). This may include the network context as defined in 3GPP TS 28.104 [2] as well as other conditions that may be applicable to the ML entity but are not part of network characteristics e.g. the time of day, season of the year. As part of ML model performance management there is the identification of the problem that the ML model is meant to address or deal with. As described in 3GPP TS 28.104[2], the differences in the network context, i.e. network status, under which data is collected to produce analytics, significantly affect the produced analytics. Similarly, the changes in the ML context, e.g. the characteristics of the data related to the network status and conditions used for ML model training, testing and deployment may affect the ML entity performance, thus may represent a problem for the ML entity. Thus, management capabilities are needed to enable awareness of the ML context in terms of the identification as well as monitoring and reporting of changes in ML context as part of the identification of the problem that the ML entity is meant to address or deal with.

5.1.7.2 Use cases

5.1.7.2.1 ML context monitoring and reporting

ML context related to ML model training, testing and deployment needs to be identified by characterizing the input data, used by the ML model, is targeted to work. As an example, such characterization may be done based on the statistical properties of data. Monitoring of such ML context serves to detect the changes and anomalies in the ML context. Some anomalies may be considered as a problem that ML entity is facing as it may lead to its performance degradation. Therefore, the consumer of the related AI/ML service needs to be informed about such observed ML context change.

5.1.7.2.2 Mobility of ML Context

In several network automation use cases, the respective AI/ML inference function cannot cover the complete network by employing single ML entity instance. An ML entity may be trained for a specific local context, and similarly, a different context may be applicable for inference, so the ML entity may be characterized by different *trainingContext* and an *expecetdInferenceContext*. However, the network scopes where the data used for training and inference is collected, does not always necessarily overlap with the network scopes in which the function makes decisions. The context of ML entities or AI/ML inference function may need to distinguish between context for generating decisions or insights, the context from which it generates measurements or data as well as the context in which it is prepared before being active for inference. So, the characteristics of the respective AI/ML inference function need to be distinguished depending on the different contexts of the AI/ML inference function. As such besides the validity scope defined by the *trainingContext* and an *expecetdInferenceContext*, the ML entity should also be characterized by specific *measurement scopes*, where the input measurements are collected. And these may also be separately defined for the 2 use cases.

5.1.7.2.3 Standby mode for ML entity

Where the respective AI/ML inference function cannot cover the complete network in one ML entity instance, multiple instances of ML entities may be required, one for each specific network scope, such as a cell. When a network automation use case requires several ML entities instances, where each has its own limited validity scope (a geographical area or a subnetwork), transfers of machine learning context, i.e. "handovers" between the ML entities covering different validity scopes (not necessarily identical to cell coverage area), are needed. Accordingly, the ML entities therein may have different roles, either as active or standby decision makers.

Consider the use case where a different ML entity instance is needed for each Base Station, i.e. the validity scope defined by the *expecetdInferenceContext* is a specific gNB. An instance of this is predictive ML-driven handover where an ML entity is trained to decide the optimal handover point and target cell, based on the UE measurements. Furthermore, the model inference is done in the UE. When the UE hands over to a cell in another gNB, which is in another validity scope, a new ML entity instance fitting the new validity scope needs to be deployed in the UE. This is illustrated by figure 5.1.7.2.3-1 a) where the UE uses ML entity instance 1 in both cells 1 & 2 but when the UE hands over to cell 3, which is outside the validity area of ML entity instance 2 needs to be deployed to the UE.

However, the deployment may require uploading the required ML entity instance into the UE and initializing the ML entity, for example to collect and feed the necessary input data to setup the required internal states, such as in Long-Short Term Memory (LSTM) Recurrent Neural Networks (RNNs). Accordingly, it may take significant time before the new ML entity becomes active and operational. Moreover, if the UE hands over back to cell 2 after a short stay in cell 3 (ping pong), the UE needs to immediately re-deploy ML entity instance 1, compounding the problem further.

To minimize this risk, there should be a "prepared scope" defined for each ML entity, which is the scope within which the ML entity is deployed and initialized but not activated for inference.

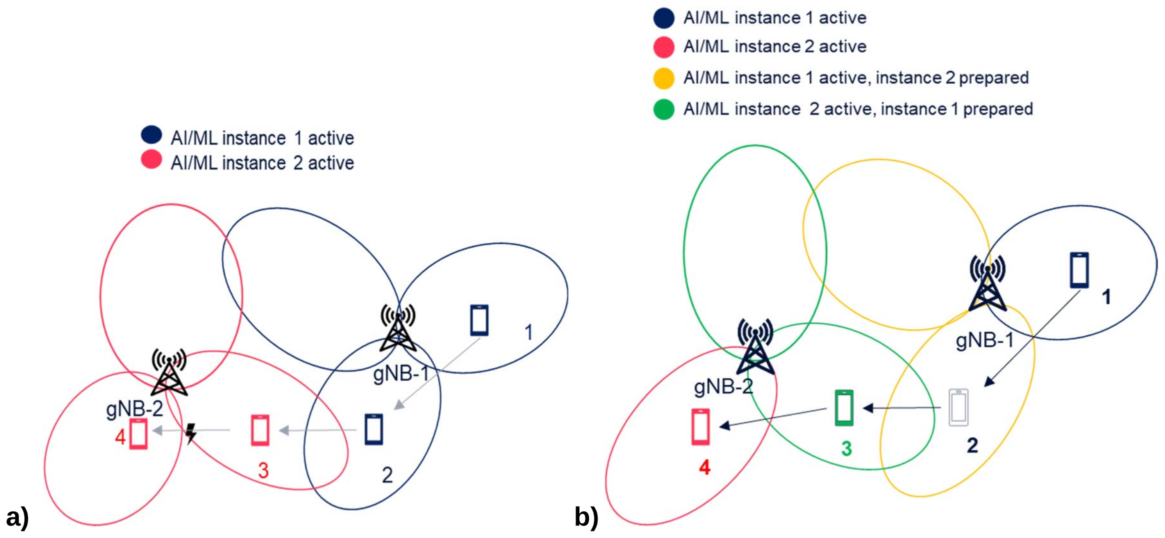


Figure 5.1.7.2.3-1: Example mobility of ML context a) validity scopes, b) validity and standby scopes

This is illustrated by figure 5.1.7.2.3-1 b) where besides the validity areas, standby areas are defined for each ML entity instance, e.g. ML entity 1 is active in cells 1 and 2 but standby in cell 3. This implies that the ML entity 1 should be availed to the UE in cell 3 even if the UE cannot use ML entity instance 1 in cell 3. To support this, it needs to be possible to configure both, the validity areas and the standby areas for ML entities and to define their role in them, i.e. either active, or prepared.

5.1.7.3 Potential requirements

REQ-ML_CTX -1: The MLT MnS producer should have a capability to identify and monitor the ML context, as well as to inform the MnS consumer about observed changes in ML context.

REQ-ML_CTX-2: The MLT MnS producer should have a capability for an authorized MnS consumer to configure or read the measurement scope of an ML entity for ML training.

REQ-ML_CTX-3: The 3GPP Management system should have a capability for an authorized MnS consumer to configure or read the validity scope of an ML entity for AI/ML inference.

REQ-ML_CTX-4: The MnS producer responsible for interface configuration should have a capability for an authorized MnS consumer to read the ML entity's inference prepared scope that defines the network scope within which the ML entity is prepared to be in standby mode in preparation for elevating to active mode.

5.1.7.4 Possible solutions

5.1.7.4.1 MLContext <<datatype>> on MLEntity

The IOC `MLContext is a <<datatype>>` attribute on the `MLEntity`. The `MLContext is` notifiable, so that any interested party can subscribe to a notification on the `MLContext.`

When there is a change in the `MLContext`, e.g. as observed from the statistical properties of data, the notification is sent to the entity that subscribed to the notification.

The `MLContext` has the following attributes which can be configured by the MnS consumer when defining an `MLContext` to be monitored:

- Attribute "area of interest" identifying a scope e.g. the geographical area to be taken into account.
- Attribute "area granularity" defining the size of the sub-areas of the area of interest for which the statistical properties of data should be identified. It can be expressed for example in km or as a description of a relevant part of the network (e.g. building, street, block, district, city, or state). In case area granularity attribute is not specified by the MnS consumer, contexts related to different areas are determined according to the data distribution detected in the area of interest.
- Attribute " reporting_threshold " indicating when the deviation in data statistics compared to previously determined context needs to be reported. It can be numeric attribute, e.g. indicating the percentage of changes between the currently monitored data statistics and previously identified data statistics.

The notification delivers the `MLContextReport` that contains the information on partitioning of area of interest into smaller areas (i.e. sub-areas) based on statistical properties of data. The report may also comprise the statistical properties of identified sub-areas. Furthermore, the report may include the information on detected changes in data statistics. Hereby, either the complete information on current data statistics or the actual "delta" compared to previous data statistics may be indicated to the MnS consumer.

The `MLContextReport` MOI is contained by the `MLTrainingFunction` or `MLInferenceFunction` MOI.

5.1.7.4.2 Mobility of `MLContext`

To support Mobility of MLContext, extend MLContext (3GPP TS 28.105 [4], clause 7.4.3) with additional parameters *monitoringScope*, *validityScope* and *preparedScope*. The *monitoringScope* is where the data used for training and inference is collected, the *validityScope* is the network scopes in which the function makes decisions while the *preparedScope* is the network scopes in which the function is prepared to be ready for inference.

Table 5.1.7.4.2-1: Extended attributes for MLContext

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
`monitoringScope`	?	T	T	?	?
`validityScope`	?	T	T	?	?
`preparedScope`	?	T	T	?	?

5.1.7.5 Evaluation

The solution described in clause 5.1.7.4.1 is consistent with the `MLEntity` <<IOC>> and enhances the existing information element `MLContext` with three attributes, which are to configure & monitor three types of contexts. It is a fully NRM-based approach and reuses the existing provisioning MnS operations and notifications for context configuration and monitoring. It introduces the `MLContextReport` information class to enable a versatile solution for deliveries of context notifications. It provides the means to facilitate both capturing the information on the context of the `MLEntity`, as well enabling the notifications on the context change using the consistent NRM-based approach. Therefore, the solution described in clause 5.1.7.4.1 is a feasible solution for ML context.

The solution described in clause 5.1.7.4.2 enhances the `MLContext` datatype with attributes that characterize the scope of the `MLEntity`. This enables the network or management functions to read the scope and determine the respective scope for which the ML entity supports. It also enables the consumers to configure the scopes differently even where the `MLEntity`changes contexts. Therefore, the solution described in clause 5.1.7.4.2 is a feasible solution for Mobility of `MLContext`.

5.1.8 ML entity capability discovery and mapping

5.1.8.1 Description

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

- a decision-making capability which is in the form of triple <x,y,z> indicating:
    - x: the object or object types for which the ML entity can undertake optimization or control
    - y: the configurable attributes on object or object types x, which the ML entity optimizes or controls to achieve the desired outcomes
    - z: the performance metrics which the ML entity optimizes through its actions
- an analysis capability which is in the form of tuple <x,z> indicating:
    - x: the object or object types for which the ML entity can undertake analysis
    - z: the network context (on object x) for which the ML entity produces analysis

Different network functions may need to rely on existing AI/ML capabilities to accomplish the desired automation. However, the applicability of the ML-based solutions and the details of such ML-based solutions (i.e. which ML entities are applied and how) for accomplishing those automation functionalities is not obvious. On a high-level, such ML-based solutions may be categorized into different cases, either with or without ML orchestration. In both cases, management services are required to identify the capabilities of the involved ML entities and to map those capabilities to the desired logic.

5.1.8.2 Use cases

5.1.8.2.1 Identifying capabilities of ML entities

Network functions, especially network automation functions, may need to rely on AI/ML capabilities that are not internal to those network functions to accomplish the desired automation. For example, as stated in 3GPP TS 28.104 [2], "an MDA Function may optionally be deployed as one or more AI/ML inference function(s) in which the relevant 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 fulfilment solution may need to employ the capabilities of existing AI/ML to fulfil the intents. In any such case, management services are required to identify the capabilities of those existing ML entities.

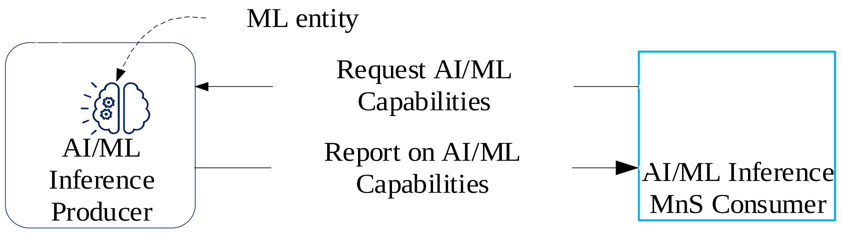

ML entity
AI/ML Inference Producer
Request AI/ML Capabilities
Report on AI/ML Capabilities
AI/ML Inference MnS Consumer

Figure 5.1.8.2.1-1: Request and reporting on AI/ML capabilities

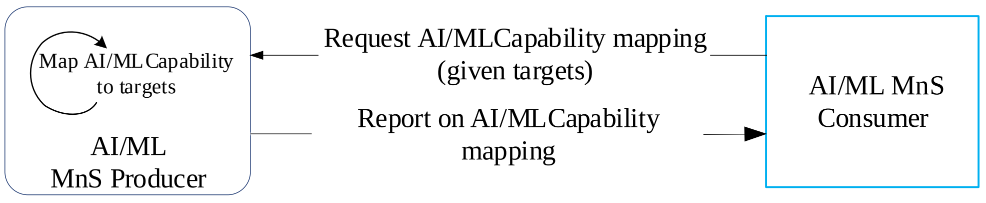
Figure 5.1.8.2.1-1 shows that the consumer may wish to obtain information about AI/ML capabilities to determine how to use them for the consumer's needs, e.g. for fulfilment of intent targets or other automation targets.

5.1.8.2.2 Mapping of the capabilities of ML entities

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

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

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


Map AI/ML Capability to targets
AI/ML MnS Producer
Request AI/MLCapability mapping (given targets)
Report on AI/MLCapability mapping
AI/ML MnS Consumer

NOTE: Figure 5.1.8.2.2-1 shows that the consumer may wish to obtain the mapping of AI/ML capabilities to some management tasks to determine how to use them for the consumer's needs, e.g. for its intent targets or other automation targets. The management tasks may for example include specific metrics to be optimized, but the candidate tasks to be considered are to be agreed at the normative phase.

Figure 5.1.8.2.2-1: Mapping execution logic to AI/ML Capabilities

5.1.8.3 Potential requirements

REQ-ML_CAP-1: The 3GPP Management system should have a capability for an authorized MnS consumer to request the AI/ML MnS Producer for the capabilities of existing ML entities available within the producer of AI/ML inference.

REQ-ML_CAP-2: The AI/ML MnS Producer should have a capability to report to an authorized MnS consumer the capabilities of an ML entity as a decision described as a triplet <object(s), parameters, metrics> with the entries respectively indicating: the object or object types for which the ML entity can undertake optimization or control; the configuration parameters on the stated object or object types, which the ML entity optimizes or controls to achieve the desired outcomes; and the network metrics which the ML entity optimizes through its actions.

REQ-ML_CAP-3: The AI/ML MnS Producer should have a capability to report to an authorized MnS consumer the capabilities of an ML entity as an analysis described as a tuple <object(s), characteristics> with the entries respectively indicating: the object or object types for which the ML entity can undertake analysis; and the network characteristics (related to the stated object or object types) for which the ML entity produces analysis.

REQ-ML_CAP-4: The 3GPP Management system should have a capability to enable an authorized MnS consumer to request an AI/ML MnS Producer for a mapping of the consumer's targets to the capabilities of one or more ML entities.

5.1.8.4 Possible solutions

The network functions may rely on available AI/ML capabilities to achieve the desired outcome. Such available AI/ML capabilities may need to be discovered as a first step.

The following solution (related to the workflow depicted in figure 5.1.8.2.2-1**)** may be applicable:

- when the AI/ML MnS Consumer requests for information on available ML entities and their supported AI/ML capabilities from the AI/ML MnS Producer (e.g. inference producer), the AI/ML MnS Producer provides the AIML_capability in the following form:
  - a decision-making capability in the form of triple <x,y,z> indicating:
    * x: the object or object types for which the ML entity can undertake optimization or control.
    * y: the configurable attributes on object or object types x, which the ML entity optimizes or controls to achieve the desired outcomes.
    * z: the performance metrics which the ML entity optimizes through its actions.
  - an analysis capability in the form of tuple <x,z> indicating:
    * x: the object or object types for which the ML entity can undertake analysis.
    * z: the network context (on object x) for which the ML entity produces analysis.
- introduce the <<datatype>> attribute representing the AI/ML capability e.g. named as `AIML_capability` as The attribute is a property of any AI/ML MnS Producer or any function that has or contains ML entity e.g. for any AI/ML inference function, ML testing function or ML training function. The AIML_capability may also be added to the ML entity; and
- the attribute for the AI/ML capability will as such have three attributes:
  - An attribute for the managed object: This is conditionally mandatory as either the object or the object type should be stated. It is mandatory if the managed object type is not included.
  - An attribute for the managed object type: This is also conditionally mandatory as either the object or the object type should be stated. It is mandatory if the managed object is not included.
  - An attribute for the configurable attributes on the managed object or managed object types: This is optional as it only applies for decisions and not for analysis type capabilities.
  - An attribute for the metrics: This which includes either the performance for the decision or the analyses metrics or context should be mandatory.

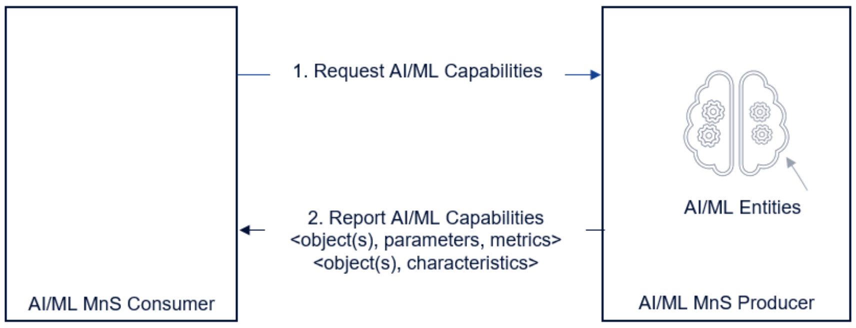

1. Request AI/ML Capabilities
AI/ML Entities
2. Report AI/ML Capabilities
<object(s), parameters, metrics>
<object(s), characteristics>
AI/ML MnS Consumer
AI/ML MnS Producer

Figure 5.1.8.4-1: AI/ML capability request and report

5.1.8.5 Evaluation

The solution described in clause 5.1.8.4 adopts the NRM-based approach, which reuses the existing provisioning MnS operations and notifications. Moreover, the solution enables reuse of the Discovery MnS to discover both the AI/ML functionality and their capabilities. Introducing the AI/ML_capability <<datatype>> enables a working solution with or without the discovery MnS. Via the discovery MnS this AI/ML_capability will be the returned outcome and without the discovery MnS, the consumer can instead read this AI/ML_capability attribute off the MOI.

Therefore, the solution described in clause 5.1.8.4 is a feasible solution.

5.1.9 AI/ML update management

5.1.9.1 Description

Due to the complexity and time-varying nature of network, the ML entities previously deployed may no longer be applicable to the current network status after running for a period of time. Typically, the performance of a trained model may degrade over time (this is referred to as model drift [5]).

Therefore, the ML entity/entities need to be updated in a timely manner to ensure an up to date optimum inference performance in the network or system.

5.1.9.2 Use cases

5.1.9.2.1 ML entities updating initiated by producer

The ML entity updating may be initiated by the AI/ML MnS producer. In order to keep the model at a requested level, the AI/ML MnS producer may periodically conduct ML entity retraining with new available training data. Once a new version ML entity is obtained after the training is finished, it can be used to update the current ML entity with this new version. In another condition, the AI/ML MnS producer may initiate ML entity updating based on the running model performance. For example, if the performance of the running ML model is decreased under a predefined threshold, the AI/ML MnS producer may decide to start re-training and then update the ML entity to a new version which performs better.

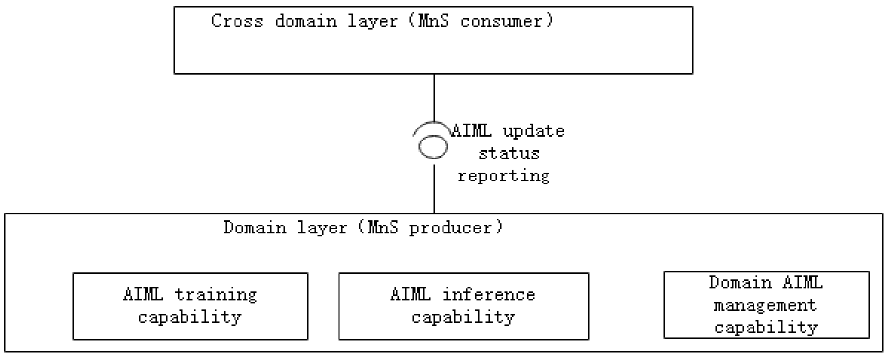

Cross domain layer (MnS consumer)
AIML update status reporting
Domain layer (MnS producer)
AIML training capability
AIML inference capability
Domain AIML management capability

Figure 5.1.9.2.1-1: ML entities update initiated by producer

5.1.9.3 Potential requirements

REQ-AIML_UPD-CON-1: The AI/ML MnS producer should have a capability to update the ML entities and inform an authorized consumer about the update status.

5.1.9.4 Possible solutions

Following is the proposed solution based on information model defined in 3GPP TS 28.105 [4]:

- Extend the `MLEntity <<dataType>>` with an attribute "updatedTime", which indicates the time that the ML entity is updated. For a trained ML entity, the value of "mLEntityVersion" indicates the version number of the ML entity. When the ML entities updating is initiated and successfully executed, the value of "mLEntityVersion" is modified to be the new version number, and the value of "updatedTime" is the time that the updating is finished. Then AI/ML MnS producer can use the "notifyMOIAttributeValueChange" operation to inform the authorized MnS consumer about the ML update Status including the updated `MLEntity` version number and the corresponding updating time.

5.1.9.5 Evaluation

The solution described in clause 5.1.9.4 enhances the existing `MLEntity <<dataType>>` with a new attribute "updatedTime" to indicate the update time of ML entity. Then, the "notifyMOIAttributeValueChange" operation can be reused to inform the update information of ML entities to consumers. The existing attributes and operations are reused to a larger extent, thus the proposed solution can be easily realized.

Therefore, the solution described in clause 5.1.9.4 is a feasible solution.

5.1.10 Performance evaluation for ML training

5.1.10.1 Description

In ML model training phase (including training, validation, and testing), the performance of ML model needs to be evaluated. The related performance indicators need to be collected and analysed.

5.1.10.2 Use cases

5.1.10.2.1 Performance indicator selection for ML model training

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

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

- Model training resource performance indicators: the performance indicators of the system that the model trains. e.g. "training duration" etc.

- Model performance indicators: performance indicators of the model itself, e.g. "accuracy", "precision", "F1 score", etc.

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

5.1.10.2.2 Monitoring and control of AI/ML behavior

In a typical network operation, an operator configures and operates an ML entity according to the corresponding manual of the entity. Usually, the operator does not need to know the details of the ML entity's internal-decision making process and implementations, simply due to "too many" ML entities running for inference in the network and also due to too much details and information that are deemed to be redundant or unnecessary for the operator, plus it is in the vendor's interest not to disclose any internal aspects of the implementation of their automation solutions.

However, the operator may still need to guide and evaluate the solutions and to configure/reconfigure them to achieve the desired outcomes in an ML entity-agnostic manner. For example, consider the load balancing automation use case (AutoLB) summarized by table 5.1.10.2.2-1. An AutoLB ML entity helps to decide how to distribute load among network objects. The ML entity has specific actions that it can take while the operator also has operational actions that it needs to take to customize or steer the solution, e.g. to switch off the solution, to reconfigure the solution, to change the solutions input. As such, the behavior of the ML entity in terms of the actions it takes under any given conditions needs to be related to the configuration actions from the MnS consumer (e.g. the operator).

Table 5.1.10.2.2-1: Operability of the Automated load balancing

Automation use case	Description	ML entity's context	Example ML entity's decisions	Operator's Actions
load balancing (AutoLB)	Distribute load among different network objects, e.g. among cells	- Amount of traffic, No. of users, etc.	- Select CIO values	- Set the maximum CIO values - Deactivate AutoLB

So, it is important that even without knowing the details of the ML entity, the operator needs to have understanding of the ML entity's overall behaviour. And, if a part of the ML entity's decisions/actions are not what is preferred by the operator, the operator needs to customise the ML entity in order to produce the expected or optimum solutions. Additionally, some measures or means are needed to enable the operators to associate their actions to the context and to enable the operator to provide information regarding the expected behaviour of the AI/ML capabilities to facilitate the AI/ML solution. This could result in a re-design or re-training of the ML entity, according to the workflow pictured in figure 4.3.1-1.

5.1.10.2.3 ML entity performance indicators query and selection for ML training/testing

The ML entity performance evaluation and management is needed during training and testing phases. The related performance indicators need to be collected and analysed. The MnS producer of ML training/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 AI/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. MnS producer for ML training/testing can be queried to provide the information on supported performance indicators referring to ML entity training/testing phase. Such performance indicators in training phase may be for example accuracy/precision/recall/F1-score/MSE/MAE /confusion matrix, and in test phase may be data drift in data statistics. Based on supported performance indicators in different phases as well as based on consumer's requirements, the MnS consumer for ML training or ML testing may request a sub-set of supported performance indicators to be monitored and used for performance evaluation. Management capabilities are needed to enable the MnS consumer for ML training or ML testing to query the supported performance indicators and select a sub-set of performance indicators in training or testing phase to be used for performance evaluation.

5.1.10.2.4 ML entity performance indicators selection based on MnS consumer policy for ML training/testing

ML entity performance evaluation and management is needed during ML training and testing phases. The related performance indicators need to be collected and analysed. The MnS producer for ML training or testing 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 AI/MnS consumer for ML training or testing may have differentiated levels of interest in the different performance dimensions or metrics. Thus, depending on its use case, the AI/ML MnS consumer may indicate the preferred behaviour and performance requirement that needs to be considered during training and testing of/from the ML entity by the ML MnS producer for ML training or testing. These performance requirements need not indicate the technical performance indicators used for ML training, testing or inference, such as "accuracy" or "precision" or "recall" or "Mean Squared Error", etc. The ML AI/MnS consumer for ML training or testing may not be capable enough to indicate the performance metrics to be used for training and testing. Instead, the AI/ML MnS consumer may indicate the requirement using a policy or guidance that reflects the preferred performance characteristics of the ML entity. Based on the indicated policy/guidance, the ML MnS producer for ML training or testing may then deduce and apply the appropriate performance indicators for training or testing. Management capabilities are needed to enable the ML MnS consumer for ML training or testing to indicate the behavioural and performance policy/guidance that may be transformed by the MnS producer to a technical performance indicator during training or testing.

5.1.10.3 Potential requirements

REQ-MODEL_PERF-TRAIN-1: The MLT MnS producer should have a capability to allow an authorized consumer to get the capabilities about what kind of ML models the training function is able to train.

REQ-MODEL_PERF-TRAIN-2: The MLT MnS producer should have a capability to allow an authorized consumer to query what performance indicators are supported by the ML training function for each kind of ML model.

REQ-MODEL_PERF-TRAIN-3: The MLT 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 kind of ML model.

REQ-MODEL_PERF-TRAIN-4: The MLT 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.

REQ-AI/ML_BEH-1: The 3GPP management system should have a capability to inform an authorized AI/ML MnS consumer (e.g. the operator) about the behavior of the ML entity, in an ML entity agnostic manner without the need to expose its internal characteristics.

REQ-AI/ML_BEH-2: The 3GPP management system should have a capability that enables an authorized AI/ML MnS consumer (e.g. the operator) to configure the behavior of the ML entity, in an ML entity agnostic manner that does need to expose its internal characteristics.

REQ-AI/ML_PERF -SEL-1: The MLT MnS producer should have a capability allowing the authorized MnS consumer to discover supported AI/ML performance indicators related to ML training and testing and select some the desired indicators based on the MnS consumer's requirements.

REQ-AI/ML_PERF-POL-1: The AI/ML MnS producer should have a capability allowing the authorized MnS consumer to indicate a performance policy related to ML model training and testing phases.

5.1.10.4 Possible solutions

5.1.10.4.1 Possible solutions for performance indicator selection for ML model training

This solution uses the instances of following IOCs or attribute for interaction between MnS producer and consumer to support the performance indicator selection for ML model training:

1) The IOC or attribute representing the ML training capability, for example named as `MLTrainingCapability`, contained by `MLTrainingFunction` (see 3GPP TS 28.105 [4]).

   This IOC or attribute is created by the MnS producer and contains the following attributes:

- inference type of the ML model that the ML training function supports to train;
- supported performance metrics (see `performanceMetric` in 3GPP TS 28.105 [4]).

2) The IOC `MLTrainingRequest` (see 3GPP TS 28.105 [4]) with the existing `performanceRequirements` attribute. The `performanceMetric` element of the `ModelPerformance` data type for the `performanceRequirements` attribute is semantically extended to indicate the MnS consumer selected performance indicator/metric.

NOTE: The name of the IOCs and attributes are to be decided in normative phase.

5.1.10.4.2 Possible solutions for monitoring and control of AI/ML behavior

To allow for monitoring and control of AI/ML behavior:

- The contexts and actions of the AI/ML MnS producer are grouped into operational modes represented by abstract states that are understood by both the AI/ML MnS producer and the AI/ML MnS Consumer:
  * For example, the Robocar may be considered to have a few (e.g. two) abstract states:
    - Normal operations, where the Robocar may be simply given a destination and let to act as it wishes.
    - Extraneous circumstances, which represents unusual conditions such an accident on the road (as learned from the radio), abnormal street conditions such an unusually wet street due to pipe splashing water onto the street or a street power line bent into the road. In such cases the operator actions may be different, e.g. to ask the car to make a sudden stop or sudden turn.
  * The expected number of abstract states depends on use case but is in general a small number. So, the maximum number of abstract states may be set to a small value but large enough to support most use cases (e.g. a set of states numbered 0-16 or 0-63).
- Each ML entity or AI/ML inference function should have an object say called abstract behavior that contains characteristics of the abstract behavior of the ML entity or AI/ML inference function. The abstract behavior may be an IOC names say, `abstractBehavior` and name contained on the ML entity or AI/ML inference function. The abstract behavior contains 2 attributes, the candidate abstract states and the applied abstract states.
- A list of candidate abstract states and their candidate actions and a list of the selected and configured abstract states and their respective selected actions.
- Introduce a datatype for the candidate abstract state, say called `candidateAbstractState`:
  * Introduce `candidateAbstractStates` as an attribute of the abstract behavior. The `candidateAbstractState` is a list of abstract states and where each state has a list of candidate abstract actions for that abstract state.
  * Each `candidateAbstractState` may have a string identifier of the abstract state, a human readable description and a list of possible actions that may be selected for that state. As such there should be an attribute for possible actions, say called `possibleActions` that holds the possible actions for that state. The `possibleActions` attribute may be an enumeration of the actions from which the MnS consumer may chose those to be applied.
- Introduce a datatype for the applied abstract states, say called `appliedAbstractStates`:
  * The `appliedAbstractStates` is a list of state-action tuples. Each state may be represented by an identifier for the respective state as listed in the `candidateAbstractStates`. Similarly, each action may be represented by an identifier for the respective action as listed in the `possibleActions` of the respective `candidateAbstractState`.

5.1.10.4.3 Possible solutions for ML entity performance indicators query and selection

This solution extends the `ModelPerformance` data type to specify which ML performance indicators can be supported by ML entity or its hosting function (e.g. MLTrainingFunction or MLInferenceFunction). The same data type can be used to activate the notification on specific ML performance indicators based on the request by the authorized MnS consumer.

SupportedMlPerformance <<dataType>>

This data type specifies the performance indicator which can be supported by an ML entity or a function (e.g. MLTrainingFunction or MLInferenceFunction). It contains the tuples of `supportedPerformanceMetric` and `activatedPerformanceMetric` attributes. The `supportedPerformanceMetric` indicates performance metric which ML entity or a function is capable of providing e.g. accuracy/precision/recall/F1-score/MSE/MAE. The authorized MnS consumer should be notified only on a specific subset of such performance metrics for which the `activatedPerformanceMetric` indicator is set.

Attributes

Attribute name	Support Qualifier	isReadable	isWritable	isInvariant	isNotifyable
`supportedPerformanceMetric`	M	T	F	F	T
`activatedPerformanceMetric`	M	T	F	F	T

Attribute definitions

Attribute Name	Documentation and Allowed Values	Properties
`SupportedPerformanceMetric`	It indicates the performance metric which ML entity or a function is capable of providing e.g. "accuracy", "precision", "F1 score", etc. allowedValues: N/A.	Type: String multiplicity: 1 isOrdered: N/A isUnique: True defaultValue: None isNullable: False
`ActivatedPerformanceMetric`	It indicates whether the ML MnS consumer activated the notifications on specific performance metric. Setting this attribute to "TRUE" the `SupportedPerformanceMetric` will be notified to the consumer.	Type: Boolean multiplicity: 0..1 isOrdered: N/A isUnique: N/A defaultValue: FALSE isNullable: False

5.1.10.4.4 Possible solutions for policy-based performance indicator selection

Following is the proposed solution based on information model defined in 3GPP TS 28.105 [4]:

- Existing ModelPerformance <<datatype>> as part of MLTrainingRequest IOC may be extended optionally with attribute that represents the behavioural requirements as a policy. This attribute may be named as "trainingPolicyIndicator".

- This attribute may contain information on the magnitude of the sensitive inference as a triplet. Some examples may look like following.

EXAMPLE 1: False Positives, less, 10:

- The above example indicates the training to be performed such that the probability of false positives is less than 10 % in case of classification.

EXAMPLE 2: Over-Prediction, high, 80

- The above example indicates the training to be performed such that the probability of over-prediction is greater than 80 % in case of regression.

This information along with the existing "performanceScore" and "performanceMetric" may help the ML Training Producer to train the ML entity efficiently for the specific use case. This attribute may be applicable when configured in MLTrainingRequest IOC and not applicable in MLTrainingReport IOC.

A similar solution can be applied to the testing and inference functions.

5.1.10.5 Evaluation

The solution described in clause 5.1.10.4.1 uses the NRM based solution for the consumer to query the supported ML training capabilities and reuses the existing IOC and attributes in maximum extend. The new and existing NRMs can be easily and clearly correlated in this solution. Therefore, the solution described in clause 5.1.10.4.1 is a feasible solution.

The solution described in clause 5.1.10.4.2 enhances the existing information model for the `MLEntity` with 1 IOC that contains two attributes. These allow to hold the candidate abstract behavior for the ML entity and the applied, e.g. as set by the AI/ML MnS consumer. These information elements should support management and control of the abstract behavior of the ML entity or the related AI/ML inference function. Therefore, the solution described in clause 5.1.10.4.2 is a feasible solution for monitoring and control of AI/ML behavior.

The solution described in clause 5.1.10.4.3 is NRM-based approach and reuses the existing provisioning MnS operations. It is consistent with the ML entity definitions and enhances its existing attributes. It provides the means to facilitate both capturing the information on the supported performance indicators in different ML phases as well as selecting the performance indicators to be provided using the consistent NRM-based approach.

5.1.11 Configuration management for ML training phase

5.1.11.1 Description

As defined in 3GPP TS 28.105 [4], ML training can be initiated by MnS consumer or MnS producer.

The ML training function may be located in the management system or in the NF (e.g. gNB or NWDAF). When ML training is performed, it takes a significant amount of resources. Therefore, the producer-initiated ML training needs to be controlled, especially when the training function is co-located with other functions (e.g. inference function).

5.1.11.2 Use cases

5.1.11.2.1 Control of producer-initiated ML training

For producer-initiated ML training, the MnS producer has its own algorithm to trigger and perform the ML training.

However, the MnS consumer may expect the training to be performed under certain conditions, for example when the inference performance of the existing ML entity running in the inference function does not meet the target, or the network environment is changed. So the consumer may provide the policy containing the conditions (e.g. inference performance metrics & threshold, network conditions) for the MnS producer to trigger the ML training.

The MnS consumer may also want to avoid the ML training during busy traffic time (especially when the ML training function is located in the NF) and only allow the ML training to occur within a pre-configured time window.

The consumer may even choose to deactivate the ML training, if the training performance consistently cannot meet the performance requirements.

Therefore, the MnS consumer needs to be able to control the producer-initiated ML training with the configurations.

5.1.11.3 Potential requirements

REQ-MLTRAIN_CFG-1: The ML training MnS producer should have a capability to allow the authorized MnS consumer to configure activation, deactivation and retraining related policies.

REQ-MLTRAIN_CFG-1: The ML training MnS producer should have a capability to allow the authorized MnS consumer to activate and deactivate the ML training function.

REQ-MLTRAIN_ACT-1: The ML training MnS producer should have a capability to inform an authorized MnS consumer about the activation and deactivation of the ML training function.

5.1.11.4 Possible solutions

5.1.11.4.1 ML training policy configuration

A data type or abstract class describing the policy (e.g. condition) for controlling the ML training function, and this data type or abstract class can be used or inherited by the MOI representing the ML training function (i.e. `MLTrainingFunction` defined in 3GPP TS 28.105 [4]).

The policy contains the conditions (e.g. thresholds of the performance measurements indicating the inference performance, network conditions (e.g. number of active UEs with certain capabilities, changes of neighbour cells), etc.) for triggering the ML training.

The ML Training MnS producer monitors the conditions and triggers the ML training according to the configured policy.

5.1.11.4.2 ML training activation and deactivation

5.1.11.4.2.1 General framework for activation and deactivation

This subclause describes the general framework for activation and deactivation of ML training function.

A data type or abstract class describing the activation properties, and this data type or abstract class can be used or inherited by the MOI representing the ML training function (i.e. `MLTrainingFunction` defined in 3GPP TS 28.105 [4]).

This general framework supports the general properties for all types of activation/deactivation, including:

- Activation type: which can be instant activation/deactivation, scheduled activation/deactivation. And this data type or abstract class is extended with the attributes supporting these specific types of activation.

5.1.11.4.2.2 Instant activation and deactivation

The generic framework described in clause 5.1.11.4.2.1 is extended with the following attributes to support instant activation and deactivation:

- Switch for "activated" and "deactivated" status.

5.1.11.4.2.3 Schedule based activation and deactivation

The generic framework described in clause 5.1.11.4.2.1 is extended with the following attributes to support the schedule-based activation and deactivation:

- The schedule for activation/deactivation.

5.1.11.5 Evaluation

The solutions described in clause 5.1.11.4 is a fully NRM-based approach and reuses the existing provisioning MnS operations for ML training configuration. This solution extends the existing IOC representing the ML training function (i.e. `MLTrainingFunction` defined in 3GPP TS 28.105 [4]) with attributes defined by data type or abstract class for controlling the ML training, thus the changes are minimal on the existing NRMs.

Therefore, the solution described in clause 5.1.11.4 is a feasible solution.

5.1.12 ML Knowledge Transfer Learning

5.1.12.1 Description

It is known that existing ML capability can be leveraged in producing or improving new or other ML capability. Specifically, using transfer learning knowledge contained in one or more ML entities may be transferred to another ML entity. Transfer learning relies on task and domain similarity to deduce whether some parts of a deployed ML entity can be reused in another domain / task with some modifications. As such, aspects of transfer learning that are appropriate in multi-vendor environments need to be supported in network management systems. However, ML entities are likely to not be multi-vendor objects, i.e. it will in most cases not be possible to transfer an ML entity from function to another. Instead, the knowledge contained in the model should be transferred instead of transferring the ML entity itself. For example, the knowledge contained in an ML entity deployed to perform mobility optimization by day can be leveraged to produce a new ML entity to perform mobility optimization by night. As such and as illustrated by figure 5.1.12.1-1, the network or its management system needs to have the required management services for ML Transfer Learning (MLKLT), where ML Transfer Learning refers to means to allow and support the usage and fulfilment of transfer learning between any two ML entities.

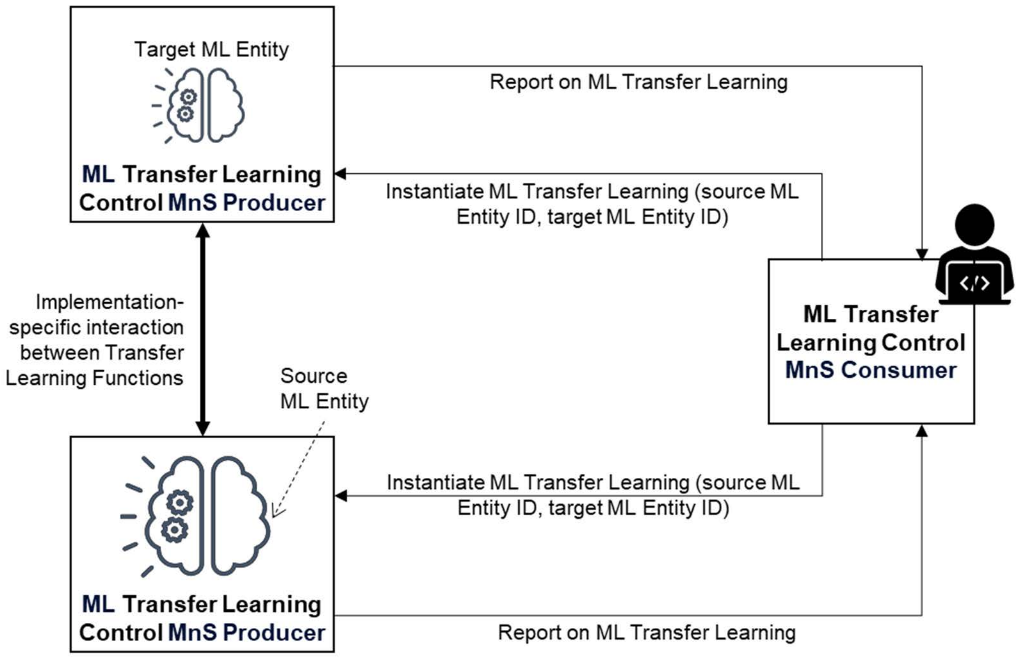

Target ML Entity
ML Transfer Learning Control MnS Producer
Report on ML Transfer Learning
Instantiate ML Transfer Learning (source ML Entity ID, target ML Entity ID)
Implementation-specific interaction between Transfer Learning Functions
Source ML Entity
ML Transfer Learning Control MnS Consumer
Instantiate ML Transfer Learning (source ML Entity ID, target ML Entity ID)
ML Transfer Learning Control MnS Producer
Report on ML Transfer Learning

Figure 5.1.12.1-1: ML Knowledge Transfer Learning (MLKLT) flow between the source MLKLT (which is the entity with the pre-trained ML entity), the peer MLKLT (which is the entity that shall train a new ML entity) and the MLKLT MnS consumer (which may be the operator or another management function that wishes to trigger or control MLKLT)

5.1.12.2 Use cases

5.1.12.2.1 Discovering sharable Knowledge

For the transfer learning, it is expected that the source ML Knowledge Transfer Learning MnS producer shares its knowledge with the target ML Training function, either simply as single knowledge transfer instance or through an interactive transfer learning process. The concept of knowledge here represents any experiences or information gathered by the MLEntity in the ML Knowledge Transfer Learning MnS producer through training, inference, updates, or testing. This information or experiences can be in the form of - but not limited to - data statistics or other features of the underlying ML model. It may also be the output of an MLEntity. The 3GPP management systems should provide means for an MnS consumer to discover this potentially shareable knowledge as well as means for the provider of MLKLT to share the knowledge with the MnS consumer.

5.1.12.2.2 Knowledge sharing and transfer learning

The transfer learning may be triggered by a MnS consumer either to fulfil the learning for itself or for it to be accomplished through another ML Training function. The entity containing the knowledge may be an independent managed entity (the ML entity). Alternatively, the ML model may also be an entity that is not independently managed but is an attribute of a managed ML entity or ML function in which case MLKLT does not involve sharing the ML model or parts thereof but may imply implementing the means and services to enable the sharing of knowledge contained within the ML entity or ML-enabled function. The 3GPP management system should provide means and the related services needed to realize the ML transfer learning process.

Specifically, the 3GPP management system should provide means for an MnS consumer to request and receive sharable knowledge as well as means for the provider of MLKLT to share the knowledge with the MnS consumer or any stated target ML Training function. Similarly, the 3GPP management system should provide means for an MnS consumer to manage and control the MLKLT process and the related requests associated with transfer learning between two ML entities or between the two ML entities and a shared knowledge repository.

The two use cases should address the three scenarios represented by figures 5.1.12.2.2-1 to 5.1.12.2.2-4. Note that, the use case 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.

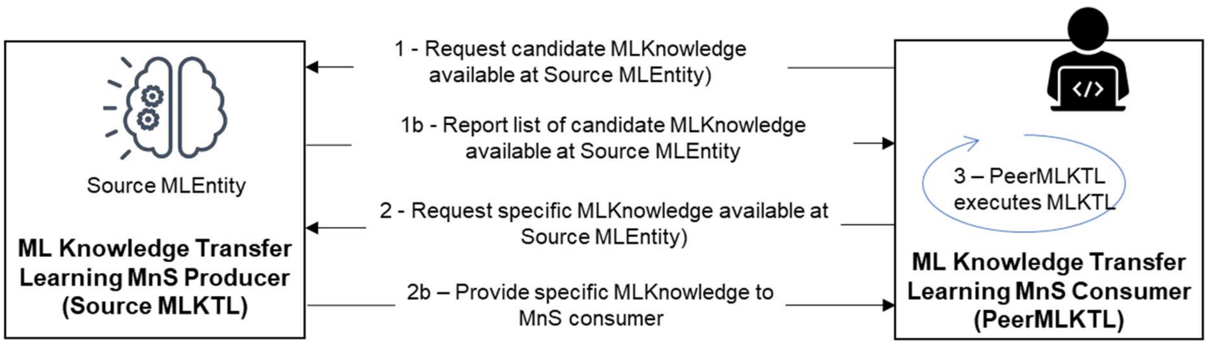

Source MLEntity
ML Knowledge Transfer Learning MnS Producer (Source MLKTL)
1 - Request candidate MLKnowledge available at Source MLEntity)
1b - Report list of candidate MLKnowledge available at Source MLEntity
2 - Request specific MLKnowledge available at Source MLEntity)
2b – Provide specific MLKnowledge to MnS consumer
3 – PeerMLKTL executes MLKTL
ML Knowledge Transfer Learning MnS Consumer (PeerMLKTL)

Figure 5.1.12.2.2-1: Scenario 1 - Interactions for ML-Knowledge Transfer Learning (MLKLT) to support training at the ML knowledge Transfer MnS consumer - the ML knowledge Transfer MnS consumer obtains the ML knowledge which it then uses for training the new ML entity based on knowledge received from the MLKLT source MnS producer

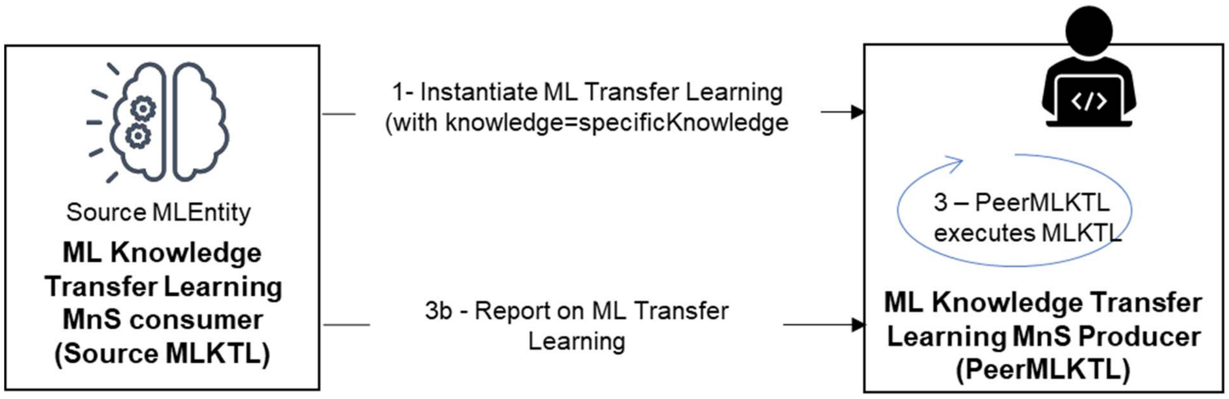

Source MLEntity
ML Knowledge Transfer Learning MnS consumer (Source MLKTL)
1- Instantiate ML Transfer Learning (with knowledge=specificKnowledge
3b - Report on ML Transfer Learning
3 – PeerMLKTL executes MLKTL
ML Knowledge Transfer Learning MnS Producer (PeerMLKTL)

Figure 5.1.12.2.2-2: Scenario 2 - interactions for ML-Knowledge Transfer Learning (MLKTL) to support training at the ML knowledge transfer MnS consumer triggered by the MLKTL Source - the ML Transfer Learning MnS consumer acting as the MLKTLSource (the source of the ML knowledge) triggers the training at the ML knowledge Transfer MnS consumer by providing the ML knowledge to be used for the training, the ML Transfer Learning MnS consumer then undertakes the training

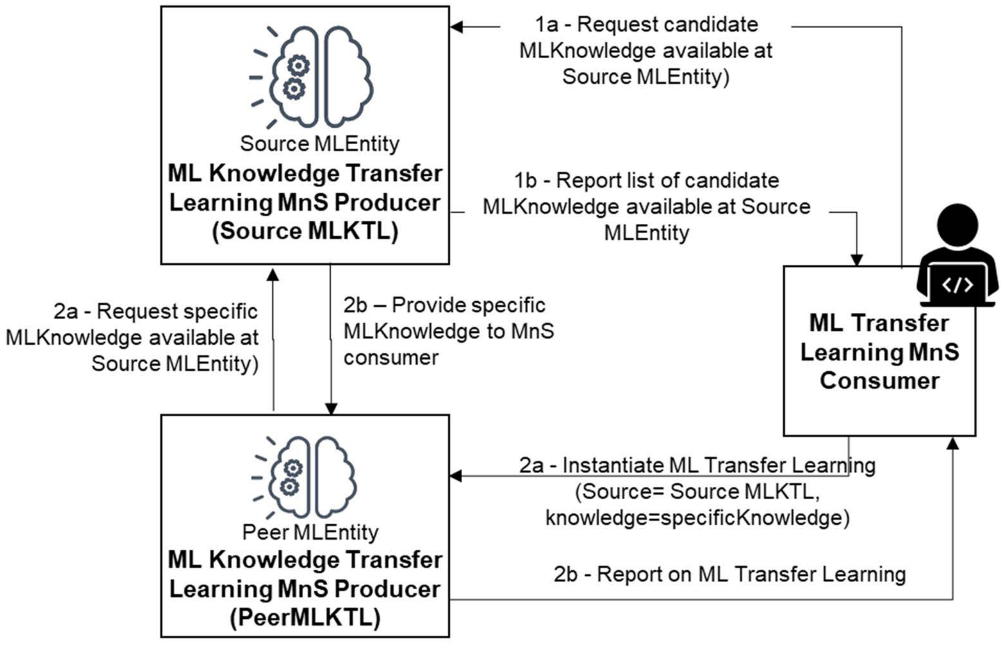

Source MLEntity
ML Knowledge Transfer Learning MnS Producer (Source MLKTL)
1a - Request candidate MLKnowledge available at Source MLEntity)
1b - Report list of candidate MLKnowledge available at Source MLEntity
ML Transfer Learning MnS Consumer
2a - Request specific MLKnowledge available at Source MLEntity)
2b – Provide specific MLKnowledge to MnS consumer
Peer MLEntity
ML Knowledge Transfer Learning MnS Producer (PeerMLKTL)
2a - Instantiate ML Transfer Learning (Source= Source MLKTL, knowledge=specificKnowledge)
2b - Report on ML Transfer Learning

Figure 5.1.12.2.2-3: Scenario 3 - interactions for ML-Knowledge Transfer Learning (MLKLT) to support training at the Peer ML knowledge Transfer MnS producer who is different from the ML knowledge Transfer MnS consumer - the ML knowledge Transfer MnS consumer triggers training at the MLKLT peer MnS producer. The MLKLT MnS consumer then obtains the ML knowledge from the MLKLT source MnS producer and then uses the knowledge for training the new ML entity based on knowledge received from the MLKLT source MnS producer

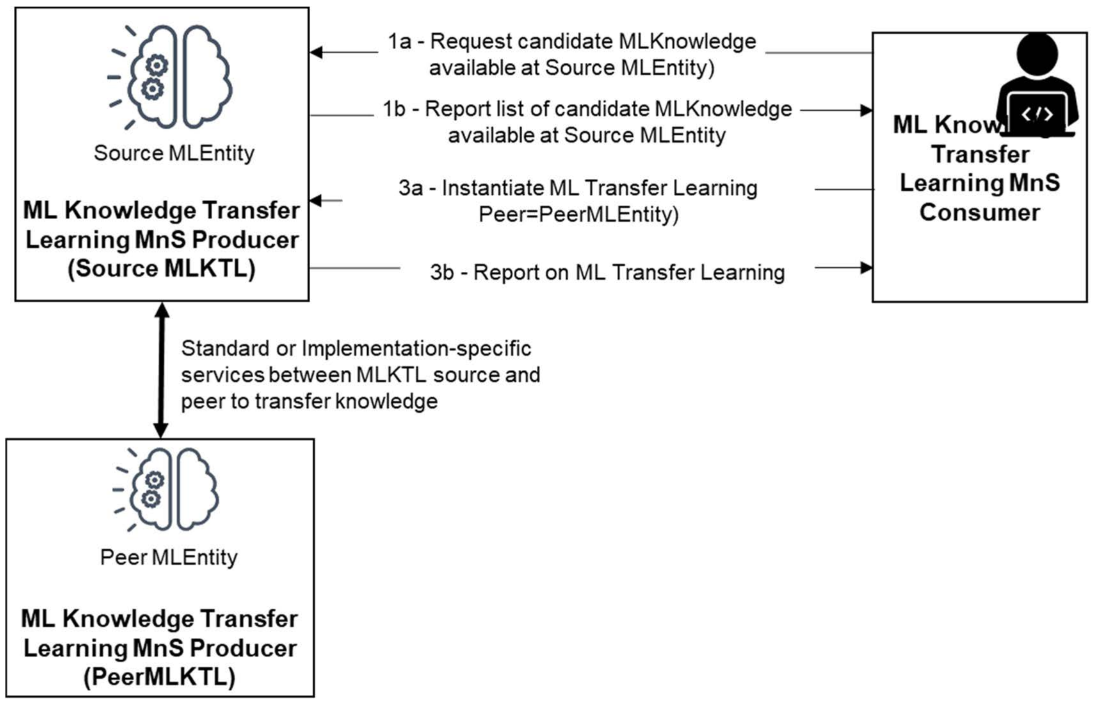

Source MLEntity
ML Knowledge Transfer Learning MnS Producer (Source MLKTL)
1a - Request candidate MLKnowledge available at Source MLEntity)
1b - Report list of candidate MLKnowledge available at Source MLEntity
3a - Instantiate ML Transfer Learning Peer=PeerMLEntity)
3b - Report on ML Transfer Learning
ML Knowledge Transfer Learning MnS Consumer
Standard or Implementation-specific services between MLKTL source and peer to transfer knowledge
Peer MLEntity
ML Knowledge Transfer Learning MnS Producer (PeerMLKTL)

Figure 5.1.12.2.2-4: Scenario 4 - interactions for ML-Knowledge Transfer Learning (MLKLT) to support training at the Source ML knowledge Transfer MnS producer - the ML knowledge Transfer MnS consumer triggers training at the MLKLT source MnS producer. The MLKLT MnS consumer then obtains the ML knowledge from the MLKLT source MnS producer and then uses the knowledge for training the new ML entity based on knowledge received from the MLKLT source MnS producer

5.1.12.3 Potential requirements

REQ-MLKLT-1: The 3GPP management system should have a capability enabling an authorized MnS consumer to discover the available shared knowledge from a given MLKLT MnS producer according to a stated set of criteria.

REQ-MLKLT-2: The 3GPP management system should have a capability enabling an authorized MnS consumer to request a MLKLT MnS producer to provide some or all the knowledge available for sharing according to some stated criteria.

REQ-MLKLT-3: The 3GPP management system should have a capability for a MLKLT MnS producer to report to an authorized MnS consumer on the available shared knowledge according to a ReportingCriteria specified in a request for information on available Knowledge.

REQ-MLKLT-4: The 3GPP management system should have a capability enabling an authorized MnS consumer to request a MLKLT MnS producer to trigger and execute a transfer learning instance to a specified ML entity or ML-enabled function. Accordingly, the MLKLT MnS producer receives an instantiation of MLKLT as an MLKLTJob specifying the target MLKLT ML-enabled function or ML entity.

REQ-MLKLT-1: The 3GPP management system should have a capability for an authorized MnS consumer (e.g. an operator or the function/entity that generated the request for available Knowledge or for information thereon) to manage the request for knowledge or its information and subsequent process, e.g. to suspend, re-activate or cancel the MLKnowledgeRequest; or to adjust the description of the desired knowledge.

REQ-MLKLT-1: The 3GPP management system should have a capability for an authorized MnS consumer (e.g. an operator or the function/entity that generated the request for MLKLT) to manage or control a specific MLKLTJob, e.g. to start, suspend or restart the MLKLTJob; or to adjust the transfer learning conditions or characteristics i.e. Modify MLKLTJob attributes.

REQ-MLKLT-1: The 3GPP management system should have a capability enabling an MLEntity to register available knowledge to a shared knowledge repository, e.g. through a MLKnowledgeRegistration process.

REQ-MLKLT-1: The 3GPP management system should have a capability enabling KnowledgeRepo to act as the MLKLT MnS Producer to enable an authorized MnS consumer to request the shared knowledge repository to provide information on the available knowledge according to some given criteria.

REQ-MLKLT-1: The 3GPP management system should have a capability enabling KnowledgeRepo to act as the MLKLT MnS Producer to enable an authorized MnS consumer to request the KnowledgeRepo to provide some or all the knowledge available for sharing according to some given criteria.

REQ-MLKLT-1: The 3GPP management system should have a capability enabling KnowledgeRepo to act as the MLKLT MnS Producer to enable an authorized MnS consumer (e.g. an operator or the function/entity that generated the MLKnowledgeRequest) to manage the request, e.g. to suspend, re-activate or cancel the MLKnowledgeRequest ; or to adjust the description of the desired knowledge.

REQ-MLKLT-1: The 3GPP management system should have a capability enabling KnowledgeRepo to act as the MLKLT MnS Producer to enable an authorized MnS consumer (e.g. an operator) to manage or control a specific MLKLTJob, e.g. to start, suspend or restart the MLKLTJob; or to adjust the transfer learning conditions or characteristics.

5.1.12.4 Possible solutions

Discovering sharable Knowledge

To discover sharable knowledge:

- The MnS consumer may send a request to the MLKLT MnS producer to provide information on the available sharable knowledge. In other words, the MLKLT MnS producer receives a request to report on the available sharable knowledge.

- The request may be generic or may state a set of criteria which the knowledge should fulfil.

- The request may be referred to as MLKnowledgeInfoRequest.
- The MLKnowledgeInfoRequest must have informational description (Metadata description) of the task and domain related to the required knowledge or given a network problem.
- An ML entity or a function containing an ML entity may register its available knowledge to a shared knowledge repository, e.g. through a MLKnowledgeRegistration process.
- The MLKnowledgeRegistration must contain informational description (Metadata description) of the task and domain related to the registered knowledge or suitable network problem.

Knowledge sharing and transfer learning

To share knowledge:

- Introduce an IOC for an ML Knowledge request. The MnS consumer may send a request to the MLKLT MnS producer to share a specific kind of knowledge. i.e. the MLKLT MnS producer receives a request to provide sharable knowledge, The request may be referred to as MLKnowledgeRequest.
- Introduce an IOC for an ML transfer learning process or job which is instantiated for any request for transfer learning or ML knowledge transfer. The MLKLT MnS producer instantiates a ML transfer learning process. The process may be referred to as MLKLTJob.
- The MLKLTJob is responsible for adapting the required knowledge into a shareable format with the MLKLT consumer.
- MLKLTJob may be a continuous process where knowledge is shared with the MLKLT consumer frequently to account for updates in the knowledge.

NOTE: It may also be the case that the consumer directly instantiates the MLKLTJob without a separate request.

5.1.12.5 Evaluation

The solution described in clause 5.1.12.4 proposes simple information objects that can enable ML functions to exchange their knowledge to be used towards transfer learning but in a way that enables the vendor specific aspects of the ML models not to be exposed. Therefore, the solution described in clause 5.1.12.4 is a feasible solution to be developed further in the normative specifications.

5.2 Management Capabilities for AI/ML inference phase

5.2.1 AI/ML Inference History

5.2.1.1 Description

For different automation requirements, network and management automation functions (e.g. gNB, MDAS, SON) may apply Machine Learning functionality to make the appropriate inferences in different contexts. Depending on the contexts, the ML entity may take different decisions at inference with different outcomes. The history of such inference decisions and the context within which they are taken may be of interest to different consumers.

5.2.1.2 Use cases

5.2.1.2.1 Tracking AI/ML inference decision and context

The deployed ML entity may take different decisions at inference in different contexts and with different outcomes. The selected decisions may need to be tracked for future reference, e.g. to evaluate the appropriateness/ effectiveness of the decisions for those contexts or to evaluate degradations in the ML entity's decision-making capability. For this, the network not only needs to have the required inference capabilities but needs also to have the means to track and enable usage of the history of the inferences made by the ML entity.

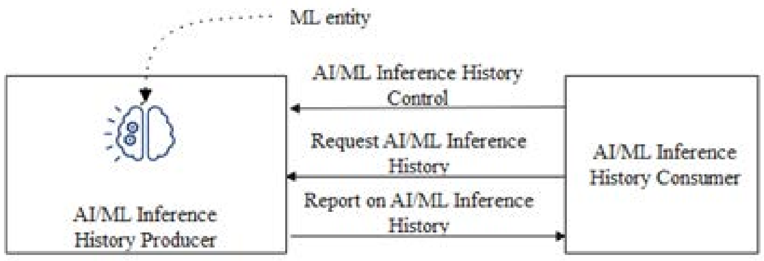

ML entity
AI/ML Inference History Control
Request AI/ML Inference History
Report on AI/ML Inference History
AI/ML Inference History Producer
AI/ML Inference History Consumer

Figure 5.2.1.2.1-1: Example use and control of ML inference history request and reporting

5.2.1.3 Potential requirements

REQ-MLHIST-1: The producer of ML inference history should have a capability allowing an authorized consumer to request the inference history of a specific ML entity.

REQ-MLHIST-2: The producer of ML inference history should support a capability to enable an authorized consumer (e.g. the function/entity that generated the Request for ML inference history) to define the reporting characteristics related to a specific instance of ML inference history or the reporting thereof.

5.2.1.4 Possible solutions

- Introduce ML Inference History (named e.g. MLInferenceHistory) as an IOC, which may be contained in a ManagedFunction, ManagementFunction or subnetwork. The MLInferenceHistory then may contain or be associated with the critical properties and modules needed to accomplish ML Testing, including:
    - The list of MLInferenceHistoryRequests.
    - The list of ML entities either under Testing or to be considered for Testing.
    - MLInferenceHistoryReporting to report on MLInferenceHistoryRequests or their related outcomes.
- Introduce ML Inference History Request (named e.g. MLInferenceHistoryRequest) as an IOC, which may be instantiated by the consumer (e.g. an operator, a managed functions or a management function) for any required history. Each MLInferenceHistoryRequest:
    - May be associated to exactly one deployed ML entity.
    - May contain specific reporting requirements, e.g. one attribute ReportingPeriod may define how the MLInferenceHistory may report about the MLInferenceHistoryRequest.
    - May have a source to identify where its coming from. The sources may for example be an enumeration defined for network functions, operator roles, or other functional differentiations.
- The MLInferenceHistoryRequest may prescribe a list of Reporting-Context, which describes the list of constraints or conditions that may evaluate to true when the Reporting is executed. The Reporting-Context may be a triple <Attribute, Condition, ValueRange > where:
    - Attribute: describes a specific attribute of or related to the object or the use case that relates to the ML entity on which reporting is executed. It may also refer to the characteristics of such object (e.g. its control parameter, gauge, counter, KPI, weighted metric, etc.). It may also refer to an attribute related to the operating conditions of the object or use case (such as weather conditions, load conditions, etc.).
    - Condition: expresses the limits within which the Attribute is allowed/supposed to be. The allowed values for the condition may include: "is equal to"; "is less than"; "is greater than"; "is within the range"; "is outside the range".
    - ValueRange: describes the range of values that are applicable to the Attribute as constrained by the Condition.

- Introduce a ML Inference History Reporting (named e.g. MLInferenceHistoryReporting) as an IOC, which may be used to model the capability of compiling and delivering reports and notifications about MLInferenceHistory or its associated MLInferenceHistoryRequests. The MLInferenceHistory may generate one or more ML Inference History Reports via one or more instances of MLInferenceHistoryReporting:
    - Each ML Inference History Report may be associated to one or more ML entities for which InferenceHistory is requested and/or reported.

NOTE: Potential alignment with the solutions that are being/will be developed on historical data handling as part of the on-going Rel-18 work on management data collection is to be investigated.

5.2.1.5 Evaluation

The solution described in clause 5.2.1.4 adopts the NRM-based approach, proposing three new information object classes with clear association relationship internally and with existing information element "MLEntity". It fully reuses the existing provisioning MnS Operations and notifications for control of Inference History Request and reporting. The implementation of this NRM-based solution is straightforward.

Therefore, the solution described in clause 5.2.1.4 is a feasible solution for AI/ML Inference History.

5.2.2 Orchestrating AI/ML Inference

5.2.2.1 Description

A network automation system may involve or apply multiple AI/ML inference functions and/or ML entities each of which only has a limited view of the network scope. For their effective operation, it may be necessary to apply orchestration mechanisms (be it centralized or otherwise) to orchestrate both the operation of the AI/ML inference functions as well as the execution of the actions recommended by the AI/ML inference functions.

NOTE: The AI/ML inference function is of any function that employs the capabilities of a trained mathematical ML entity (the ML model) or Decision Matrix to make inferences for a specific use case. Such a function may for example optimize load distribution among cells, detect anomalies from data or evaluate the likelihood interference among a set of cells.

5.2.2.2 Use cases

5.2.2.2.1 Knowledge sharing on executed actions

The actions and effects of employing and applying AI/ML inference cannot be known beforehand since they are based on the learnings of the ML entities. An AI/ML inference function may be to optimize one set of parameters but its actions may impact another function. In that case mechanisms are needed to counteract conflicts and or minimize potential negative impacts resulting from conflicting actions brought up by applying AI/ML inference.

When an ML entity A executes an action on the network, that action may affect other network functions. Most critical is that those actions may affect the learning environment (i.e. the training data) of another ML entity, say ML entity B. Correspondingly, the ML entity B needs to be informed when such actions are taken by any ML entity A.

5.2.2.2.2 Knowledge sharing on impacts of executed actions

AI/ML inference functions are able to adjust to adjust their behavior depending on context and on all the information they receive. When an ML entity in function A executes an action on the network that affect other network functions, the ML entity in function A may be able to adjust its behavior to minimize its impact on the other network functions if such an ML entity is informed of its impact on the other network functions. To account for such impacts, the network functions that are affected or the 3GPP management system, needs to inform the ML entity in function A of the observed impacts of the action of the ML entity in function A on the other network functions.

In otherwards, it is necessary that when an action is taken by ML entity in function A, after an appropriate interval (specific to either A or B as may be needed), the network function B (and the other network functions that notice impacts on their metrics or input data) should report their metrics to A. Correspondingly, ML entity in function A may aggregate the reported observations with its own metrics to evaluate the global effect of its actions. In doing so, ML entity in function A is able to learn the best actions that concurrently optimize it's (A's) objective(s) and also minimize the effects on the peers.

The report from B to all may contain values on known KPIs and metrics, e.g. those standardized in 3GPP TS 28.552 [8] and TS 28.554 [14].

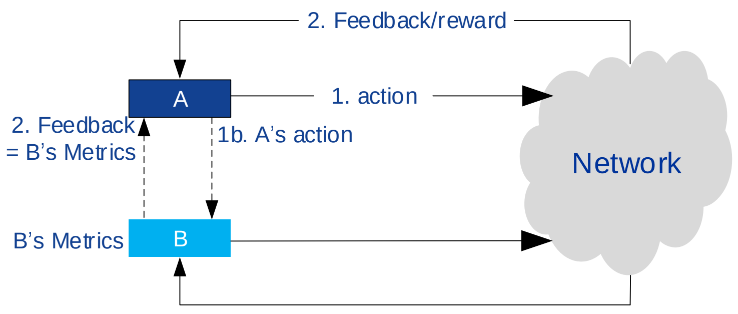


Figure 5.2.2.2.2-1: Distributed coordination of Cognitive Network Automation Functions (NAF)

5.2.2.2.3 Abstract information on impacts of executed actions

In a multivendor environment, the KPIs semantics differ and KPIs that measure one event may be named and computed differently by two vendors. e.g. the Handover rate (H) could be Handovers per user per unit time or Handovers per cell per unit time. Consequently, there is no guarantee that the exchanged KPI or metric values will be interpretable by the ML entity A when it receives that metric.

Instead, it is better when the ML entity B expresses its level of dissatisfaction or impact of the action that was taken by ML entity A. The level of dissatisfaction or impact may be expressed in terms of an Action Quality Indicator (AQI) that is a generic measure that uses a fixed scale to quantify the effect of one function on another. This is similar to the way the Composite Available Capacity (CAC) was specified for cell load to communicate used vs. available cell capacity among cells from different vendors and with different total resources.

For the AQI, if ML entity A takes an action, its effects on the peers will range from an extremely negative impact, e.g. like Mobility Load balancing causing too many mobility related Radio Link Failures; through mild effects that are insignificant (like MLB causing a few handover ping pongs) and to very positive effects (like MRO unexpectedly removing overload in a cell). Consequently, a simple linear measure can easily be used to capture these effects.

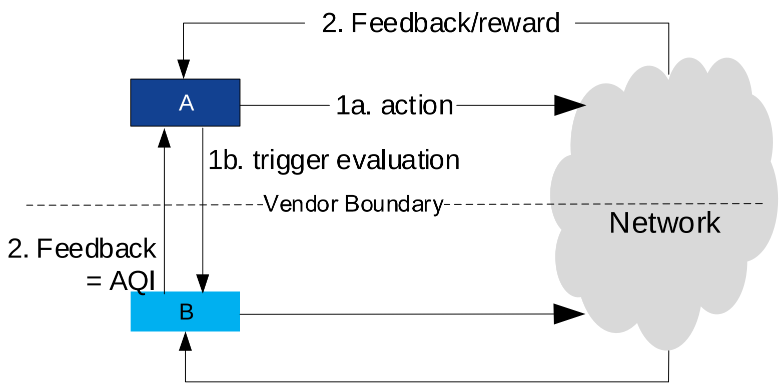


Figure 5.2.2.2.3-1: Multi-vendor coordination of AI/ML inference network automation functions

5.2.2.2.4 Triggering execution of AI/ML inference functions or ML entities

A network automation system may involve or apply multiple AI/ML inference functions or ML entities. These ML entities may not conflict with one another but may focus on only a subset of the problems and may propose changes that are suboptimal since each focus on only a subset of the network control parameters. It may happen frequently that the individual ML entities do not know the expected end-to-end performance of the network, i.e. the ML entities need to be explicitly called to act by an entity which has a wider view of the network problems and the capabilities of the ML entities. For example, consider the distributed AI/ML inference functions (i.e. those instantiated within the gNB) with effects across multiple managed objects such as interference management which may impact multiple cells. Such AI/ML inference functions may not be able to have a wider view of the network state. As such, a centralized controller (i.e. a controller that with a wider and common view to the set of managed objects) is needed to control and coordinate both centralized and distributed AI/ML inference functions. Specifically, the controller may (based on received network data and analytics insight):

- diagnose network problem(s) to identify the nature of the problem; and
- receive the capabilities of the available AI/ML inference functions either directly from the AI/ML inference functions or from a Capability Library that acts as a registry to which the capability of each NAF is added each time a new NAF introduced into the system; and
- evaluate the capabilities of the AI/ML inference functions to identify the best (set and sequence of) AI/ML inference functions to address the identified problem(s); and
- trigger the ML entities to act, providing at trigger time any required extra generalized or specific information.

5.2.2.2.5 Orchestrating decisions of AI/ML inference functions or ML entities

Given the multiple ML entities which may differ in terms of source vendors and behavioural characteristics, the operator may not find it appropriate to grant access to the network to all the different ML entities (both for security and operability reasons).

In that case, there is a need for an orchestration functionality that takes responsibility for the end-to-end performance of the Autonomous Network and that supervises the ML entities to guarantee the end-to-end performance. The orchestration functionality receives the recommended changes from the ML entities, evaluates the proposed changes and their likely effects, decides the changes that should be executed on the network (e.g. to minimize concurrent changes on the same network resources) and informs the ML entities of the respective feedback related to their recommended actions. The orchestration function may also (re)configure the ai based on the observed effects of the actions of the ML entities (e.g. to redefine the control parameter space of the individual ML entities). In either cases, the orchestration function may rely on network states analytics functions which may provide insights that characterize the state of the network into specific states. Such insights may for example characterize whether the network is experiencing low traffic states or anomaly states.

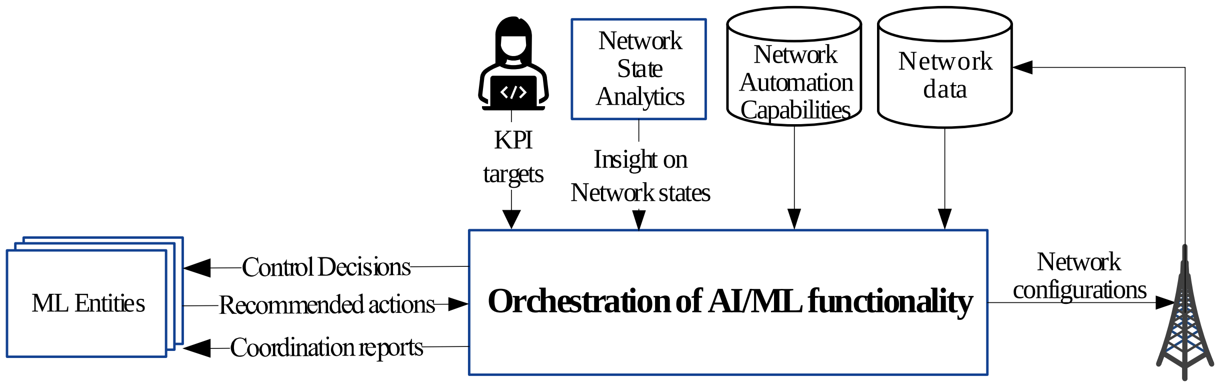


Figure 5.2.2.2.5-1: Orchestrating AI/ML

5.2.2.3 Potential requirements

REQ-ML_ORCH-1: The AI/ML inference MnS producer should have a capability to inform an authorized consumer (e.g. another AI/ML inference function) of actions undertaken by the producer of AI/ML inference.

REQ-ML_ORCH-2: The AI/ML inference MnS producer should support the capability to request a producer of AI/ML action evaluation (e.g. another AI/ML inference function) to evaluate one or more actions undertaken by the producer of AI/ML inference.

REQ-ML_ORCH-3: The AI/ML inference MnS producer should support the capability to specify to the producer of AI/ML-Action-evaluation (e.g. another AI/ML inference function) requested to evaluate one or more actions undertaken by the producer of AI/ML inference the timing within which the consumer should report the observed effects of that evaluated actions.

REQ-ML_ORCH-4: The AI/ML inference MnS producer should support the capability to report the metrics of another AI/ML inference MnS producer that are affected by the one or more actions undertaken by a specific AI/ML inference producer.

REQ-ML_ORCH-5: The AI/ML inference MnS producer should support the capability to report an Action Quality Indicator as the abstraction of the impacts of the one or more actions undertaken by a specific first AI/ML inference producer on a specific metric of the first AI/ML inference producer.

REQ-ML_ORCH-6: The 3GPP Management system should have a capability for an authorized consumer to configure a producer of AI/ML orchestration to monitor recommendations of multiple AI/ML inference functions and decide on the appropriate recommendation to activate on the network.

5.2.2.4 Possible solutions

A single solution may be provided to support the different requirements in clause 5.2.2.3 as follows:

Information elements:

- Introduce an <<IOC>> for centralized Orchestration of AI/ML inference functions. The <<IOC>> which may be named `AIMLOrchestration`, would function as the centralized Automation Controller that takes responsibility for the end-to-end performance of the complete set of network functions that apply AI/ML capabilities.

- Introduce an <<IOC>> for a function that contains AI/ML to be used for network automation. This may be called an AI/ML inference function or network automation function since it is likely to have similar features with or without AI/ML. This <<IOC>> may be named as a `NetworkAutomationFunction` <<IOC>>.

- Introduce an <<IOC>> for a Network Automation Capability Library as an attribute of the `AIMLOrchestration`. The <<IOC>> which may be named a `CapabilityLibrary`, stores the capabilities of the AI/ML inference functions or ML entities that the `AIMLOrchestration` needs to orchestrate. The `AIMLOrchestration` uses a Network Automation Capability Library as a database in which it registers the capabilities of the different network automation functions available, i.e. when an AI/ML inference function or ML entity is added to the system, its capabilities or the problems it can solve as well as the KPIs it optimizes are registered with the `CapabilityLibrary` The `AIMLOrchestration` may populate the `CapabilityLibrary` by querying the individual AI/ML inference functions or ML entities for their capabilities.

- Introduce a <<`dataType`>> for the network performance Targets. Through the network performance Targets, the `AIMLOrchestration` receives the technical objectives that are expected to be achieved. These are set either by the human operator or by a network automation function responsible for deriving concrete objectives from the operators desired goals. Such an objectives-setting function is here referred to as the Network Objectives Manager. The `AIMLOrchestration` monitors the NAFs to ensure that all are contributing towards achieving the objectives and not towards impeding objective achievement.

- Introduce a <<`dataType`>> on the `AIMLOrchestration` for the Network state. The datatype which may be called the NetworkState indicates, and labels specific unique states of the network as derived from specific combinations of raw network data. The state may be derived from an external ML entity that shares that state with all interested entities or it may be derived by the `AIMLOrchestration`. The Network state aids the `AIMLOrchestration` not only to relate states observed in different time periods but to also reference states in a way that is understandable to other entities, e.g. while communicating to the AI/ML inference functions or ML entities.

- Introduce a <<`datatype`>> on the `AIMLOrchestration` for a recommended action from a network automation function. The datatype which may be called the recommendedAction<<datatype>> captures the recommended policy and configuration change of a specific network automation function, e.g. an AI/ML inference function or ML entity towards the `AIMLOrchestration`. - All recommendations for policy changes as computed by the network automation functions for their respective objectives are communicated via this recommendedAction. Such a recommendedAction may be a hash function of parameter to parameter-value annotated with an indication of the time within which the change should be activated or otherwise discarded.

- The `recommendedAction` <<`datatype`>> may also include a field for the eventual action that is selected by the `AIMLOrchestration`, say called the selectedAction. This is written by the `AIMLOrchestration` with the specific values that have been applied by the `AIMLOrchestration` following which a notification may be sent to the network automation function that generated the recommendedAction. The selectedAction may also be used by the `AIMLOrchestration` to inform the network automation functions if the recommended policy change has been activated or not and possibly the reason thereof. The respective message sent to a network automation function may for example be verbalized as follows:

  - "In network state A1, when policy X changed from configuration X1 to X2, the observed effect on the network KPI vector v exceeded a predefined threshold. Consequently, policy configuration X2 is now barred from your applicable control and operational parameter spaces."

NOTE 1: the `AIMLOrchestration` is also tasked with reconfigurations of the control and operational parameter spaces of the network automation functions to adjust the limits within which the network automation functions may operate. For example, for a load balancing function, the `AIMLOrchestration` may adjust the limits to which the load balancing function may adjust the Cell Individual Offset (CIO) by setting the maximum or minimum CIO or steps within which the CIO may be changed. For these reconfigurations, the `AIMLOrchestration` may use existing NRMs for the network automation functions, e.g. the `MLEntity` NRM, to change the attributes of the network automation functions. For example, the `AIMLOrchestration` may mask a part of the control and operational parameter spaces such that those masked values become inaccessible for the network automation function. It may also use the existing NRMs to activate or deactivate particular network automation functions as may be necessary (e.g. based on network context).

NOTE 2: The `AIMLOrchestration` may be the only responsible entity for activating the selected inference decisions onto the network. For this, the `AIMLOrchestration` may activate the successful recommended policies and/or configurations on the network via the existing NRMs for the network objects or existing CM capabilities.

- Introduce an <<IOC>> on the network automation function, e.g. on the AI/ML inference function or on the ML entity for a request for monitoring. The IOC which may called `metricMonitoringRequest`, may be used by the `AIMLOrchestration` or by any network automation function A to request another network automation function B to start a monitoring of the metrics of the of network automation function B and subsequently report the outcomes of the monitoring.

- Introduce a <<`datatype`>> on the `AIMLOrchestration` for an indication of the observed effect of a given action on the metrics of given network automation function. The IOC which may be called the `ActionQualityIndicator`, provides information to the source network automation function (i.e. the function that generated the action) about how good or bad that action was to the metrics of the reporting function.

Usage of the information Elements:

- The `AIMLOrchestration` or a multi-functional analytics service of the `AIMLOrchestration` evaluates the network state to diagnose what the network or network resource problem might be. In general, such a problem cannot be concluded from a single KPI, otherwise, the NAF responsible for that KPI should be triggered by default. Instead, it is typically a rare event which can only be determined from multiple KPIs. For identifying the problem, the `AIMLOrchestration` may collect data on KPIs. Counters, CM values etc. The combination of KPIs. Counters, CM values, etc., may be correlated e.g. using an analytics service to identify the problem and the specific combination may be labelled as a specific network state.

NOTE 3: The problem may also be pointed to using analytics service such as MDAS.

- For the identified problem, the `AIMLOrchestration` finds the most appropriate network automation function to trigger. It could also be the case that there are multiple network automation functions responsible for a given KPI, which could say happen if there is an open network automation platform to which multiple vendors have supplied network automation functions. In such a case, the network automation function to be triggered by default is not obvious and either the `AIMLOrchestration` or an analytics function needs to figure out the best network automation function to trigger. The `AIMLOrchestration` queries the `CapabilityLibrary` to match the identified problem to one of the sets of network automation functions that are registered in the library.
- The `AIMLOrchestration` then triggers the identified network automation function to find an appropriate action for the problem. The trigger may be sent via a ProblemResolutionRequest sent by the `AIMLOrchestration` to the network automation function. The `ProblemResolutionRequest` may include an identifier for the managed object related to the problem as well as the KPIs. Counters, CM values related to the observed problem.
- The selected network automation function submits a proposed recommended action to the `AIMLOrchestration` for execution. The AIML orchestrator may also undertake coordination action (as described next) to ensure the action is not opposite to the interests of other network automation functions.
- At the end of the cycle, the `AIMLOrchestration` determines the next action, either to recall the previous network automation function to find a new configuration, or to call a different network automation function to attempt the same or related problem or to move to start a new cycle for a completely different problem if the previous problem has been successfully solved.

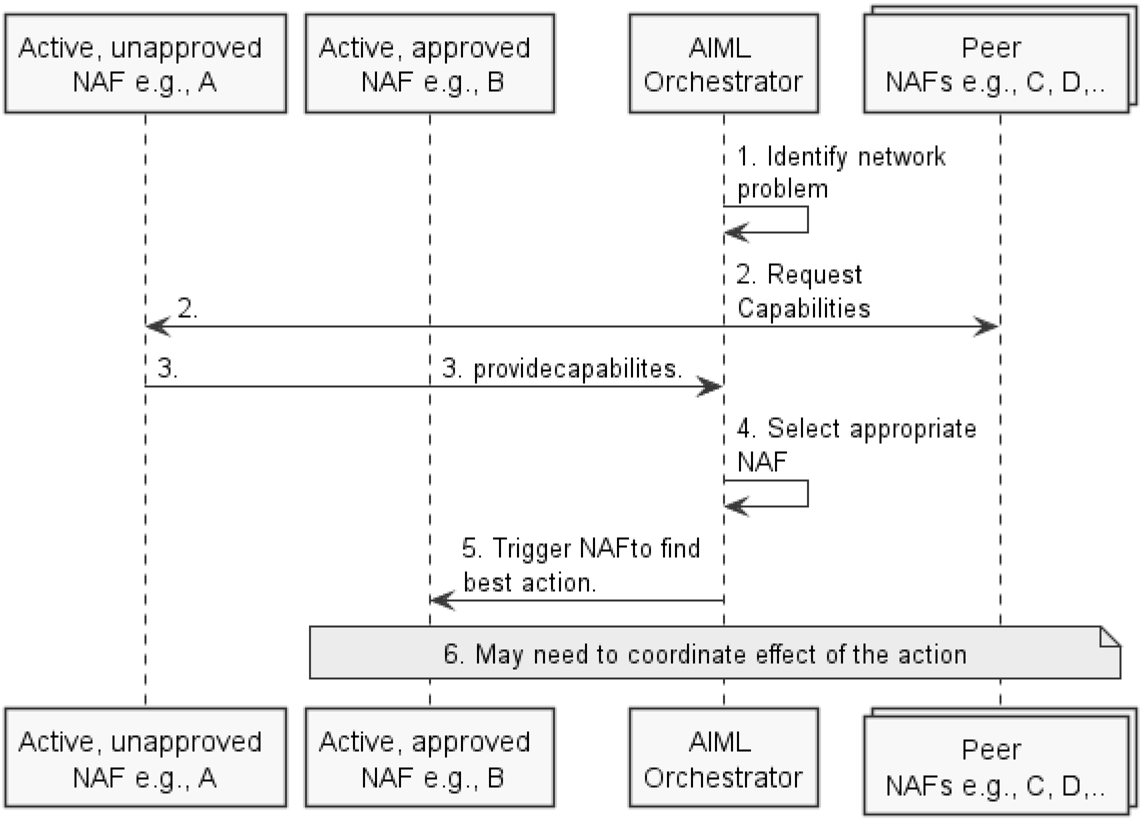

Active, unapproved NAF e.g., A
Active, approved NAF e.g., B
AIML Orchestrator
Peer NAFs e.g., C, D,..
1. Identify network problem
2. Request Capabilities
2.
3.
3. providecapabilites.
4. Select appropriate NAF
5. Trigger NAFto find best action.
6. May need to coordinate effect of the action

Figure 5.2.2.4-1: Identifying and triggering automation capabilities among multiple network automation functions

To Orchestrate the decisions of AI/ML network automation functions, e.g. AI/ML inference functions or ML entities (see figure 5.2.2.4-2):

- Each network automation function generates a recommendation which it proposes to the `AIMLOrchestration` for implementation. The `AIMLOrchestration` takes recommendations for changes from the network automation functions and takes a decision whether to implement the policy changes or not. The policy changes may be stated by the network automation functions as hash functions of parameter to parameter-values annotated with the time within which the values should be activated or else be discarded.

- The `AIMLOrchestration` undertake control tasks for the recommended and approved configuration changes, e.g. concurrency control, to ensure that their action will not conflict with other ongoing or proposed actions. In case of conflicts the `AIMLOrchestration` may choose to schedule the action to a different time from when it is proposed.

- At the right time, the `AIMLOrchestration` implements the action onto the network and subsequently manages the coordination.

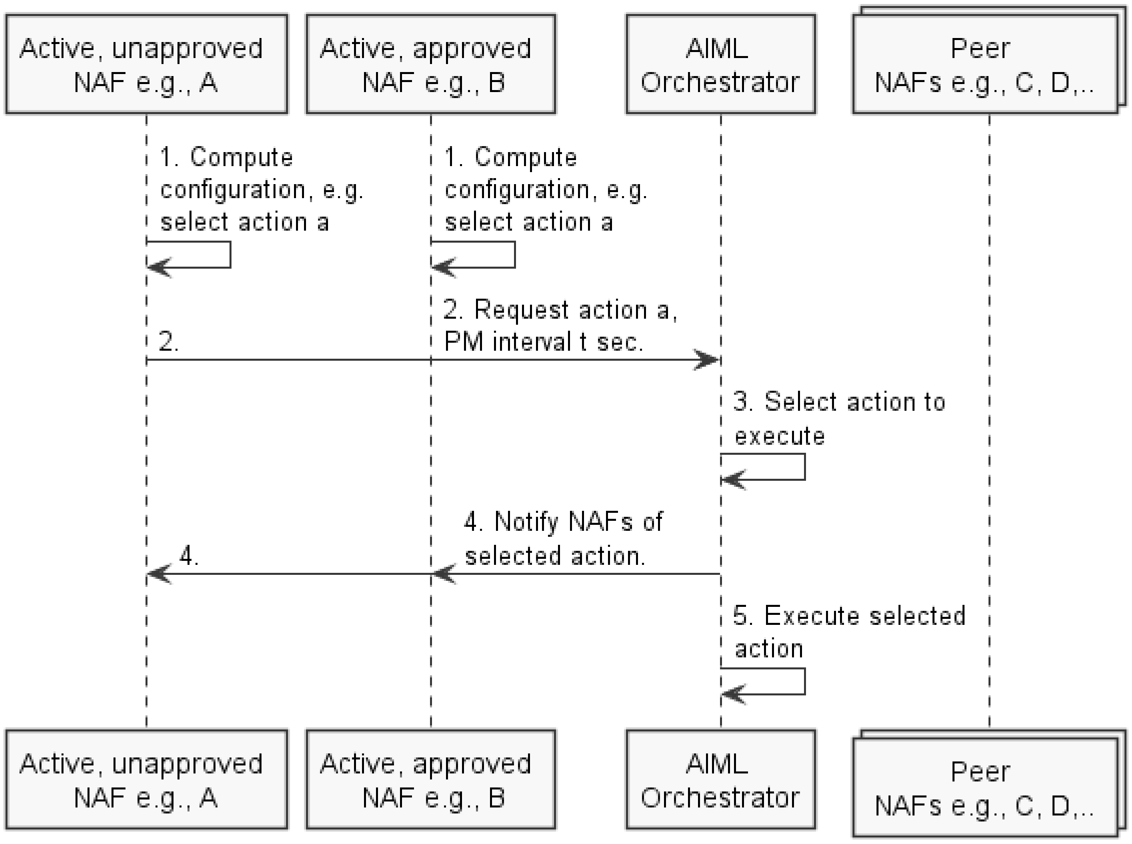

Active, unapproved NAF e.g., A
Active, approved NAF e.g., B
AIML Orchestrator
Peer NAFs e.g., C, D,..
1. Compute configuration, e.g. select action a
1. Compute configuration, e.g. select action a
2.
2. Request action a, PM interval t sec.
3. Select action to execute
4.
4. Notify NAFs of selected action.
5. Execute selected action
Active, unapproved NAF e.g., A
Active, approved NAF e.g., B
AIML Orchestrator
Peer NAFs e.g., C, D,..

Figure 5.2.2.4-2: Orchestrating the decision among multiple network automation functions

To share knowledge and coordinate the impacts executed decisions of AI/ML network automation functions, e.g. AI/ML inference functions or ML entities (see figure 5.2.2.4-3):

- The `AIMLOrchestration` is a meta-learning agent responsible for learning if any of the availed network automation functions is behaving outside its expected region and for taking the accordingly appropriate counter-measures. So, for the activated configuration changes, the `AIMLOrchestration` manages the transaction among network automation functions that are intended to coordinate their actions and executions.

- Following the execution of changes, the `AIMLOrchestration` triggers the other network automation functions (besides the one requesting the change) to start a monitoring period to identify any negative effects on their metrics.

- At the end of the observation period the network automation functions evaluate the effects of the changes and report to the `AIMLOrchestration` their network (metric) status observations in form of Action Quality Indicators. The network automation functions consume performance assurance (PM), fault supervision (FM) and provisioning (CM) services on their respective network elements and domains to evaluate the effect of the executed policy changes or configurations.

- The network automation functions report the observed effects in terms of the Action Quality Indicators to the `AIMLOrchestration` for aggregation. Note that, in a distributed implementation of the coordination, the Action Quality Indicators may also be reported directly to the network automation function which generated the action that was executed.

- In the subsequent AQI handling, the `AIMLOrchestration` evaluates the network status inputs from the multiple network automation functions and network domains to learn the effects of the configuration changes, i.e. the `AIMLOrchestration` determines if the effects are acceptable or not.

- Where a given change is determined to be out of the expected range, such a change needs to be labelled accordingly. For example, the change may be barred from ever being re-applied or from being reused in the specific context.

- The `AIMLOrchestration` informs the respective network automation function of the evaluation outcome (e.g. by sending the aggregate AQI). The `AIMLOrchestration` also accordingly re-configures the network automation functions, when necessary, e.g. by changing the network automation function's applicable control parameter spaces or its performance targets.

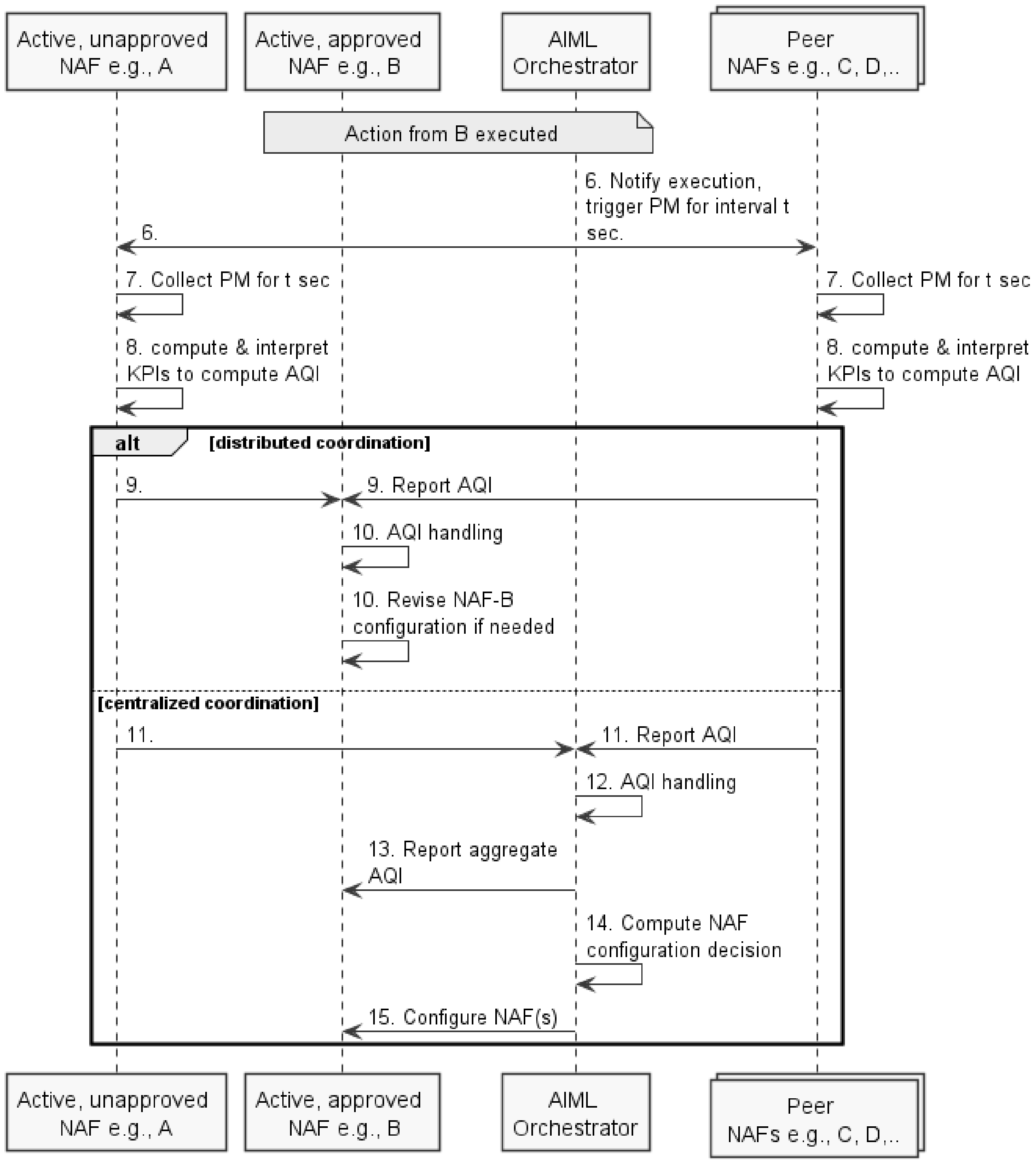

Active, unapproved NAF e.g., A
Active, approved NAF e.g., B
AIML Orchestrator
Peer NAFs e.g., C, D,..
Action from B executed
6. Notify execution, trigger PM for interval t sec.
6.
7. Collect PM for t sec
7. Collect PM for t sec
8. compute & interpret KPIs to compute AQI
8. compute & interpret KPIs to compute AQI
alt [distributed coordination]
9.
9. Report AQI
10. AQI handling
10. Revise NAF-B configuration if needed
[centralized coordination]
11.
11. Report AQI
12. AQI handling
13. Report aggregate AQI
14. Compute NAF configuration decision
15. Configure NAF(s)

Figure 5.2.2.4-3: The control and coordination transaction of network automation functions requests and actions

5.2.2.5 Evaluation

The solution described in clause 5.2.2.4 introduces new information elements that together solve all three aspects for the orchestration of AI/ML inference, i.e. the identification of the right AI/ML inference function or ML entity to solve a given problem, the orchestration of the actions of the AI/ML inference function or ML entities and the sharing of knowledge among the different AI/ML inference functions or ML entities about the performance of the individual AI/ML inference functions or ML entities.

Therefore, the solution described in clause 5.2.2.4 is a feasible solution for Orchestrating AI/ML inference.

5.2.3 Coordination between the ML capabilities

5.2.3.1 Description

For ML in 5GC or RAN, the ML capabilities in 5GC or RAN may be needed to coordinate with 3GPP management analytics and possibly other aspects in order to improve the overall performance.

Typically, due to the type of the collected data used for model training/inference for 5GC or RAN, the performance of a model in 5GC or RAN may be biased in some respects. On the other hand, the 3GPP management system collects data of longer range and from a wide scope of RAN nodes/5GC, which consequently implies that the predictions calculated by the management system will be unbiased towards the overall RAN nodes/5GC. However, 3GPP management system predictions may lack insight of patterns related to specific node behaviour or finer granularity of time. Hence with coordination or alignment of the ML capability among 5GC/RAN and 3GPP management system, the overall performance may be improved.

To enable the coordination between the ML capabilities, the configuration (e.g. a triggering condition, i.e. when a result is needed from the RAN analytics to MDA) may be needed. On the other hand, the result of the coordination may be communicated towards the consumer(s) and hence there is a need to enhance the reporting in order to capture the deviation of predictions (and/or the related context) so the consumer can gain a better understanding regarding the coordination of ML capabilities.

5.2.3.2 Use cases

5.2.3.2.1 Alignment of the ML capability between 5GC/RAN and 3GPP management system

Generally, the typical data from 5GS data is measurements (PM, KPI), which may be modeled as "time series data" and the analytics of "time series data" is normally to learn the seasonality, trend, etc., patterns. Different types of seasonality patterns exist, e.g. daily, weekly, monthly, seasonally, annually, etc. A RAN node collects finer granularity data for short duration. The finer seasonality pattern will be well captured in a timely manner, while the 3GPP management system with longer range of data (which is more aggregated) will more accurately capture the higher level of seasonality patterns. Hence, combing the analytics results from RAN, 5G core and 3GPP management system or between RAN and 5G core may improve the accuracy for overall predictions.

On another matter, the data collected from one RAN or 5G core node would tend to be biased for that specific RAN node or NF, which implies that the prediction with the data learned and inferred will tend to be biased for that specific node. On the other hand, the 3GPP management system collects data of longer range and from a large amount of different RAN nodes or NFs, which consequently implies that the prediction will be unbiased towards the overall RAN nodes or 5G core area. Hence combining the results from both the RAN or 5G core and 3GPP management system, or between NWDAF and RAN may also improve the overall predictions accuracy (and mitigate the bias).

5.2.3.3 Potential requirements

REQ-AIML_COORD-01: 3GPP management system should have the capability to allow an authorized consumer to configure an ML capability regarding the correlation of predictions and statistics between MDAS and RAN function, or MDAS and NWDAF.

REQ-AIML_COORD-02: 3GPP management system should have the capability to report the result of correlation of predictions and statistics between MDAS and RAN function, or MDAS and NWDAF.

5.2.3.4 Possible solutions

5.2.3.4.1 Possible solution #1

1) Introduce the information Elements (e.g. instance of IOC or a dataType) for interaction between ML MnS producer and consumer (e.g. the RAN analytics or NWDAF, or entity consuming the RAN analytics or NWDAF) to support coordination of the ML capability between 5GC/RAN and 3GPP management system: `MLCapabilityCoordinationRequest` and a response informant element `MLCapabilityCoordinationResponse`. This information element may represent the triggering configuration (or a triggering policy) for predictions coordination from two ML capabilities. This information Element may allow an MOI (or MOI using this information element) to be created on the ML (inference) MnS Producer and may contain the following attributes:

   - The analytics deviation indicator, such as a threshold (determining that the prediction calculated by a data analytics function exceeds a configurable certain value, corresponding to the prediction available at a different data analytics function, by a "threshold").
   - The requested analytics (analytics type name, list of analytics values or output, time intervals, confidence degree, etc.).
   - The target objects, e.g. gNBs, and the related characteristics.
   - Area of interest, geographical area or TA.

2) MLCapabilityCoordinationResponse- this information element may represent the response indicating the analytics or data obtained according to the MLCapabilityCoordinationRequest. This information Element may be created by the ML MnS (inference) producer towards the MnS consumer and includes output analytics which can be statistics or predictions. The MLCapabilityCoordinationResponse and MLCapabilityCoordinationRequest may also be data attribute in analytic request and response or analytics report.

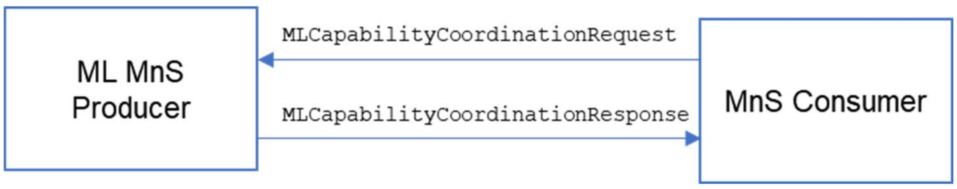

ML MnS Producer
MLCapabilityCoordinationRequest
MLCapabilityCoordinationResponse
MnS Consumer

Figure 5.2.3.4.1-1: Interaction between ML MnS producer and consumer to support coordination of the ML capability

5.2.3.5 Evaluation

The solution described in clause 5.2.3.4.1 proposes simple information elements and procedure that may enable MnS Producer to trigger configuration to be used for analytic coordination. This NRM based solution reuses the existing provisioning MnS Operations and notifications for control and reporting. Therefore, the solution described in clause 5.2.3.4.1 is a feasible solution to be developed further in the normative specifications.

5.2.4 ML entity loading

5.2.4.1 Description

ML entity loading refers to the process of making an ML entity available in the operational environments, where it could start adding value by conducting inference (e.g. prediction). After a trained ML entity meets the performance criteria per the ML entity testing, the ML entity could be loaded in target inference function(s) in 3GPP system, e.g. via a software installation, file transfer, or a configuration management procedure and subsequently activated. The ML entity loading may be requested by the consumer or initiated by the producer based on the loading policy (e.g. the threshold of the testing performance of the ML entity, threshold of the inference performance of the existing ML model, predefined time schedule, etc.) provided by the consumer.

The loading of ML entity has no implication about "push" or "pull" method.

After an ML entity is loaded in the target inference function, the data fed to the ML entity may change to the level where it is different from the data used in the initial prior training of the respective ML entity. To improve model performance with the changed data, the ML entity therein may need to be retrained and reloaded.

5.2.4.2 Use cases

5.2.4.2.1 ML entity loading control and monitoring

This use case is appliable to the deployment scenario where the ML training function and inference function are not co-located.

After the ML entity is trained and tested, the ML entity needs to be loaded by the ML entity loading MnS producer to the target inference function(s) per the request from the MnS consumer or initiated based on a consumer predefined loading policy.

NOTE: ML entity loading MnS producer may be a separate entity or co-located with the MnS producer of the inference function or training function.

One potential reflection of loading policy is to enable a scheduled loading. ML models are typically trained and tested to meet specific requirements for inference, addressing a specific use case or task. Inference requirements could change regularly. For example, a network node supported by AI/ML capability may require employing a specifically trained/different type of ML entity at different time of day, or a specific day in the week with an already known repeated pattern. For example, a gNB providing coverage for a specific location is scheduled to accommodate different load level and/or pattern of services at different time of the day. A dedicated ML model (specifically trained and/or varying type altogether) may be required.

Once the ML entity has been loaded in the target inference function(s), some MnS consumers may need to know the available information of ML entity and to determine the next appropriate action. In this case the MnS consumer needs to be notified about the ML entity loading or be able to retrieve the loading information of the ML entity. This would allow the consumer to e.g. request ML entity re-training if e.g. performance fall below certain threshold or request the loading of different ML entity altogether, etc.).

The general information used to describe a loaded ML entity may include:

- Resource information, which describes the static parameters of the ML entity (e.g. mLEntityVersion, mLEntityId, trainingContext, see 3GPP TS 28.105 [4]).
- Management information, which describes the information model that is used for ML entity lifecycle management (e.g. activation flag, status, creation time, last update time).
- Capability information, which describes the capability information (e.g. inference type, performance metrics).

NOTE: The liability aspect on loading the ML entity to inference function is FFS.

5.2.4.3 Potential requirements

REQ-MODEL_DPL-CON-1: The ML entity loading MnS producer should have a capability allowing the consumer to request and retrieve loading information of an ML entity.

REQ-MODEL_DPL-CON-2: The ML entity loading MnS producer should have a capability to notify the consumer about the loading information of an ML entity.

REQ-MODEL_DPL-CON-3: The ML entity loading MnS producer should have a capability allowing the consumer to request the loading of an ML entity to the target inference function(s).

REQ-MODEL_DPL-CON-4: The ML entity loading MnS producer should have a capability allowing the consumer to provide the loading policy for an ML entity.

5.2.4.4 Possible solutions

5.2.4.4.1 NRM based solution

This solution uses the instances of following IOCs for interaction between ML loading MnS producer and consumer to support the ML entity loading, where the ML loading MnS producer could be part or a separate entity of the inference function:

1) The IOC representing the ML entity loading request, named for example as `MLEntityLoadingRequest`.

   This IOC is created by the ML entity loading MnS consumer on the producer, and it contains the following attributes:

   - identifier of the ML entity to be loaded;
   - the identifier (e.g. DN) of target inference functions where the ML entity is loaded to. This attribute is optional if the target inference function is itself that provides the ML entity loading MnS.

2) The IOC representing the ML entity loading policy, for example named as `MLEntityLoadingPolicy`.

   This IOC is created by the ML entity loading MnS consumer on the producer, so that the producer can load the ML entity according to the policy without an explicit loading request from the consumer, and it contains the following attributes:

   - identifier or inference type of the ML entity to be loaded;
   - trigger of ML entity loading, including e.g. pre-defined scheduled loading, a threshold of the testing performance of the ML entity and/or a threshold of the inference performance of the existing ML entity in the target inference function(s);
   - identifier (e.g. DN) of target inference functions where the ML entity is loaded to. This attribute is optional if the target inference function is itself that provides the ML entity loading MnS.

3) The IOC representing the ML entity loading process, for example named as `MLEntityLoadingProcess`.

   This IOC is created by the ML entity loading MnS producer and reported to the consumer, and it contains the following attributes:

   - identifier of the ML entity being loaded;
   - associated ML entity loading request;
   - associated ML entity loading policy;
   - identifier (e.g. DN) of the target inference function; This attribute is optional if the target inference function is itself that provides the ML entity loading MnS;
   - loading progress;
   - control of the loading process, like cancel, suspend and resume.

   How to load the ML entity by the MnS producer is vendor specific.

4) The IOC representing the ML entity loaded in the inference function, for example by extension of the existing IOC (`MLEntity`) representing the ML entity, or by a new IOC.

   This IOC is created by the ML loading MnS producer and reported to the consumer, and it contains the following attributes:

   - identifier of the loaded ML entity;
   - associated trained ML entity (e.g. DN of the MOI representing the trained ML entity), which is to be loaded to the inference function;
   - associated ML entity loading process;

- status (such as activated, de-activated, etc.) of the loaded ML entity.

The examples of IOCs and their relations between the IOCs are depicted in figure 5.2.4.4.1-1.

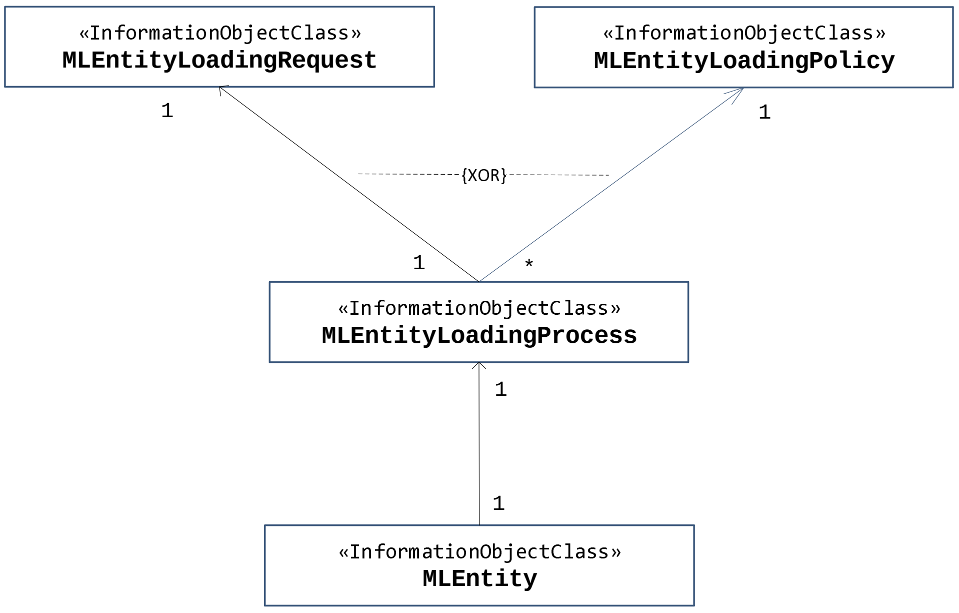


Figure 5.2.4.4.1-1: Example of ML entity loading related NRMs

NOTE: Further details including e.g. the name of the IOCs and corresponding attributes are to be decided in normative phase.

5.2.4.5 Evaluation

The solution described in clause 5.2.4.4.1 adopts the NRM-based approach, which to a great extent reuses the existing provisioning MnS operations and notifications. This solution is also consistent with the approach used by ML training MnS defined in 3GPP TS 28.105 [4]. It does not only reuse the existing capabilities (provisioning MnS operations and notifications), but also cater for the flexibility that is needed to facilitate both co-located and separate implementation and deployment options of ML training and/or testing MnS and ML loading MnS by using the consistent NRM-based approach.

Therefore, the solution described in clause 5.2.4.4.1 is considered a feasible solution.

5.2.5 ML inference emulation

5.2.5.1 Description

A trained ML entity can be used for inference within the stated scope e.g. on a managed function or in a management function. Accordingly, there may be an AI/ML inference MnS producer that is responsible for executing the inference.

5.2.5.2 Use cases

5.2.5.2.1 AI/ML inference emulation

After an ML entity is trained, validation is done to ensure the training process is completed successfully. Typically, validation is done by preserving part of the training data set and using it after training to check whether the ML entity has been trained correctly or not. However, even after the ML entity is validated during development, inference emulation is necessary to check if the ML entity containing the ML entity is working correctly under certain runtime context or using certain inference emulation data set. In principle, the two operations are similar on a functional level, where both of them check the ML performance against given context or data to ensure the ML functionality is functioning correctly. But inference emulation involves interaction with third parties, e.g. the operators who use the ML entity or third-party systems that may rely on the results computed by the ML entity. For these reasons, it is necessary to support inference emulation, specifically to support means:

- For a given MnS consumer to request for a specific AI/ML capability to be executed in ML inference emulator environment.
- For a given MnS consumer to request a specific ML inference emulator to execute a given AI/ML capability.
- For a managed function to act as a ML inference emulator and execute AI/ML capabilities in a controlled way.

The network or its management system needs to have the capabilities and provide the services needed to enable the MnS consumer to request inference emulation and receive feedback on the inference emulation of a specific ML entity or of an application or function that contains an ML entity.

5.2.5.2.2 Managing ML inference emulation

The 3GPP management system may have resources for multiple emulation environments to be used depending on need. These may include simulation environments, a digital twin of the network, a test network or the real network under curtain constrained conditions, e.g. for a selected set of UEs. The multiple emulation environments may represent different levels of trust that the operator or management system has in the ML entity or AI/ML inference functions. Correspondingly, 3GPP management system needs to have means and method for Orchestrating the inference emulation i.e. say called the inference emulation orchestrator or inference emulation function. Accordingly:

- the emulation progression process involves choosing the right type and instance of an emulation environment to which an ML entity, AI/ML inference function or the action thereof may be tested depending on the needs of the function to be tested and the available emulation environments and their resources;
- the emulation process may also involve executing the ML entity, AI/ML inference function or its action on the real network but in a controlled fashion, e.g. only within certain hours or only on cells with a particular kind of load or only on cells in a particular area or in limited subscriber groups.

Relatedly, the actions taken by the inference emulation function may include:

- Controlling the allowed parameter space/ranges of the parameters optimized by the ML entity or AI/ML inference function depending on the emulation environment to which the ML entity, AI/ML inference function or the actions are being executed.
- Adjusting the parameter space in consideration of the observed behaviour of the ML entity or AI/ML inference function.
- Deploying the actions of the ML entity or AI/ML inference function on a selected emulation environment or on the real network.
- Blocking the ML entity or AI/ML inference function from being used on the network.

5.2.5.3 Potential requirements

REQ-AI/ML_EMUL-1: The MnS producer for AI/ML inference emulation should have a capability to allow an authorized MnS consumer to query the available emulation environment(s).

REQ-AI/ML_EMUL-2: The MnS producer for AI/ML inference emulation should have a capability to inform an authorized MnS consumer of the available emulation environment(s).

REQ-AI/ML_EMUL-3: The MnS producer for AI/ML inference emulation should have a capability to allow an authorized MnS consumer to request an ML inference emulation for a specific ML entity or entities.

REQ-AI/ML_EMUL-4: The MnS producer for AI/ML inference emulation should have a capability to allow an authorized MnS consumer to request for ML Inference Emulation for a specific ML entity using specified data or data with specifically stated characteristics and inference emulation features.

REQ-AI/ML_EMUL-5: The MnS producer for AI/ML inference emulation should have a capability to inform authorized MnS consumer about the status of the emulation of an ML entity under emulation.

REQ-AI/ML_EMUL-6: The MnS producer for AI/ML inference emulation should have a capability to allow an authorized MnS consumer (e.g. an operator) to manage or control a specific ML inference emulation process, e.g. to start, suspend or restart the inference emulation; or to adjust the inference emulation conditions or characteristics.

REQ-AI/ML_EMUL-7: The MnS producer for AI/ML inference emulation should have a capability to allow an authorized MnS consumer to request reporting, and receive reports on the progress and outcome of an emulation process.

REQ-AI/ML_EMUL-8: The MnS producer for AI/ML inference emulation should have a capability to allow an authorized MnS consumer to configure an ML entity or AI/ML inference function supporting with the level of trust that expresses the degree to which the ML entity or AI/ML inference function or the different action thereof have been confirmed as trusted.

REQ-AI/ML_EMUL-9: The MnS producer for AI/ML inference emulation should have a capability to graduate an ML entity, AI/ML inference function or the different action thereof through different levels of trust each expressing a different degree to which the ML entity, AI/ML inference function or action has been confirmed as trusted.

5.2.5.4 Possible solutions

1) Introduce an IOC with the properties of the ML inference emulation function. This may be termed as an `MLInferenceEmulationFunction` to be name-contained in either a `Subnetwork`, a `ManagedFunction` or a `ManagementFunction`. The `MLInferenceEmulationFunction` may be a separate function or may be name-contained in an inference function.

   This IOC contains attributes including the following:

   - list of the ML entities which can be emulated and possibly in which emulation environments;
   - indication of progression of the `MLEntity or` AI/ML inference function that is under emulation to indicate the degree to which the inference has been emulated and trusted;
   - different characteristics for which different emulations may be supported;
   - a hierarchy for different emulation environments, e.g. an emulator that uses only a test network vs. an emulator that activates the actions in the real network at specified time such as the maintenance window.

2) Introduce an IOC representing an available emulation environment, e.g. a new IOC named as `AvailableEmulationEnvironment,` or `EmulationSubNetwork, or` the existing `SubNetwork` IOC with an attribute indicating it is a subnetwork used for emulation.

   The instance of this IOC is created by the MnS producer to allow the consumer to query, or be informed of, the information of the available emulation environments. This IOC contains the following properties (e.g. the subordinated IOC or attributes):

   - inference functions or ML entities under emulation.

3) Introduce an IOC for the request for ML inference emulation which shall capture the consumer's requirements for inference emulation. This may be named as an `MLInferenceEmulationRequest` to be name-contained by the `MLInferenceEmulationFunction`.

The `MLInferenceEmulationRequest` shall be associated with at least 1 `MLEntity` for which the inference emulation is being executed.

This IOC contains attributes including the following:

- identifier of the ML entities requested for emulation;
- identifier of the selected emulation environment;
- time window for the emulation.

Each `MLInferenceEmulationRequest` may have a `RequestStatus` field that is used to track the status of the specific `MLInferenceEmulationRequest` or the associated `MLInferenceEmulationProcess`. The `RequestStatus` is an enumeration with the possible values as: "Pending" if no action has been taken for `the` request; "Triggered" when an `MLInferenceEmulationProcess` has been instantiated; "Suspended" when the request or its job has been suspended by MnS consumer or producer and "Served" when the job has run to completion.

4) Introduce an IOC for the process of ML inference emulation from the objects for inference emulation shall be instantiated. This may be named as an `MLInferenceEmulationProcess` to be name-contained by the `MLInferenceEmulationfunction`.

   The `MLInferenceEmulationProcess` shall be associated with at least one `MLEntity` for which the inference emulation is being executed.

   This IOC contains the following attributes:

   - progress indicator;
   - identifier of the corresponding emulation request.

5) Introduce a report on ML inference emulation, which may provide reporting on ML emulation for one or more `MLInferenceEmulationRequests or MLInferenceEmulationProcesses`. This report may be equivalent to the inference report, and:

   - the ML inference emulation report may indicate the emulation environments in which the entity has been emulated and passed;
   - the ML inference emulation report may also include the specific performance metrics for that emulation environments.

   May also introduce the IOCs and datatypes for request, process and report on ML Inference and then or update these IOCs with the features of the ML Inference Emulation as introduced above.

6) The IOCs, attributes and performance measurements for the solutions of performance evaluation for AI/ML inference for inference phase, as described in clause 5.2.6.4, which can be reused for monitoring the inference performance of the ML entities during the emulation.

7) The IOCs and attributes for configuration management of 5G system, as defined in 3GPP TS 28.541 [20], which can be reused for configuring the emulation environment.

5.2.5.5 Evaluation

The solution described in clause 5.2.5.4 reuses the existing provisioning MnS operations and notifications in combination with extensions of the NRM. Requests for inference emulation for a given ML entity may be instantiated using provisioning management service implemented via CRUD (Create, Read, Update, Delete) operations on the request objects. The solution provides the flexibility to allow any function that can execute n ML entity to be the MnS producer for ML inference emulation, e.g. an inference function or a generic sandbox function.

Therefore, the solution described in clause 5.2.5.4 is a feasible solution to be developed further in the normative specifications.

5.2.6 Performance evaluation for AI/ML inference

5.2.6.1 Description

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

5.2.6.2 Use cases

5.2.6.2.1 AI/ML performance evaluation in inference phase

In the inference phase, the inference function (including MDAF, NWDAF and RAN intelligence functions) uses one or more ML entities for inference and generates the inference output.

In the inference phase, the performance of a running ML entity may degrade over time due to changes in network state, which will affect the related network performance and service. Thus, it is necessary to evaluate performance of the ML entity during the inference process. If the inference output is executed, the network performance related to each inference function also needs to be evaluated.

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

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

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

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

To monitor the performance in the inference phase, the MnS producer responsible for 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. The 3GPP management system needs to provide the capability to allow the performance evaluation process to be configured. In addition, an authorized consumer may want to know if the ML entity is working well and request the performance evaluation results of a specific ML model. Accordingly, the performance evaluation results reporting related period or threshold can also be configured.

5.2.6.2.2 ML entity performance indicators query and selection for AI/ML inference

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

The AI/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. MnS producer for ML inference can be queried to provide the information on supported performance indicators referring to ML entity inference phase. Such performance indicators in inference phase may be confidence. Based on supported performance indicators in inference phase as well as based on consumer's requirements, the MnS consumer for ML inference 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 inference to query the supported performance indicators and select a sub-set of performance indicators to be used for performance evaluation.

5.2.6.2.3 ML entity performance indicators selection based on MnS consumer policy for AI/ML inference

ML entity performance evaluation and management is needed during inference phase. The related performance indicators need to be collected and analysed. The MnS producer for inference 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 inference 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 inference of/from the ML entity by the ML MnS Producer for ML inference. The ML MnS consumer for inference may not be capable enough to indicate the performance metrics. Instead, the AI/ML MnS consumer may indicate the requirement using a policy or guidance that reflects the preferred performance characteristics of the ML entity. Based on the indicated policy/guidance, the AI/ML MnS producer may then deduce and apply the appropriate performance indicators for inference. Management capabilities are needed to enable the AI/ML MnS consumer for inference to indicate the behavioural and performance policy/guidance that may be transformed by the MnS producer to a technical performance indicator during inference.

5.2.6.2.4 AI/ML abstract performance

The AI/ML inference MnS consumer is typically interested in understanding the performance of a given AI/ML inference instance, but it is not guaranteed that the MnS consumer understands the applicable AI/ML performance metrics, i.e. it is not always the case that the AI/ML MnS consumer is able to interpret the various metrics on performance KPIs (accuracy, confidence etc) speed, computational resource usage, etc. Relatedly, it may be necessary to provide means to abstract the measured metrics into indices that can be standardized, as such can be easily interpreted by any MnS consumer of AI/ML-related performance management. Thereby, the AI/ML inference function can request for qualification and abstraction of its performance by which a report is generated indicating the qualified abstract performance. Relatedly, an AI/ML inference MnS consumer can request the AI/ML inference MnS producer or the performance abstraction MnS producer for the abstract performance of a specific ML entity or AI/ML inference function. This allows the MnS consumer to interpret the performance even without knowing the details of the specific applicable metrics.

5.2.6.3 Potential requirements

REQ- AI/ML_PERF-INF-1: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to get the inference output provided by an inference function (MDAF, NWDAF or RAN intelligence function).

REQ- AI/ML_PERF-INF-2: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to provide feedback about an inference output.

REQ- AI/ML_PERF-INF-3: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to be informed about the actions taken that were triggered by the inference output provided by an inference function (MDAF, NWDAF or RAN intelligence function).

REQ- AI/ML_PERF-INF-4: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to collect the performance data related to an inference function (MDAF, NWDAF or RAN intelligence function).

REQ- AI/ML_PERF-INF-5: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to collect the performance data for evaluating the performance of an ML entity during inference.

REQ-AI/ML_PERF-INF-6: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to request the performance evaluation results of a specific ML entity.

REQ-AI/ML_PERF-INF-7: The MnS producer responsible for AI/ML inference performance management should have a capability to allow an authorized consumer to control the ML entity performance evaluation functionality.

REQ-AI/ML_PERF -SEL-1: The MLT MnS producer should have a capability allowing the authorized MnS consumer to discover supported AI/ML performance indicators related to AI/ML inference and select some the desired indicators based on the MnS consumer's requirements.

REQ-AI/ML_PERF-POL-1: The AI/ML MnS producer should have a capability allowing the authorized MnS consumer to indicate a performance policy related to AI/ML inference phase.

REQ-AI/ML_PERF-ABS-1: The 3GPP management system should have a capability for an authorized MnS consumer (e.g. an operator) to configure an abstract performance range that defines the minimum and maximum performance as expressed on an abstract performance index.

REQ-AI/ML_PERF-ABS-2: The 3GPP management system should have a capability for an authorized MnS consumer (e.g. the producer of AI/ML services such as the producer of ML training or AI/ML inference services) to request the MnS producer to abstract and qualify one or more ML performance metrics of one or more specific ML entities.

REQ-AI/ML_PERF-ABS-3: The 3GPP management system should have a capability for an authorized MnS consumer (e.g. the MnS consumer of AI/ML services such as the MnS consumer of ML training) to request the abstract performance of one or more specific ML entities.

REQ-AI/ML_PERF-ABS-4: The 3GPP management system should have a capability for an authorized MnS consumer (e.g. the operator) to request the MnS producer to provide the abstract performance as a performance abstraction report of one or more ML performance metrics of one or more ML entities.

5.2.6.4 Possible solutions

5.2.6.4.1 Possible solutions for AI/ML performance evaluation in inference phase

This solution comprises of the following aspects:

- For getting the inference output, the MDA MnS (see 3GPP TS 28.104 [2]) already supports MDA reporting by notifications, file and data streaming. The same approach can be applied to reporting other kinds of inference output (NWDAF analytics report, RAN intelligence output). A common data format may be defined for all kinds of inference outputs, and the format will be decided in normative phase.
- For providing the feedback about the inference output, the IOC representing the feedback, for example named as `InferenceFeedback`, can be used to allow the MnS consumer to create an instance on the producer. This IOC contains the following attributes:
    - inference report id;
    - indication of whether there are actions to be taken triggered by the inference report;
    - feedback for the inference report, e.g. lack of confidence or accuracy for a specific output information element.
- For being informed about the actions taken trigged by the inference output, the NRM notification representing the already taken actions triggered by the inference is used. For example defining a new IOC named as `ActionsTriggeredByInferenceOutput`, or enhancing the existing notifications for the NRMs. The notification contains the following information:
    - inference report id that triggers the action;
    - actions taken (this information is already supported when enhancing the existing notifications).
- For monitoring the network performance related to each inference function, the performance measurements related to each inference function need to be defined to allow the MnS consumer to collect:
    - For the performance measurements related to MDAF, the performance measurements listed in the analytics enabling data for each MDA capability can be used for performance evaluation of MDAF (see 3GPP TS 28.104 [2]).
    - For the performance measurements related to NWDAF, the studies are described in 3GPP TR 28.864 [6].

- For the performance data related to RAN intelligence functions, including RAN intelligence ES function, RAN intelligence MRO function, RAN intelligence MLB function, the MDT data and following performance measurements for MRO, Energy Efficiency and MLB respectively can be reused:
    - for RAN intelligence ES function, the measurements related to distributed energy saving (see clause 6.2.3.1.3.2 of 3GPP TS 28.310 [7], 3GPP TS 28.552[8]) for NG-RAN can be reused;
    - for RAN intelligence MRO function, the measurements related to D-MRO (see clauses 7.1.2.3.1 and 7.1.6.3.1 of 3GPP TS 28.313 [9], 3GPP TS 28.552 [8]) can be reused;
    - for RAN intelligence MLB function, the measurements related to D-MLB (see clauses 7.1.5.3.1 of 3GPP TS 28.313 [9], TS 28.552 [8]) can be reused.

- For controlling the performance evaluation process, the IOC representing the evaluation control information, for example named as `EvaluationControl`, can be used to allow the MnS consumer to control the performance evaluation capability of the producer. This IOC contains the following attributes:
    - identifier of the ML entity to be evaluated;
    - indication of selected performance indicator;
    - indication of performance evaluation period;
    - indication of reporting condition, e.g. reporting period, performance threshold.

5.2.6.4.2 Possible solutions for ML entity performance indicators query and selection for AI/ML inference

The solutions given in clause 5.1.10.4.3 are applicable for ML entity performance indicators query and selection for AI/ML inference.

5.2.6.4.3 Possible solutions for policy-based performance indicator selection based on MnS consumer policy for AI/ML inference

The solutions given in clause 5.1.10.4.4 are applicable for ML entity performance indicators selection based on MnS consumer policy for AI/ML inference.

5.2.6.4.4 Possible solutions for AI/ML performance abstraction

Introduce an IOC for AI/ML performance abstraction as the entity that is the producer of AI/ML performance abstraction and supports all the related services for request and delivery of qualified ML performance Abstraction. The IOC may be named `MLPerformanceAbstraction`.

`MLPerformanceAbstraction` may be name-contained in either a `Subnetwork`, a `ManagedFunction` or a `ManagementFunction`:

- The `MLPerformanceAbstraction` receives a request for the qualification and abstraction of one or more ML Performance metric(s) of a specific ML entity.

  The request might be an IOC and may be named `MLPerfQualRequest`.

- The request may contain the raw metrics (Confusion Matrix, Precision and Recall, F1-score, AU-ROC, etc.) or the input(s) and the expected output(s) of the stated ML entity for which performance abstraction is desired.

- For each request, the `MLPerformanceAbstraction` provides a response that contains the report on the qualified abstract performance. The report might be named `MLAbstractPerfReport`.

Abstraction of ML Performance

An IOC is introduced to support ML performance abstraction. It might be named `mlPerformanceIndex.` The `mlPerformanceIndex` has a pre-defined index range that specifies the absolute minimum and maximum performance. It is introduced as an attribute to the `mlPerformanceIndex` and might be named `mlPerformanceIndexRange`:

- The `mlPerformanceIndexRange` is standardized and known by both the consumers and the producers of AI/ML services and may be applied for different performance metrics.
- For each performance metric, the performance abstraction producer should map the specific performance value to the predefined `mlPerformanceIndexRange` to generate the specific `mlAbstractPerfIndex` value for that performance metric value. This can then be communicated to the consumers, who do not need to know the original performance metric value or its interpretation but can still make sense of the achieved performance.
- The `mlPerformanceIndex` may be computed based on only one performance metric. However, an aggregate index may also be computed for a combination of multiple performance metrics, to generate the specific `mlAggregatePerfIndex` value.

Requesting and Reporting on ML Performance Abstraction

The `MLPerformanceAbstraction` has the capability to compute an abstraction of the performance of a given ML entity given the achieved performance of the ML entity on the specific metrics. A `mlPerformanceIndexRange` is configured onto the `MLPerformanceAbstraction` to indicate the fixed range on which all performances are to be mapped:

- For each request to abstract and qualify the performance of the a given ML entity, an MnS consumer creates a new request, might be named `MLPerfQualRequest`, on the `MLPerformanceAbstraction`, i.e. `MLPerfQualRequest` should be an IOC that is instantiated for each request to abstract and qualify performance.
- Any request for qualifying and abstracting performance state the following:
  - `mLFunctionID`: the identifier of the specific AI/ML inference function the MnS consumer wishes to have performance qualified and abstracted. In some cases, the request may be submitted by the network function having ML capabilities itself, in such a case the network function submits its own DN.
  - `mLEntityId`: The request may optionally state the identifier of the specific `ML entity` for which the MnS consumer wishes to have performance qualified and abstracted.
  - `mlPerformanceMetrics`: The request indicates the specific one or more ML-related performance metrics and their values that should be evaluated by the `MLPerformanceAbstraction` for generating the abstract performance index.
- Following the request, the `MLPerformanceAbstraction` computes the `mlPerformanceIndex` as the abstraction of the performance metric values as fitted to the specified `mlPerformanceIndexRange`.

  For the computed `mlPerformanceIndex,` the `MLPerformanceAbstraction` compiles report containing the computed `mlPerformanceIndex`. Then it forwards it to the MnS consumer (the function that requested for the performance abstraction) to notify the MnS consumer about the outcomes of the performance abstraction. Subsequent to reporting the `MLPerformanceAbstraction` may also publish the abstract performance to some shared publication space. The report is a data type and might be named `MLAbstractPerfReport`.

5.2.6.5 Evaluation

The solutions described in clause 5.2.6.4.1:

- reuses already defined MDA reporting mechanisms for inference output reporting, which may require some minimal change to make the solution (e.g. reporting format) applicable to all kinds of inference functions;

- uses NRM based solution for providing the feedback and informing the taken actions, which makes the new and existing NRMs can be easily and clearly correlated;
- reuses the existing performance measurements for monitoring the network performance related to each inference function.

Therefore, the solutions described in clause 5.2.6.4.1 are feasible.

The solution described in clause 5.2.6.4.4 reuses the existing provisioning MnS operations and notifications in combination with extensions of the NRM. Indeed, requests for qualifying and abstracting performance of ML training, AI/ML inference function or an ML entity may be instantiated using provisioning Management service implemented via CRUD (Create, Read, Update, Delete) operations on the request objects. The solution provides the flexibility to allow any function to be the MnS producer for ML performance abstraction, e.g. the training function or the inference function. It also allows any function that utilizes ML related results to consume that resulting report for the performance abstraction.

Therefore, the solution described in clause 5.2.6.4.4 is a feasible solution to be developed further in the normative specifications.

5.2.7 Configuration management for AI/ML inference phase

5.2.7.1 Description

The AI/ML inference function (e.g. NG-RAN intelligence ES function as described in 3GPP TR 37.817 [15]) may use the ML entity for inference.

The AI/ML inference function needs to be configured (e.g. with policies, targets, conditions where applicable) in order to conduct inference in the 5G system aligning with the consumer´s expectation.

To enable the AI/ML inference function to perform inference using the preferred ML entity, the relevant ML entity needs to be able to be activated and deactivated.

As described in clause 4.7 in 3GPP TR 28.813 [3], RAN domain ES can use AI to formulate energy saving solutions. Therefore, the ML entities which enabled RAN domain ES function should be controlled by 3GPP management system. The ML entity configuration needs to be triggered to enable RAN domain ES function.

The AI/ML configuration can be initiated by the MnS consumer or initiated by the MnS producer.

The following aspects are described for AI/ML configuration:

- Configuration for AI/ML inference function.
- Configuration for ML entity for RAN domain ES function.
- Activation for AI/ML inference capabilities on ML entities and inference functions.

5.2.7.2 Use cases

5.2.7.2.1 ML entity configuration for RAN domain ES initiated by consumer

The ML entity configuration may be initiated by the AI/ML MnS consumer of Cross domain management. AI/ML MnS Consumer monitor network performance and determine whether to trigger the ML entity configuration. For example, for ES purpose, AI/ML MnS Consumer collects the information of the capacity booster cells and coverage cells inside the RAN domain area, then makes the decision for activation ML entity. The AI/ML MnS Consumer may configure policies for activation/deactivation of the ML entity.

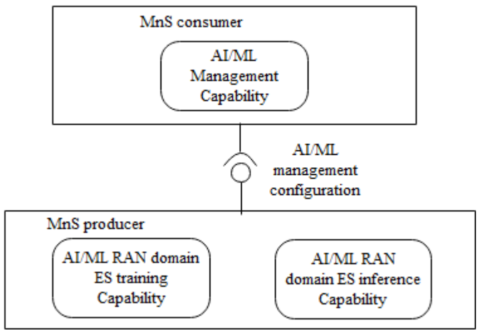

MnS consumer
AI/ML Management Capability
AI/ML management configuration
MnS producer
AI/ML RAN domain ES training Capability
AI/ML RAN domain ES inference Capability

Figure 5.2.7.2.1-1: ML entity configuration initiated by MnS consumer

5.2.7.2.2 ML entity configuration for RAN domain ES initiated by producer

The ML entity configuration may be initiated by the AI/ML MnS producer. AI/ML MnS producer can determine` whether to trigger ML entity configuration based on network performance and service requirements. In this case, the AI/ML MnS producer responsible for AI/ML management needs to have a capability to trigger the ML entities and inform an authorized AI/ML MnS consumer about the ML entity status.

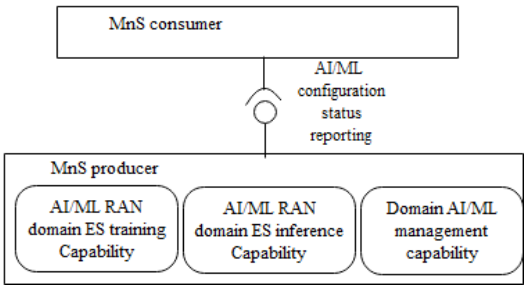

MnS consumer
AI/ML configuration status reporting
MnS producer
AI/ML RAN domain ES training Capability
AI/ML RAN domain ES inference Capability
Domain AI/ML management capability

Figure 5.2.7.2.2-1: ML entity configuration initiated by producer

5.2.7.2.3 Partial activation of AI/ML inference capabilities

An ML entity may provide the AI/ML inference capabilities for a scope (e.g. a specific list of NR cells) of the radio coverage area as either of a decision-making capability or an analysis capability. For a given AI/ML inference function, it can be very difficult to accurately "predict" or quantify the benefits of using an ML entity or an inference capability for the ML entity or inference function in a given context of operational system, before using it.

Furthermore, it is also necessary to ensure that AI/ML inference capabilities of an ML entity or an inference function that are being activated in operational system will bring the expected/planned benefits and will not further downgrade the existing network performance. Moreover, it is important to provide means to check which particular AI/ML inference capabilities of an ML entity or an inference function are beneficial to be activated in a given context of operational network. Correspondingly, the MnS producer for AI/ML inference management may provide different steps through which the capabilities of an ML entity or inference function may be activated progressively. This abstraction phased activation of the scope of the ML entities may be referred to as "Abstract activation steps". For example, with such Abstract activation steps technique, the producer may support a capability to allow only a sub-scope to be activated e.g. to only allow inference activation for a limited or specific number of cells covering part of a geographical coverage area and not the whole city or only for a certain limited period of time (say between 18:00 and 6:00) rather than for the entire operation time.

Another approach to implement partial or progressive activation of AI/ML inference capabilities for an ML entity or an inference function would be through a predefined policy which may include e.g. time scheduled or conditional progressive or phased activation of the inference capabilities or the scope of activation.

So, it is possible that the AI/ML inference function is configured to start using a newly deployed ML entity for one part (e.g. one NR cell of the gNB) of the function but the existing ML entity for the rest parts, and then gradually switch to use the new ML entities for the larger or full scope, by activating/deactivating the AI/ML inference capabilities in the corresponding scope for the ML entities.

Together, these imply that it is important to ensure that the AI/ML MnS consumer has a finer control on activation and de-activation of AI/ML inference capabilities for an ML entity or an inference function.

5.2.7.2.4 Configuration for AI/ML inference initiated by MnS consumer

The MnS consumer monitors the network performance and determines on whether to, and when to trigger the AI/ML inference configuration or re-configuration. For example, for NG-RAN intelligence ES function (as described in 3GPP TR 37.817 [b]), the MnS consumer collects the performance data of the capacity booster cells and coverage cells, then makes the decision for configuring or re-configuring the inference function with a policy may include e.g. performance targets of the inference function, or the activation/deactivation of the ES function and/or the associated ML entities.

In this case, the MnS consumer may need to initiate the AI/ML inference configuration/reconfiguration.

The inference function has a set of configurable attributes whose values can be changed by an MnS consumer. As such the MnS consumer may set the values of these inference function configuration attributes. Note that inference function configuration attributes are different from Network Configuration Parameters (NCPs), which are the actual network parameters whose values are set by the inference function. Changes in inference function configuration attributes values translate into changes in the behaviour of the inference function but not necessarily in the instantaneous behaviour of the network. An example inference function configuration attribute is the maximum allowed value to which a configuration parameter may be set, e.g. the maximum value to which or by which a cell individual offset may be adjusted.

5.2.7.2.5 Configuration for AI/ML inference selected by producer

The MnS producer monitors the network performance and determines on whether to, and when to trigger the AI/ML inference configuration or re-configuration. For example, NG-RAN intelligence ES function (as described in 3GPP TR 37.817 [15]), per the performance of the energy efficiency result by execution of the inference output, the MnS producer may decide to activate or deactivate the inference function, or decide to use another ML entity for inference. In this case, the MnS producer may initiate the configuration and inform an authorized MnS consumer about the configurations. The configuration actions conducted by the MnS producer may also be triggered by a predefined configuration policy.

Inference functions are characterized by contexts e.g. via the associated ML entities (see 3GPP TS 28.105 [4] and clause 5.1.7 ML context). Networks (and thus inference functions) will have several contexts which all cannot be addressed by a single configuration setting. To enable, the MnS producer to monitor the network performance and determines on whether to, and when to trigger the AI/ML inference configuration or re-configuration, the MnS producer should be configured with objectives based on which the MnS producer can configure its attributes.

5.2.7.2.6 Enabling policy-based activation of AI/ML capabilities

If the activation procedure is entirely relying on the AI/ML MnS consumer to micro-manage every activation step, such process may require extensive signalling between the AI/ML MnS consumer and producer and intrinsically lacks the automation potential. On the other hand, the activation procedure cannot be left fully to the producer either, as the producer may not have a "full picture" on other ML entities/capabilities that are currently in operation, activated by different producers on the request from MnS consumer. The producer needs to be instructed by the MnS consumer on the ways to perform the adequate activation of AI/ML capabilities.

The activation may be instructed via one or more AI/ML activation policies, where an AI/ML activation policy is a sequence of tuples of conditions and activation settings that may be executed by the AI/ML producer. Conditions may define specific outcomes on performance metrics for which a particular activation may be executed while activation settings define specific attributes of the AI/ML capability activation scope (e.g. object or object type, network context, activation time window) for which AI/ML capability should be activated.

5.2.7.3 Potential requirements

REQ-AIML_INF_CFG- -1: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to configure the inference function.

REQ-AIML_INF_CFG -2: The MnS producer responsible for AI/ML inference management should have a capability to configure inference function and inform an authorized MnS consumer about the configurations of the AI/ML inference function.

REQ-AIML_INF_ACT-1: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to activate an AI/ML inference function.

REQ-AIML_INF_ACT-2: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to deactivate an AI/ML AI/ML inference function.

REQ-AIML_INF_ACT-3: The MnS producer responsible for AI/ML inference management should have a capability to inform an authorized MnS consumer about the activation and deactivation of an AI/ML inference function.

REQ-AIML_INF_ACT-4: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to partially or progressively activate/deactivate the AI/ML inference capabilities for an inference function.

REQ-ML_ENTITY_ACT-1: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to activate an ML entity.

REQ-ML_ENTITY_ACT-2: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to deactivate an ML entity.

REQ-ML_ENTITY_ACT-3: The MnS producer responsible for AI/ML inference management should have a capability to inform an authorized MnS consumer about the activation and deactivation of an ML entity.

REQ-ML_ENTITY_ACT-4: The MnS producer responsible for AI/ML inference management should have a capability to allow an authorized MnS consumer to partially or progressively activate/deactivate the AI/ML inference capabilities for an ML entity.

REQ-ML_ENTITY_ACT-5: The 3GPP management system should have a capability to allow an authorized MnS consumer to define the policies for activation of AI/ML capabilities in order to instruct the AI/ML MnS producer on how to perform the AI/ML capability activation (e.g. when and where to activate which AI/ML capabilities).

REQ-ML_ENTITY_ACT-6: the 3GPP management system should have a capability to allow a producer to activate the AI/ML capabilities based on the policies specified by the AI/ML MnS consumer.

5.2.7.4 Possible solutions

5.2.7.4.1 AI/ML inference function configuration

5.2.7.4.1.1 Configuration for AI/ML inference initiated by MnS consumer

No new IOCs, or data types are needed to enable (re-)configuration of inference function, but some attributes (e.g. the allowed range or maximum), need to be configurable (i.e. writable) by the authorized MnS consumer.

5.2.7.4.1.2 Configuration for AI/ML inference selected by producer - Context-specific configuration

Introduce a datatype for context specific configuration of inference functions objectives, e.g. called objectiveModel. The datatype captures the desired targets and their prioritization for the specific AI/ML inference function. Example entries may show the configuration for an inference function responsible for optimizing energy consumption which match the location/area and time of optimization with the amount of energy consumed. These entries of the objectiveModel are:

- **IF NOT** location = urban **THEN** energy consumption < 40% **WITH** priority 0.5
- **IF NOT** timeWindow in [08:00, 17:59] **THEN** energy consumption < 50% **WITH** priority 0.1}

  Where:

  - **L**ocation indicates a type of geographical environment. It may be modelled as an enumeration of different types of geographical environments including among other "an urban environment", "a rural environment ", "a peri-urban environment ", "a highway environment", etc.
  - TimeWindow indicates a range of time. It may be modelled as a timeWindow.

NOTE: The inference function is assumed to have a capability for a context derivation function with which to match the NCPs with the appropriate contexts, derives the network context space relevant to the inference function (also called the function's context model) and then derives the context-specific configurations for the functions or network.

The MnS producer may be configured to have access to a pre-defined template corresponding to network context.

5.2.7.4.2 AI/ML activation

5.2.7.4.2.1 General framework for activation and deactivation

This subclause describes the general framework for activation and deactivation of AI/ML inference capabilities, ML entities and inference function in inference phase.

A data type or abstract class describing the activation properties, and this data type or abstract class can be used or inherited by the MOI representing the inference function and ML entity.

This general framework supports the general properties for all types of activation/deactivation including:

- Activation type: which can be instant activation/deactivation, Policy-based activation and deactivation, scheduled-based activation/deactivation, Gradual activation and deactivation:
    - Instant activation and deactivation: The AI/ML inference capabilities to be instantly activated/deactivated on the ML entity or inference function.
    - Policy-based activation and deactivation: The AI/ML inference capabilities to be activated/deactivated on the ML entity or inference function based on a given policy.
    - Schedule-based activation and deactivation: The AI/ML inference capabilities to be activated/deactivated on the ML entity or inference function based on a given schedule.
    - Gradual activation and deactivation: The AI/ML inference capabilities to be activated/deactivated on the ML entity or inference function based on a given scope.

And this data type or abstract class is extended with the attributes supporting these specific types of activation.

5.2.7.4.2.2 Instant activation and deactivation

The generic framework described in clause 5.10.4.2.1 is extended with the following attributes to support instant activation and deactivation:

- The AI/ML inference capabilities to be instantly activated/deactivated on the ML entity or inference function.

5.2.7.4.2.3 Policy based activation and deactivation

The generic framework described in clause 5.10.4.3.1 is extended with the following attributes to support the policy-based activation and deactivation:

- AI/ML inference capabilities to be activated/deactivation on the ML entity or inference function based on the given policy.
- The policy (e.g. condition) for activation/deactivation.

5.2.7.4.2.4 Schedule based activation and deactivation

The generic framework described in clause 5.10.4.3.1 is extended with the following attributes to support the schedule-based activation and deactivation:

- AI/ML inference capabilities to be activated/deactivation on the ML entity or inference function based on the given schedule.
- The schedule for activation/deactivation.

5.2.7.4.2.5 Gradual activation and deactivation

This solution extends the general framework for activation to support gradual/partial/progressive activation.

Multiple options may be considered for the solutions:

1) Using Activation attributes on the NRM

Introduce a <<`datatype`>> attribute for partial activation (say called "partialActivation") in the `MLEntity` or its function. This can have two <<`datatype` >> attributes - an "activationScope and an "activationLevel".

The activationScope specifies the information on particular network scope and AI/ML capabilities to be activated. It is configured by the MnS consumer to limit the activation of selected AI/ML capabilities to the desired extent. The scope may include:

- information on the network context, e.g. specific RATs and the object(s) or object types for which the AI/ML capability is applicable;
- information on the subscope of the applicable expectedRuntimeContext which may include at least one or combination of the following:
  - object subscope - identifying a subset of the objects with respect to which a certain AI/ML capability should be activated;
  - network characteristics (related to the stated object or object types) for which the `MLEntity` produces analytics;
  - control parameter sub scope - identifying a subset of the parameters of the stated object or object types which the `MLEntity` optimizes or controls and for which ten a certain AI/ML capability should be activated;
  - metric sub scope - identifying a subset of the network metrics which the `MLEntity` optimizes through its actions for which then a certain AI/ML capability should be activated.

The activationScope is explicitly stated by the MnS consumer for the desired scope and subscope.

Following the activation, a notification may be provided, e.g. via a `MLGradualActivationResponse <<datatype>>` that represents the response upon partial or gradual activation of `MLEntity`. This IOC is created by the MnS producer and reported to the MnS consumer, and it contains the following attributes:

- `MLEntity` ID - identifier of the ML entity to which the gradual activation applies;
- status, e.g. activated/deactivated;
- information on particular AI/ML capabilities that have been activated;
- scope under which particular AI/ML capabilities have been activated.

2) Using abstractActivtion levels on the NRM

Introduce a data type on the MnS producer that exposes the abstract activation levels supported by the MnS producer. These may be contained in a datatype called `SupportedMLActivationLevels` which is a list of candidate levels. Each entry in the list is of <<`datatype` >> `MLActivationLevel` a << `datatype` >> representing an individual step in which the activation (or de-activation) can be performed at the MnS Producer.

The `MLActivationLevel` contains the following attributes:

- identifier of the abstracted activation level, e.g. low, medium, high;
- information on (the set of) AI/ML capabilities to be activated (or de-activated) for a given abstracted activation level;
- information on the scope under which the given AI/ML capabilities will be activated (or de-activated) for a given abstracted activation level.

Introduce an attribute for a selected activation level. This may be termed as SelectedActivationLevel - this is an enumeration of the Identifiers of the abstracted activation level which can be configured by the MnS consumer to select the preferred activation level.

5.2.7.5 Evaluation

The solutions described in clause 5.2.7.4 is a fully NRM-based approach and reuses the existing provisioning MnS operation for AI/ML inference configuration. This approach supports both MnS consumer-initiated and MnS producer-initiated configuration based on the existing provisioning MnS operations and notifications. It enables a versatile activation of AI/ML capabilities. This provides the means to better control the usage of AI/ML capabilities in the network using the consistent NRM-based approach.

Therefore, the solution described in clause 5.2.7.4 is a feasible solution.

5.2.8 AI/ML update control

5.2.8.1 Description

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

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

5.2.8.2 Use cases

5.2.8.2.1 Availability of new capabilities or ML entities

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

5.2.8.2.2 Triggering ML entity update

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

The AI/ML inference MnS consumer needs to request the AI/ML inference MnS producer to update its capabilities or its constituent ML entities and the AI/ML MnS producer should respond accordingly. For example, the AI/ML inference MnS producer may download new software that supports the required updates, download from a remote server a file containing configurations and parameters to update one or more of its constituent ML entities, or it may trigger one or more remote or local AI/ML-related processes (including training/re-training, testing, etc.) needed to generate the required updates. Relatedly, 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 to adjust the update conditions or characteristics, e.g. the times when the update may be executed.

5.2.8.3 Potential requirements

REQ-AIML_UPDATE-1: The 3GPP management system should have a capability for the AI/ML inference MnS producer to inform an authorized MnS consumer of the availability of AI/ML capabilities or ML entities or versions thereof (e.g. as learned through a training process or as provided via a software update) and the readiness to update those AI/ML capabilities when requested.

REQ-AIML_UPDATE-2: The 3GPP management system should have a capability for the AI/ML inference MnS producer to inform an authorized MnS consumer of the expected performance gain if/when the AI/ML capabilities or ML entities of the respective network function are updated with/to the specific set of newly available AI/ML capabilities.

REQ-AIML_UPDATE-3: The 3GPP management system should have a capability to allow an authorized MnS consumer to request the AI/ML inference MnS producer to update its constituent ML entities using a specific version of newly available AI/ML capabilities or ML entities or by using AI/ML capabilities or ML entities with specific performance characteristics or gains.

REQ-AIML_UPDATE-4: The 3GPP management system should have a capability for the AI/ML inference MnS producer to inform an authorized MnS consumer about the process or outcomes related to any request for updating the AI/ML capabilities or ML entities.

REQ-AIML_UPDATE-5: The 3GPP management system should have a capability for the AI/ML inference MnS producer to inform an authorized MnS consumer about of the achieved performance gain following the update of the AI/ML capabilities with/to the specific newly available ML entities or set of AI/ML capabilities.

REQ-AIML_UPDATE-6: The 3GPP management system should have a capability for an authorized AI/ML inference MnS consumer (e.g. an operator or the function/entity that generated the request for updating the AI/ML capabilities) to manage the request and subsequent process, e.g. to suspend, re-activate or cancel the request or to further adjust the characteristics of the capability update.

5.2.8.4 Possible solutions

Introduce an IOC to model the AI/ML updating capabilities, it might be named `MLupdate`.

- Following a request by the AI/ML MnS consumer, the `MLupdate` creates an instance for the request at the producer and may update that instance with status related to that request. The request IOC might be named `MLupdateRequest`. `MLupdate` may also notify the MnS consumer who initiated the request of the corresponding action taken regarding the request, e.g. an associated ML update job has been instantiated for the updating:
    - It is possible, that the new updates are already available before the request, e.g. as learned through a reinforcement learning process. The consumer may be notified that new ML entities for update is available, and the notification may also include the information of the available ML entities. In such a case, the request for update may follow an indication that there are available updates and/or the information of the available ML entities for update.
    - The request for updating ML capabilities may state the identifier of the specific `MLEntity` that the consumer wishes to be updated.
    - The request for update may be constrained by specific requirements, i.e. the consumer may request that the update only happens if certain characteristics/update-trigger conditions are fulfilled. e.g. in particular, the `MLupdateRequest` may specify a `performanceGainThreshold` which defines the minimum performance gain that shall be achieved with the update.
    - An update job, say named as `MLupdateJob` can be instantiated by `MLupdate` in response to an `MLupdateRequest`. Alternatively, the `MLupdateJob` can also be instantiated directly by an authorized MnS consumer.
    - A notification for update can be send to the MnS consumer to indicate whether the specific requirements for update is fulfilled or not and also indicate the reason if not fulfilled.

- The AI/ML inference MnS producer may employ any of the other AI/ML management services to fulfil the update. For example, the AI/ML inference MnS producer may trigger a training or re-training process or may trigger a uploading process of new/updated ML entity/entities. The triggered process may provide respective notifications to the `MLupdate` or to any MnS consumer that may wish to be informed that such processes have been undertaken.

5.2.8.5 Evaluation

The `MLupdate` has the capability and a control interface to allow a MnS consumer (e.g. the operator) to configure and manage one or more `MLupdateRequests`. The control interface might for example enable the MnS consumer:

- to get the outcomes of the request using the NotifyMOIattriteChanges operation; or
- to read the characteristics of submitted `MLupdateRequests` using the readMOIattributes operation;
- to configure the submitted `MLupdateRequests`, e.g. the operator may change the priorities of one or more `MLupdateRequests`.

Note that the same operations are also needed for the `MLUpdateJobs`. Thus, the solution described in clause 5.2.8.4 reuses the existing provisioning MnS operations and notifications in combination with extensions of the NRM. And therefore, it is a feasible solution to be developed further in the normative specifications.

5.3 Common management capabilities for ML training and AI/ML inference phase

5.3.1 Trustworthy Machine Learning

5.3.1.1 Description

During ML training, testing and inference, the AI/ML trustworthiness management is needed. Based on the risk level (e.g. unacceptable, high, minimal) of the use case, the trustworthiness requirements for ML training, testing and inference may vary and therefore the related trustworthiness mechanisms need to be configured and monitored. The purpose of AI/ML trustworthiness is to ensure that the model being trained, tested, and deployed is explainable, fair and robust.

NOTE: In the context of SA5, explainability of a model refers to explaining individual decisions predicted by the model and not explaining the internal behavior of the model itself.

The EU has proposed an AI regulation act for AI/ML consisting of several key requirements that the AI/ML systems should meet (based on the risk level of the use case) for them to be considered trustworthy [10]. These requirements include, but not limited to human agency and oversight, technical robustness and safety, privacy and data governance, transparency, diversity, non-discrimination and fairness, accountability, societal and environmental well-being. Other requirements and more details on each of these requirements are described in [11]. Furthermore, ISO/IEC analyses the factors that can impact the trustworthiness of systems providing or using AI and possible approaches or requirements to improving their trustworthiness that can be used by any business regardless of its size or sector [12].

Three well known categories under the umbrella of Trustworthy Machine Learning are as follows:

Explainable Machine Learning: Explainability in machine learning refers to the ability of ML models to enable humans to understand decisions or predictions made by them.

Fair Machine Learning: Fairness in machine learning refers to the process of correcting and eliminating bias in machine learning models.

Robust Machine Learning: Robustness in machine learning refers to the process of handling various forms of errors/corruptions in machine learning models as well as changes in the underlying data distribution in an automatic way.

These features apply to the four aspects of the ML process:

- Data processing for use towards training, testing and inference.
- The training of ML entities.
- The testing of ML entities.
- The use of ML entities for inference.

5.3.1.2 Use cases

5.3.1.2.1 AI/ML trustworthiness indicators

The AI/ML trustworthiness indicators related to ML training, testing and inference need to be precisely defined. The indicators mainly include three aspects:

Explainability-related indicators: the explainability indicators of the ML entity. For example, the AI/ML MnS consumer may indicate to the AI/ML MnS producer to:

- Provide local explanation for one particular instance predicted by the ML entity without disclosing the ML entity internals.
- Provide global explanation for a group of instances predicted by the ML entity without disclosing the ML entity internals.
- Evaluate monotonicity - a quantitative metric for explainability - that measures the effect of individual features on ML entity performance by evaluating the effect on ML entity performance by incrementally adding each feature in order of increasing importance.

Fairness-related indicators: the fairness indicators of the data or the ML entity. For example, the AI/ML MnS consumer may indicate the AI/ML MnS producer to:

- Evaluate disparate impact - a quantitative measure for fairness - that measures the ratio of rate of favourable outcome for the unprivileged group to that of the privileged group.
- Evaluate Manhattan distance - a quantitative measure for fairness - that measures the average distance between the samples from two datasets.
- Evaluate average odds difference - a quantitative measure for fairness - that measures the average difference of false positive rate and true positive rate between unprivileged and privileged groups.

Robustness-related indicators: the robustness indicators of the data or the ML entity. For example, the AI/ML MnS consumer may indicate the AI/ML MnS producer to:

- Evaluate missingness ratio - a quantitative measure for robustness - that measures the percentage of missing values in the training dataset.

Depending on the use case, some or all trustworthiness indicators can be selected for monitoring and evaluation. The AI/ML MnS consumer should first determine which indicators are needed and then request the AI/ML MnS producer to monitor and evaluate the requested indicators.

5.3.1.2.2 AI/ML data trustworthiness

The training data, testing data and inference data used for ML training, testing and inference, respectively, may need to be pre-processed according to the desired trustworthiness measure of the ML model. For example:

- The samples in the training data and testing data can be labelled to include the ground-truth explanation label (in addition to the ground-truth class label). Therefore, the ML model can be trained to predict both ground-truth explanations label and ground-truth class label for an inference sample.
- The samples in the training data and testing data can be assigned weights to ensure individual or group fairness in the ML model.

- The missing features in the training data, testing data and inference data can be imputed with mean values to ensure the ML model is technically robust.
- Noise can be added to the training data and testing data to ensure that the data samples are free from any kind of poisoning attacks.

Depending on the use case, some or all data trustworthiness pre-processing techniques can be applied before training, testing and deployment of the ML model. The MnS consumer should be enabled to receive information on the supported trustworthiness-related data processing capabilities for training, testing or inference. Moreover the producer of data processing be it for training, testing or inference should enable the MnS consumer to provide requirements for trustworthiness which should then be considered in the data processing. And the MnS consumer should be enabled to define their reporting characteristics for ML trustworthiness.

5.3.1.2.3 ML training trustworthiness

The ML training may need to be performed according to the desired trustworthiness measure of the ML model. For example:

- The ML model can be trained to generate explanations for the predictions.
- The ML model can be trained to detect and mitigate biased outcomes.
- The ML model can be trained to perform well on unseen or missing data.
- The ML model can be trained together with adversarial input samples so that the trained model can detect adversaries.

Depending on the use case, one or more training trustworthiness techniques can be applied during training the ML model. Therefore, the ML training producer can be queried to provide information on the supported training trustworthiness capabilities enabling the ML training MnS consumer to request for a subset of supported training trustworthiness characteristics to be configured, measured, and reported.

5.3.1.2.4 AI/ML inference trustworthiness

The AI/ML inference may need to be performed according to the desired trustworthiness measure of the ML model. For example:

- Post-processing explanations can be generated based on one or multiple inferences generated by the ML model.
- The ML model can be trained to flip biased outcomes during inference using post-processing fairness techniques, for e.g. based on confidence value of a prediction.
- The ML model can be trained to infer well on unseen or missing inference data.
- Perturbing model predictions to obfuscate labels/confidence information to protect them from model inversion or model extraction attacks.

Depending on the use case, one or more inference trustworthiness techniques can be applied on the deployed ML model. Therefore, the AI/ML inference producer can be queried to provide information on the supported inference trustworthiness capabilities enabling the AI/ML inference consumer to request for a subset of supported inference trustworthiness characteristics to be configured, measured, and reported.

5.3.1.2.5 Assessment of AI/ML trustworthiness

The ML assessment may need to be performed according to the desired trustworthiness measure of the ML model. For example:

- The ML model can be tested to evaluate the correctness of explanations, quality of explanations, robustness of explanations and adaptiveness of explanations.
- The ML model can be tested to evaluate the robustness of fair predictions and adaptiveness of fair predictions.
- The ML model can be tested to evaluate the correctness of predictions, robustness of predictions and adaptiveness of predictions for both adversarial and non-adversarial test samples.

Depending on the use case, one or more assessment trustworthiness techniques can be applied during assessment the ML model. Therefore, the ML assessment producer can be queried to provide information on the supported assessment trustworthiness capabilities enabling the ML assessment MnS consumer to request for a subset of supported assessment trustworthiness characteristics to be configured, measured, and reported.

5.3.1.3 Potential requirements

REQ-ML_TRUST_IND-1: The AI/ML MnS producer should have a capability to define trustworthiness indicators for AI/ML data or ML entity and select some indicators based on the use case.

REQ-ML_TRUST_IND-2: The AI/ML MnS producer should have a capability to define a common trustworthiness measure covering main aspects of trustworthiness indicators of AI/ML data or ML entity.

REQ-ML_TRUST_IND-3: The AI/ML MnS producer should have a capability to enable the authorized MnS consumer to request for the desired individual or common trustworthiness measure of AI/ML data or ML entity.

REQ-ML_TRUST_IND-4: The AI/ML MnS producer should have a capability to report to the authorized MnS consumer the achieved individual or common trustworthiness measure of AI/ML data or ML entity.

REQ-ML_DATA_TRUST-1: The producer(s) of ML training, ML testing and AI/ML inference service(s) should support a capability to enable an authorized MnS consumer to request reporting on the supported data trustworthiness related pre-processing capabilities of an ML entity.

REQ-ML_DATA_TRUST-2: The producer(s) of ML training, ML testing and AI/ML inference service(s) should have a capability to pre-process the training data, testing data and inference data of an ML entity to satisfy the desired data trustworthiness measure.

REQ-ML_DATA_TRUST-3: The producer(s)of ML training, ML testing and AI/ML inference service(s) should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the data trustworthiness reports of an ML entity.

REQ-ML_TRAIN_TRUST-1: The ML training MnS producer should support a capability to enable an authorized MnS consumer to request reporting on the supported training explainability capabilities of an ML entity.

REQ-ML_TRAIN_TRUST-2: The ML training MnS producer should have a capability to train a specific ML entity using training data with explainability characteristics as defined by the MnS consumer.

REQ-ML_TRAIN_TRUST-3: The ML training MnS producer should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the training explainability reports of an ML entity.

REQ-ML_TRAIN_TRUST-4: The ML training MnS producer should support a capability to enable an authorized MnS consumer to request reporting on the supported training fairness capabilities of an ML entity.

REQ-ML_TRAIN_TRUST-5: The ML training MnS producer should have a capability to train a specific ML entity using training data with fairness characteristics as defined by the MnS consumer.

REQ-ML_TRAIN_TRUST-6: The ML training MnS producer should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the training fairness reports of an ML entity.

REQ-ML_TRAIN_TRUST-7: The ML training MnS producer should support a capability to enable an authorized MnS consumer to request reporting on the supported training robustness capabilities of an ML entity.

REQ-ML_TRAIN_TRUST-8: The ML training MnS producer should have a capability to train a specific ML entity using training data with robustness characteristics as defined by the MnS consumer.

REQ-ML_TRAIN_TRUST-9: The ML training MnS producer should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the training robustness reports of an ML entity.

REQ-ML_INF_TRUST-1: The producer of AI/ML inference should have a capability to infer using a specific ML entity rained and tested with explainability characteristics as defined by the MnS consumer.

REQ-ML_INF_TRUST-2: The producer of AI/ML inference should support a capability for an authorized MnS consumer to define the reporting characteristics related to the inference explainability reports of an ML entity.

REQ-ML_INF_TRUST-3: The producer of AI/ML inference should have a capability to infer using a specific ML entity trained and tested with fairness characteristics as defined by the MnS consumer.

REQ-ML_INF_TRUST-4: The producer of AI/ML inference should support a capability for an authorized MnS consumer to define the reporting characteristics related to the inference fairness reports of an ML entity.

REQ-ML_INF_TRUST-5: The producer of AI/ML inference should have a capability to infer using a specific ML entity trained and tested with robustness characteristics as defined by the MnS consumer.

REQ-ML_INF_TRUST-6: The producer of AI/ML inference should support a capability for an authorized MnS consumer to define the reporting characteristics related to the inference robustness reports of an ML entity.

REQ-ML_TEST_TRUST-1: The ML assessment MnS producer should support a capability to enable an authorized MnS consumer to request reporting on the supported assessment explainability capabilities of an ML entity.

REQ-ML_TEST_TRUST-2: The ML assessment MnS producer should have a capability to test a specific ML entity using assessment data with explainability characteristics as defined by the MnS consumer.

REQ-ML_TEST_TRUST-3: The ML assessment MnS producer should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the assessment explainability reports of an ML entity.

REQ-ML_TEST_TRUST-4: The ML assessment MnS producer should support a capability to enable an authorized MnS consumer to request reporting on the supported assessment fairness capabilities of an ML entity.

REQ-ML_TEST _TRUST-5: The ML assessment MnS producer should have a capability to test a specific ML entity using assessment data with fairness characteristics as defined by the MnS consumer.

REQ-ML_TEST _TRUST-6: The ML assessment MnS producer should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the assessment fairness reports of an ML entity.

REQ-ML_TEST _TRUST-7: The ML assessment MnS producer should support a capability to enable an authorized MnS consumer to request reporting on the supported assessment robustness capabilities of an ML entity.

REQ-ML_TEST _TRUST-8: The ML assessment MnS producer should have a capability to test a specific ML entity using assessment data with robustness characteristics as defined by the MnS consumer.

REQ-ML_TEST _TRUST-9: The ML assessment MnS producer should support a capability to enable an authorized MnS consumer to define the reporting characteristics related to the assessment robustness reports of an ML entity.

5.3.1.4 Possible solutions

5.3.1.4.1 ML trustworthiness indicators

This solution introduces three attributes to specify the trustworthiness indicators:

- `trustworthinessType` indicates the type of trustworthiness metric, e.g. explainability, fairness, robustness.
- `trustworthinessMetric` indicates the trustworthiness metric used to evaluate the trustworthiness of an ML entity, e.g. monotonicity for explainability, disparate impact for fairness, missingness ratio for robustness.
- `trustworthinessScore` indicates the trustworthiness score corresponding to the `trustworthinessMetric`.

The attributes may be combined into a data type `modelTrustworthiness` to specify the trustworthiness of an ML entity or process.

The ML trustworthiness indicators should be applicable to:

- data collection - to ensure that the data is trustworthy, e.g. is not biased against one age or income group;
- training process - to ensure that training is trustworthy, i.e. that even when the data is trustworthy that the manipulation or use of that data is trustworthy, e.g. that one feature is not given unnecessarily more weight in training than another feature. E.g. for ML energy saving does not weigh user density low with a result of a higher switch off of cells in low-income areas where user density is higher. Separate indicators may also be added for the testing process - to ensure that the testing process is trustworthy;

- inference process - to ensure that the inference process is trustworthy.

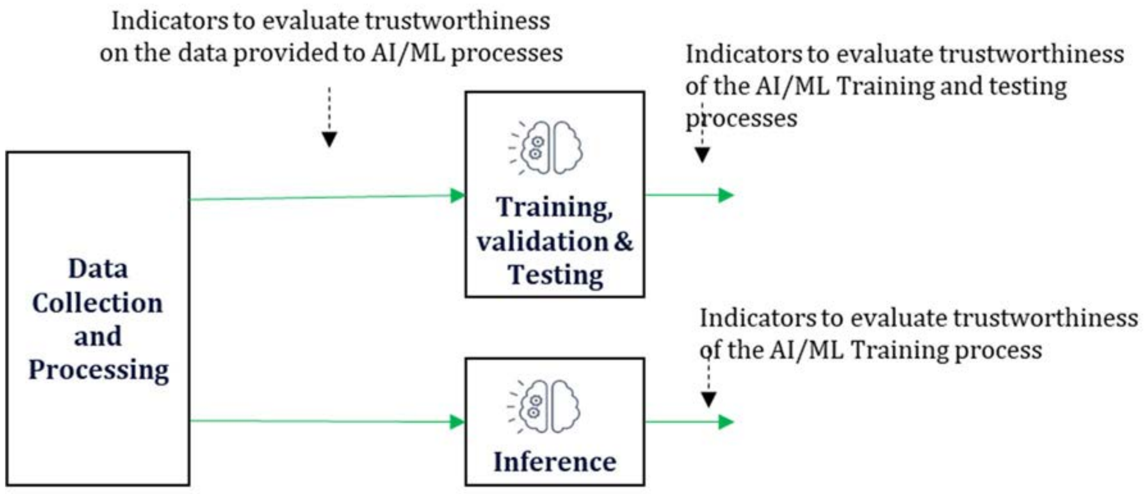

Indicators to evaluate trustworthiness on the data provided to AI/ML processes
Indicators to evaluate trustworthiness of the AI/ML Training and testing processes
Data Collection and Processing
Training, validation & Testing
Indicators to evaluate trustworthiness of the AI/ML Training process
Inference

Figure 5.3.1.4.1-1: ML trustworthiness indicators are applicable to evaluating provision of data as well as the outcomes of the training process, the testing process and the inference process

NOTE 1: The implementation of algorithms to achieve trustworthiness is vendor specific implementation details that are out of the scope of the present document.

NOTE 2: The relation between the trustworthiness indicators/metrics and the 3GPP management or network data needs further investigation.

NOTE 3: How to support the consumer and the producer to have a consistent interpretation of the trustworthiness indicators/metrics is subject to further investigation.

5.3.1.4.2 AI/ML data trustworthiness

This solution extends `MLEntity` to introduce a new attribute of datatype `SupportedMlDataTrustworthiness` indicating on the AI/ML data trustworthiness indicators that can be supported by an MLEntity. This extended `MLEntity` data type can be used to activate the notification on specific AI/ML data trustworthiness indicators based on the request by the authorized consumer.

`SupportedMlDataTrustworthiness` <<dataType>> specifies the data trustworthiness indicator(s), e.g. explainability indicators, fairness indicators, robustness indicators, which can be supported by an `MLEntity`. It contains the tuples of `supportedDataTrustworthinessMetric` and `activatedDataTrustworthinessMetric` attributes.

The `supportedDataTrustworthinessMetric` indicates data trustworthiness metrics that an `MLEntity` is capable of providing. It may be a list of trustworthiness metrics, e.g. monotonicity for explainability, disparate impact for fairness, missingness ratio for robustness, for which those which are supported have a Boolean value of "True". The authorized consumer shall be notified only on a specific subset of such data trustworthiness metrics for which the `activatedDataTrustworthinessMetric` indicator is set.

This solution extends the `MLTrainingRequest` IOC by adding a new attribute, e.g. `dataTrustworthinessRequirements`, to allow the ML entity training MnS consumer to request the ML entity training MnS producer to pre-process the training data with specified data trustworthiness requirements before training the ML model. Similarly, the `MLTrainingReport` IOC also needs to be extended with a new attribute, e.g. `achievedDataTrustworthiness`, to allow the ML entity training MnS producer to report the achieved data trustworthiness score on the training data used to train ML model.

Similarly, if and when ML testing and ML inference related IOCs (e.g. MLTestingRequest, MLTestingReport, MLInferenceRequest, MLInferenceReport) are introduced in 3GPP TS 28.105 [4], these IOCs also need to introduce new attributes, e.g. `dataTrustworthinessRequirements and achievedDataTrustworthiness, but` in the context of testing data and inference data trustworthiness.

The newly introduced attributes `dataTrustworthinessRequirements and achievedDataTrustworthiness` are of data type `modelTrustworthiness.`

5.3.1.4.3 ML training trustworthiness

This solution extends MLEntity to include a new attribute of datatype `SupportedMlTrainingTrustworthiness` indicating on the ML training trustworthiness indicators that can be supported by an `MLEntity`. This extended `MLEntity` data type can be used to activate the notification on specific ML training trustworthiness indicators based on the request by the authorized consumer.

`SupportedMlTrainingTrustworthiness` <<dataType>> specifies the training trustworthiness indicator(s), e.g. explainability indicators, fairness indicators, robustness indicators, which can be supported by an `MLEntity`. It contains the tuples of `supportedMlTrainingTrustworthinessMetric` and `activatedMlTrainingTrustworthinessMetric` attributes.

The `supportedMlTrainingTrustworthinessMetric` indicates training trustworthiness metrics that an `MLEntity` is capable of providing. The authorized consumer shall be notified only on a specific subset of such training trustworthiness metrics for which the `activatedMlTrainingTrustworthinessMetric` indicator is set.

This solution extends the `MLTrainingRequest` IOC by adding a new attribute, e.g. `trainingTrustworthinessRequirements`, to allow the ML entity training MnS consumer to request the ML entity training MnS producer to train the ML model with specified training trustworthiness requirements. Similarly, the `MLTrainingReport` IOC also needs to be extended with a new attribute, e.g. `achievedTrainingTrustworthiness,` to allow the ML entity training MnS producer to report the achieved training trustworthiness score on the ML model.

The newly introduced attributes `trainingTrustworthinessRequirements and achievedTrainingTrustworthiness` are of data type `modelTrustworthiness.`

5.3.1.4.4 AI/ML inference trustworthiness

This solution extends MLEntity to include a new attribute of datatype `SupportedMlInferenceTrustworthiness` indicating on the AI/ML inference trustworthiness indicators that can be supported by an `MLEntity`. This extended `MLEntity` data type can be used to activate the notification on specific AI/ML inference trustworthiness indicators based on the request by the authorized consumer.

`SupportedMlInferenceTrustworthiness` <<dataType>> specifies the inference trustworthiness indicator(s), e.g. explainability indicators, fairness indicators, robustness indicators, which can be supported by an `MLEntity`. It contains the tuples of `supportedMlInferenceTrustworthinessMetric` and `activatedMlInferenceTrustworthinessMetric` attributes.

The `supportedMlInferenceTrustworthinessMetric` indicates inference trustworthiness metrics that an `MLEntity` is capable of providing. The authorized consumer shall be notified only on a specific subset of such inference trustworthiness metrics for which the `activatedMlTrainingTrustworthinessMetric` indicator is set.

This solution may extend the ML Inference Request related IOC by adding a new attribute, e.g. `inferenceTrustworthinessRequirements`, to allow the ML entity inference MnS consumer to request the ML entity inference MnS producer to infer the decisions with specified inference trustworthiness requirements. Similarly, the ML Inference Response IOC may also need to be extended with a new attribute, e.g. `achievedInferenceTrustworthiness,` to allow the ML entity inference MnS producer to report the achieved inference trustworthiness score on the deployed ML model for inference.

The newly introduced attributes `inferenceTrustworthinessRequirements and achievedInferenceTrustworthiness` are of data type `modelTrustworthiness.`

5.3.1.4.5 Assessment of AI/ML trustworthiness

This solution extends MLEntity to include a new attribute of datatype `SupportedMlAssessmentTrustworthiness` indicating on the ML trustworthiness assessment indicators that can be supported by an `MLEntity`. This extended `MLEntity` data type can be used to activate the notification on specific ML trustworthiness assessment indicators based on the request by the authorized consumer.

`SupportedMlAssessmentTrustworthiness` <<dataType>> specifies the trustworthiness assessment indicator(s) that can be supported by an `MLEntity`. It contains the tuples of `supportedMlAssessmentTrustworthinessMetric` and `activatedMlAssessmentTrustworthinessMetric` attributes.

The `supportedMlAssessmentTrustworthinessMetric` indicates assessment trustworthiness metrics that an `MLEntity` is capable of providing. The authorized consumer shall be notified only on a specific subset of such assessment trustworthiness metrics for which the `activatedMlAssessmentTrustworthinessMetric` indicator is set.

Introduce the ML Assessment Request as an IOC to allow the MnS consumer to request the MnS producer for the assessment of ML entity fulfilment of trustworthiness requirements. Similarly, the ML Assessment Response IOC may also need to be introduced to allow the MnS producer to report the achieved assessment trustworthiness score on the assessed ML model. These iOCs may be name contained in an **MLAssessmentFunction** IOC

The **MLAssessmentRequest** IOC may include the **mLEntityId** (the Identifier of the ML entity that needs to be assessed) as well as attributes on:

- **candidateAssessmentData -** It provides the address(es) of the candidate assessment data source provided by MnS consumer.
- **assesmentTrustworthinessRequirements -** It is of data type modelTrustworthiness (with three attributes: trustworthinessType, trustworthinessMetric and trustworthinessScore). The trustworthinessType may be one of explainability, fairness and adversarial robustness; the trustworthinessMetric may be one of the newly proposed metrics; the trustworthinessScore is the actual value of the trustworthinessMetric.

Besides the **mLEntityId, the** Identifier of the ML entity that was assessed, the **MLAssessmentReport** IOC may include attributes for:

- **usedConsumerAssessmentData -** It provides the address(es) where lists of the consumer-provided assessment data are located, which have been used for the ML model assessment.
- **achievedAssessmentTrustworthiness -** It is of data type modelTrustworthiness (with three attributes: trustworthinessType, trustworthinessMetric and trustworthinessScore). The trustworthinessType may be one of explainability, fairness and adversarial robustness; the trustworthinessMetric may be one of the newly proposed metrics; the trustworthinessScore is the actual value of the trustworthinessMetric.

5.3.1.5 Evaluation

The solutions described in clauses 5.3.1.4.1, 5.3.1.4.2, 5.3.1.4.3 and 5.3.1.4.4 adopts the NRM-based approach, proposing a new information element (`modelTrustworthiness` datatype) with clear relationship to existing information elements `MLTrainingRequest` and `MLTrainingReport` IOCs. It fully reuses the existing provisioning MnS Operations and notifications for AI/ML data trustworthiness, ML training trustworthiness and AI/ML inference trustworthiness configuration and reporting. The implementation of this NRM-based solution is straightforward. Therefore, the solutions described is a feasible solution for AI/ML trustworthiness configuration and reporting.

The solution described in clause 5.3.1.4.5 adopts the NRM-based approach, proposing new information elements (`MLAssessmentRequest` and `MLAssessmentReport` IOCs). It reuses the existing provisioning MnS Operations and notifications for configuration and reporting of AI/ML trustworthiness assessment and the implementation of this NRM-based solution is straightforward. Therefore, the solution described in clause 5.3.1.4.5 is a feasible solution to be developed further in the normative specifications.

The solutions described in clause 5.3.1.4 are promising. However, the following issues needs to be further addressed:

1) the relation between the trustworthiness indicators/metrics and the 3GPP management or network data; and

2) how to support the consumer and the producer to have a consistent interpretation of the trustworthiness indicators/metrics.

6 Deployment scenarios

This clause provides example deployment scenarios for AI/ML related functions, including ML training function, ML testing function, and AI/ML inference function, with the corresponding AI/ML management capabilities.

The deployment scenarios for **ML training function** could be, but not limited to:

- The ML training function can be located in the cross-domain management system or the domain-specific management system (i.e. a management function for RAN or CN).For instance, the ML training function for MDA can be located in the MDAF (see 3GPP TS 28.104 [2]), and the ML training function for RAN Domain ES can be located in Domain-centralized ES function (see 3GPP TR 28.813 [18] and TS 28.310 [7]).
- The ML training function can be located in the Network function, specifically for example:
  - the ML training function for Network Data Analytics is located in the NWDAF, i.e. the ML training function is the Model Training logical function (MTLF) defined in 3GPP TS 23.288 [3];
  - the ML training function for RAN intelligence is located in gNB.

The deployment scenarios for **AI/ML inference function** could be, but are not limited to:

- The AI/ML inference function is located in the cross-domain management system or in the domain-specific management system, for instance, the AI/ML inference function is MDAF (see 3GPP TS 28.104 [2]), and the AI/ML inference function for RAN Domain ES can be located in Domain-centralized ES function(see 3GPP TR 28.813 [18] and 3GPP TS 28.310 [7]).
- The AI/ML inference function is located in NWDAF, i.e. the AI/ML inference function is Analytics logical function (AnLF) in NWDAF.
- The AI/ML inference function is located in the gNB, i.e. the AI/ML inference function is RAN intelligence function specified in 3GPP TS 38.300 [16].

The deployment scenarios for **ML testing function** could be, but are not limited to:

- The ML testing function can be located in the cross-domain management system or in the domain-specific management system.
- The ML testing function is located in the Network Function specifically for example:
  - the ML testing function for Network Data Analytics is located in the NWDAF;
  - the ML testing function for RAN intelligence is located in the gNB.

The ML training function, ML testing function, and AI/ML inference function may be deployed separately from each other, or any two or all of these functions may be co-located.

The management capabilities for ML training, ML testing, and AI/ML inference are provided by the MnS producer(s), which are located in the Management Function(s) as described in 3GPP TS 28.533 [17].

As highlighted above, deployment scenarios can be various, and the following highlights some example deployment scenarios.

Example Deployment scenario 1:

The ML training function and AI/ML inference function are located in the 3GPP management system (e.g. cross-domain management function, or domain-specific management function, etc.). For instance, the ML training function and AI/ML inference function for MDA (i.e. RAN domain-specific MDA) can be located in the RAN domain-specific MDAF and in CN domain-specific MDAF or in the cross-domain MDAF (see 3GPP TS 28.104 [2]).

The ML training function and AI/ML inference function for RAN Domain ES can both be located in the Domain-centralized ES function (see 3GPP TR 28.813 [18] and 3GPP TS 28.310 [7]) to provide training capability and inference management capability as depicted in figure 6-1.

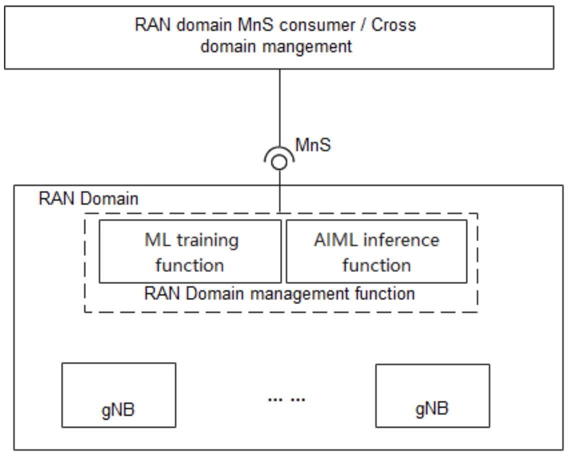

RAN domain MnS consumer / Cross domain mangement
MnS
RAN Domain
ML training function
AIML inference function
RAN Domain management function
gNB
... ...
gNB

Figure 6-1: Management for intelligence in the RAN domain

Example Deployment scenario 2:

In this deployment scenario which supports the intelligence in the RAN, the RAN domain ML training function is located in the 3GPP RAN domain-specific management function while the AI/ML inference function is located in gNB:

- **Option 2-1:** RAN domain-specific management function provides management capability for both, the ML training function (located in the RAN domain-specific management function) and the AI/ML inference capability (located at the gNB) - see figure 6-2.

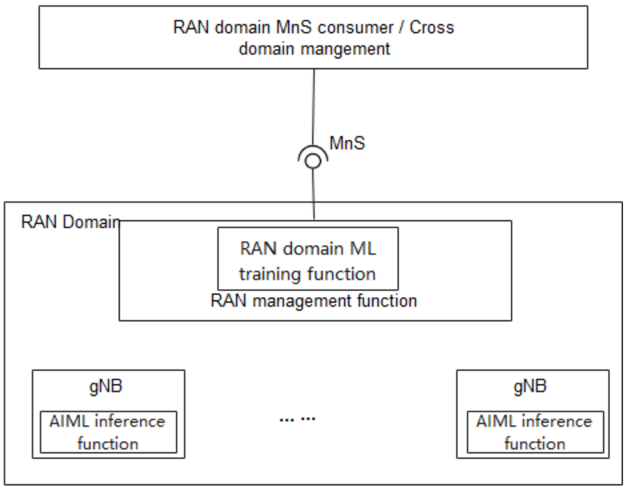


Figure 6-2: option 2-1- Intelligence in the RAN where AI/ML inference function is located in gNB and ML training is located at the RAN domain-specific management function with corresponding management capability provided by RAN domain-specific management for both ML training and AI/ML inference

- **Option 2-2:** RAN domain-specific management function provides management capability for the ML training function (located in the RAN domain-specific management function), while the management capability for AI/ML inference (located at the gNB) is also provided locally at the gNB- see figure 6-3.

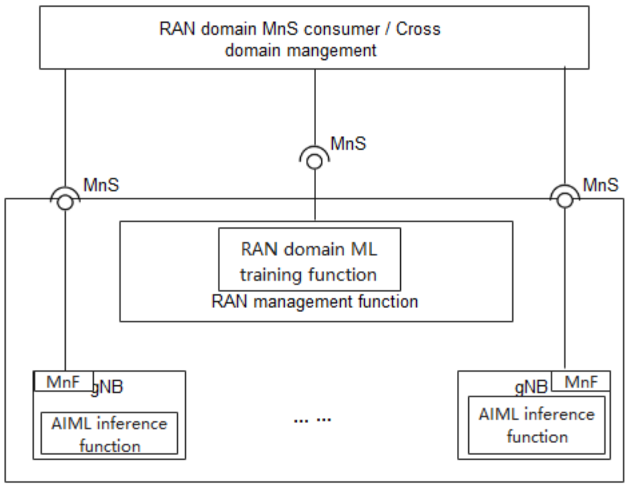


Figure 6-3: option 2-2 - Intelligence in RAN, ML training and corresponding management are located at the RAN management function while AI/ML inference function and corresponding management capability are located locally in gNB

Example Deployment scenario 3:

In this deployment scenario which supports the intelligence in RAN, the ML training function and AI/ML inference function are both located in the gNB to provide training and inference capability.

- **Option 3-1:** RAN domain-specific management function provides management capability for both the ML training function and AI/ML inference capability (both located at the gNB).

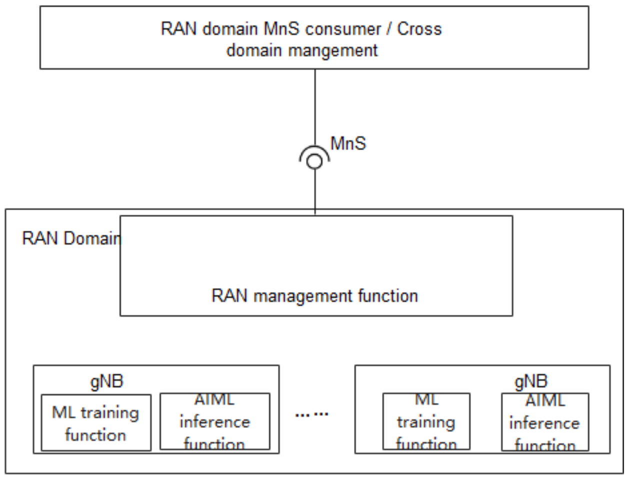


Figure 6-4: option 3-1- Management for intelligence in RAN - ML training and AI/ML inference capability are in gNB, with management capability provided by the RAN domain-specific Management function

- **Option 3-2:** management capability is provided locally at the gNB for the ML training function and AI/ML inference capability.

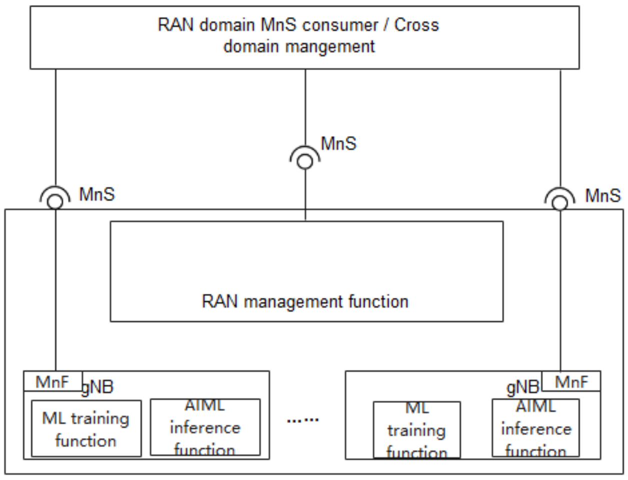


Figure 6-5: option 3-2 - management for intelligence in RAN - ML training and AI/ML inference capability and corresponding management are located in gNB

NOTE: The AI/ML RAN inference capability is standardised by the RAN WGs and is outside the scope of the present document. The split between the ML training function and AI/ML inference function within the domain is subject to implementation.

7 Conclusions and recommendations

The present technical report described the AI/ML management capabilities and services for the 3GPP 5GS (including the management and orchestration system, 5GC and NG-RAN) where AI/ML features or capabilities are employed. Clause 4 described concepts, relevant terminologies, AI/ML workflow and the overall management capabilities. The AI/ML workflow highlights three main operational phases, namely training, deployment, and inference phase. Corresponding management capabilities were identified for each of the operational phases by considering a wide range of relevant use cases along with corresponding potential requirements and possible solutions as documented in clause 5. In this clause, around 52 use cases, along with their corresponding potential requirements and possible solutions have thus far been documented and grouped under 21 categories.

The present document has also discussed and documented number of deployment-scenarios involving the AI/ML related functionalities (e.g. NWDAF, RAN intelligence and MDAF) and the corresponding management capabilities.

Moving on towards the normative specification development phase it is recommended.

To specify the AI/ML management capabilities, including use cases, requirements, and solutions for each phase of the AI/ML operational workflow for managing the AI/ML capabilities in 5GS (i.e. management and orchestration (e.g. MDA defined in 3GPP TS 28.104 [2]), 5GC (e.g. NWDAF defined in 3GPP TS 23.288 [3]) and NG-RAN (e.g. RAN intelligence defined in 3GPP TS 38.300 [16] and 3GPP TS 38.401 [19])), including management capabilities for ML training phase, management capabilities for deployment phase, and management capabilities for inference phase.

For the development of Rel-18 normative specifications, it is recommended to prioritize a subset of the already identified use cases with consideration of criteria, including the extent of relevance to the AI/ML management capabilities mapping to the three main operational phases (i.e. training, deployment, and inference) in the AI/ML workflow as well as the availability of feasible possible solution(s) in the present document.

Annex A: UML source codes

```
startuml Procedure 2. Requesting and Instantiating an MLInferenceEmulationJobs
skinparam Shadowing false
skinparam Monochrome true
!pragma teoz true
'Autonumber

skinparam maxMessageSize 130

participant "Active, unapproved \nNAF e.g. A" as NAFA
participant "Active, approved \n NAF e.g. B" as NAFB
participant "AIML \nOrchestration" as Orch
collections "Peer \n NAFs e.g. C, D,.. " as NAFC

Orch -> Orch: 1. Identify network problem
Orch -> NAFA: 2.
& Orch -> NAFC: 2. Request Capabilities
NAFA -> Orch: 3.
& NAFB -> Orch: 3. providecapabilites.
Orch -> Orch: 4. Select appropriate NAF
Orch -> NAFB: 5. Trigger NAFto find best action.

Note over NAFB, NAFC: 6. May need to coordinate effect of the action

@enduml
```

Figure A-1: Identifying and triggering automation capabilities among multiple network automation functions

```
@startuml Procedure 2. Requesting and Instantiating an MLInferenceEmulationJobs
skinparam Shadowing false
skinparam Monochrome true
!pragma teoz true
'Autonumber

skinparam maxMessageSize 130

participant "Active, unapproved \nNAF e.g. A" as NAFA
participant "Active, approved \n NAF e.g. B" as NAFB
participant "AIML \nOrchestrator" as Orch
collections "Peer \n NAFs e.g. C, D,.. " as NAFC

NAFA -> NAFA: 1. Compute configuration, e.g. select action a
& NAFB -> NAFB: 1. Compute configuration, e.g. select action a
NAFA -> Orch: 2.
& NAFB -> Orch: 2. Request action a, PM interval t sec.

Orch -> Orch: 3. Select action to execute
Orch -> NAFA: 4.
& Orch -> NAFB: 4. Notify NAFs of selected action.
Orch -> Orch: 5. Execute selected action

@enduml
```

Figure A-2: Orchestrating the decision among multiple network automation functions

```
@startuml Procedure 2. Requesting and Instantiating an MLInferenceEmulationJobs
skinparam Shadowing false
skinparam Monochrome true
!pragma teoz true
'Autonumber

skinparam maxMessageSize 130

participant "Active, unapproved \nNAF e.g. A" as NAFA
participant "Active, approved \n NAF e.g. B" as NAFB
participant "AIML \nOrchestrator" as Orch
collections "Peer \n NAFs e.g. C, D,.. " as NAFC

Note over NAFB, Orch: Action from B executed

```

```
Orch -> NAFA: 6.
& Orch -> NAFC: 6. Notify execution, trigger PM for interval t sec.
NAFA -> NAFA: 7. Collect PM for t sec
& NAFC -> NAFC: 7. Collect PM for t sec
NAFA -> NAFA: 8. compute & interpret KPIs to compute AQI
& NAFC -> NAFC: 8. compute & interpret KPIs to compute AQI

Alt distributed coordination
  NAFA -> NAFB: 9.
  & NAFC -> NAFB: 9. Report AQI
  NAFB -> NAFB: 10. AQI handling
  NAFB -> NAFB: 10. Revise NAF-B configuration if needed

Else centralized coordination
  NAFA -> Orch: 11.
  & NAFC -> Orch: 11. Report AQI
  Orch -> Orch: 12. AQI handling
  Orch -> NAFB: 13. Report aggregate AQI
  Orch -> Orch: 14. Compute NAF configuration decision
  Orch -> NAFB: 15. Configure NAF(s)

end

@enduml
```

Figure A-3: The control and coordination transaction of network automation functions requests and actions

Annex B: Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2022-03	SA5#142e	n/a	-	-	-	Initial skeleton	0.0.0
2022-04	SA5#142e	S5-222188	-	-	-	Add scope	0.1.0
2022-04	SA5#142e	S5-222477	-	-	-	Add AI/ML management overview	0.1.0
2022-05	SA5#143e	S5-223576	-	-	-	Add use case on AI/ML model performance	0.2.0
2022-07	SA5#144e	S5-224364	-	-	-	Requirements on AIML context	0.3.0
2022-07	SA5#144e	S5-224365	-	-	-	Add AI-ML workflow for 5GS	0.3.0
2022-07	SA5#144e	S5-224366	-	-	-	Clarifications into the overview and terminologies updates	0.3.0
2022-07	SA5#144e	S5-224367	-	-	-	Requirements on AIML Inference History	0.3.0
2022-07	SA5#144e	S5-224368	-	-	-	Add use case on AI-ML entity validation	0.3.0
2022-07	SA5#144e	S5-224369	-	-	-	Requirements on AIML Testing	0.3.0
2022-07	SA5#144e	S5-224386	-	-	-	Requirements on AIMLEntity Capability Discovery	0.3.0
2022-07	SA5#144e	S5-224370	-	-	-	Requirements on Pre-processed event data for ML training	0.3.0
2022-07	SA5#144e	S5-224371	-	-	-	Add use case on AI-ML model update	0.3.0
2022-07	SA5#144e	S5-224372	-	-	-	Add use case on AI-ML model deployment	0.3.0
2022-08	SA5#145e	S5-225693	-	-	-	Enhance AI-ML workflow	0.4.0
2022-08	SA5#145e	S5-225694	-	-	-	Potential solution on AIML Inference History	0.4.0
2022-08	SA5#145e	S5-225695	-	-	-	Potential Solution on AIMLEntity Capability Discovery	0.4.0
2022-08	SA5#145e	S5-225696	-	-	-	Clarification on AIMLEntity Capability Mapping	0.4.0
2022-08	SA5#145e	S5-225520	-	-	-	Add possible solution for AI-ML entity validation	0.4.0
2022-08	SA5#145e	S5-225697	-	-	-	Add possible solution for AI-ML entity testing	0.4.0
2022-08	SA5#145e	S5-225698	-	-	-	Add possible solution on AI-ML model update	0.4.0
2022-08	SA5#145e	S5-225699	-	-	-	Use case on Coordination of the AIML capability between 5GC/RAN and 3GPP management system	0.4.0
2022-08	SA5#145e	S5-225700	-	-	-	Add use case on AI-ML model configuration	0.4.0
2022-08	SA5#145e	S5-225701	-	-	-	Use case on Abstract AIML Behavior	0.4.0
2022-08	SA5#145e	S5-225702	-	-	-	Use case on impacts of AIML inference actions	0.4.0
2022-08	SA5#145e	S5-225703	-	-	-	Use case on ML gradual activation	0.4.0
2022-08	SA5#145e	S5-225704	-	-	-	Requirements on Mobility of AIML Context	0.4.0
2022-08	SA5#145e	S5-225705	-	-	-	Potential solutions on AIML context	0.4.0
2022-08	SA5#145e	S5-225802	-	-	-	Use case on orchestrating AIML inference	0.4.0
2022-08	SA5#145e	S5-225043	-	-	-	Use case on improving AI/ML (re-)training efficiency	0.4.0
2022-11	SA5#146	S5-226916	-	-	-	Addressing wording issues	0.5.0
2022-11	SA5#146	S5-226917	-	-	-	pCR 28.809 Clarifying simultaneous and separate execution of training and inference phases	0.5.0
2022-11	SA5#146	S5-226918	-	-	-	Corrections on terms and definitions	0.5.0
2022-11	SA5#146	S5-226558	-	-	-	Corrections including editorials	0.5.0
2022-11	SA5#146	S5-226924	-	-	-	Add use cases for AI/ML performance evaluation	0.5.0

Change history							
Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
2022-11	SA5#146	S5-226927	-	-	-	Add potential solutions for AI-ML performance evaluation	0.5.0
2022-11	SA5#146	S5-226919	-	-	-	Use case and potential solution on Abstraction of AIML performance	0.5.0
2022-11	SA5#146	S5-226920	-	-	-	Selection of AIML performance indicators	0.5.0
2022-11	SA5#146	S5-226921	-	-	-	Policy based selection of Performance Indicators	0.5.0
2022-11	SA5#146	S5-226928	-	-	-	Terminology on Trustworthy AI/ML	0.5.0
2022-11	SA5#146	S5-226929	-	-	-	Add use case on AIML data trustworthiness	0.5.0
2022-11	SA5#146	S5-226934	-	-	-	Add use case on AIML training trustworthiness	0.5.0
2022-11	SA5#146	S5-226935	-	-	-	Add use case on AIML trustworthiness indicators	0.5.0
2022-11	SA5#146	S5-226936	-	-	-	Add use case on AIML inference trustworthiness	0.5.0
2022-11	SA5#146	S5-226937	-	-	-	Add use case and potential solution on AIML modularity	0.5.0
2022-11	SA5#146	S5-226940	-	-	-	Add use case on AIML inference emulation	0.5.0
2022-11	SA5#146	S5-226372	-	-	-	Correct the illustration of AIML inference history request and control	0.5.0
2022-11	SA5#146	S5-226373	-	-	-	Correct the illustration of AIML entity testing and control	0.5.0
2022-11	SA5#146	S5-226942	-	-	-	Add possible solution for AI/ML configuration	0.5.0
2022-11	SA5#146	S5-226941	-	-	-	Potential solution on ML Gradual Activation	0.5.0
2022-11	SA5#146	S5-226943	-	-	-	Use case on policy-based Activation	0.5.0
2022-11	SA5#146	S5-226945	-	-	-	Potential solution on improving retraining efficiency	0.5.0
2022-11	SA5#146	S5-226986	-	-	-	Potential solution on event data for ML training	0.5.0
2022-11	SA5#146	S5-226371	-	-	-	Correct the illustration of Event data for ML training	0.5.0
2022-11	SA5#146	S5-226385	-	-	-	Add use case on measurement data for ML training	0.5.0
2022-11	SA5#146	S5-226984	-	-	-	Potential solution on Mobility of AI/ML Context	0.5.0
2022-11	SA5#146	S5-226985	-	-	-	Potential solution on Producer Initiated AIML Training	0.5.0
2022-11	SA5#146	S5-226048	-	-	-	Potential solution on Abstract AIML Behavior	0.5.0
2022-11	SA5#146	S5-226987	-	-	-	Potential solution on Orchestrating AIML inference	0.5.0
2022-11	SA5#146	S5-226988	-	-	-	Evaluations for Solution 5.6 and 5.7	0.5.0
2022-11	SA5#146	S5-227011	-	-	-	Add possible solutions for AI/ML entity deployment	0.5.0
2022-11	SA5#146	S5-226997	-	-	-	Update description and requirements on supporting online AIML entity testing	0.5.0
2022-11	SA5#146	S5-226555	-	-	-	Add use case on training data effectiveness report and analytics	0.5.0
2022-12	SA#98e	SP-221164				EditHelp review, presented for information	1.0.0
2022-12	SA#98e	SP-221213				It replaces the previous version, which was based on the wrong version of the specification	1.0.1
2023-03	SA5#147	S5-232807	-	-	-	Re-structing content of clause 5 Use cases, potential requirements, and possible solutions	1.1.0
2023-03	SA5#147	S5-232794	-	-	-	Add Definitions and terms	1.1.0
2023-03	SA5#147	S5-232793	-	-	-	Update AIML management capabilities	1.1.0
2023-03	SA5#147	S5-232795	-	-	-	Correction of terminologies	1.1.0
2023-03	SA5#147	S5-232796	-	-	-	Update the use case of ML entity performance evaluation	1.1.0
2023-03	SA5#147	S5-232808	-	-	-	Use case & Potential solution on Transfer learning	1.1.0
2023-03	SA5#147	S5-232975	-	-	-	AIML Update control	1.1.0
2023-03	SA5#147	S5-233009	-	-	-	Add use case on multiple ML entities testing	1.1.0
2023-03	SA5#147	S5- 232848	-	-	-	Potential solution and evaluation on Measurement data correlation analytics	1.1.0
2023-03	SA5#147	S5- 233040	-	-	-	Potential solution and evaluation on Training data effectiveness analytics	1.1.0
2023-03	SA5#147	S5- 233041	-	-	-	Potential solution and evaluation on Training data effectiveness reporting	1.1.0
2023-03	SA5#147	S5-232847	-	-	-	Add the evaluation of AIML update management	1.1.0
2023-03	SA5#147	S5-232281	-	-	-	Update the solution of ML entity performance evaluation	1.1.0
2023-03	SA5#147	S5-233083	-	-	-	Split AI/ML configuration management into two phases (inference and ML training)	1.1.0
2023-03	SA5#147	S5-232974	-	-	-	Potential Solution on AIML trustworthiness indicators	1.1.0
2023-03	SA5#147	S5-232614	-	-	-	Potential Solution on AIML data trustworthiness	1.1.0
2023-03	SA5#147	S5-232615	-	-	-	Potential Solution on AIML training trustworthiness	1.1.0
2023-03	SA5#147	S5-232617	-	-	-	Potential Solution on AIML inference trustworthiness	1.1.0
2023-03	SA5#147	S5-232618	-	-	-	Add assessment of AIML trustworthiness	1.1.0
2023-03	SA5#147	S5-232815	-	-	-	Updates and Corrections for Configuration Management	1.1.0
2023-03	SA5#147	S5-232643	-	-	-	Add possible solution for multiple ML entities testing	1.1.0
2023-03	SA5#147	S5-233039	-	-	-	Add potential solution and evaluation on coordination between the ML capabilities	1.1.0
2023-03	SA5#147	S5-233138	-	-	-	Add AIML capabilities deployment scenarios	1.1.0
2023-03	SA5#147	S5-232085	-	-	-	Add general conclusions and recommendations	1.1.0
2023-04	SA5#148e	S5-233574	-	-	-	Rapporteur clean-up & clarifications	1.2.0

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2023-04	SA5#148e	S5-233575	-	-	-	Add description for ML entity update	1.2.0
2023-04	SA5#148e	S5-233587	-	-	-	Update the solution of AI/ML update control	1.2.0
2023-04	SA5#148e	S5-233271	-	-	-	Replacing mandatory language from requirements	1.2.0
2023-04	SA5#148e	S5-233576	-	-	-	Add description for ML entity re-training	1.2.0
2023-04	SA5#148e	S5-233382	-	-	-	Clarifications to ML entity re-training use case	1.2.0
2023-04	SA5#148e	S5-233577	-	-	-	Add description for ML entity joint training	1.2.0
2023-04	SA5#148e	S5-233384	-	-	-	Clarifications to Training data effectiveness reporting and analytics use case	1.2.0
2023-04	SA5#148e	S5-233578	-	-	-	Add use case and requirement on Model evaluation for ML testing	1.2.0
2023-04	SA5#148e	S5-233579	-	-	-	Add potential solution on Model evaluation for ML testing	1.2.0
2023-04	SA5#148e	S5-233590	-	-	-	Add solution for AIML inference emulation	1.2.0
2023-04	SA5#148e	S5-233580	-	-	-	Add solution for configuration of AIML inference	1.2.0
2023-04	SA5#148e	S5-233588	-	-	-	Enhancement to solutions on AIML trustworthiness	1.2.0
2023-04	SA5#148e	S5-233589	-	-	-	Evaluation for AI/ML trustworthiness	1.2.0
2023-04	SA5#148e	S5-233383	-	-	-	Further clarifications and corrections to deployment scenarios	1.2.0
2023-09	SA#101	SP-230929				Presented for approval	2.0.0
2023-09	SA#101					EditHelp review and upgrade to change control version	18.0.0

History

Document history		
V18.0.0	May 2024	Publication