

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Y.3181

(09/2022)

SERIES Y: GLOBAL INFORMATION
INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS,
NEXT-GENERATION NETWORKS, INTERNET OF
THINGS AND SMART CITIES

Future networks

**Architectural framework for machine learning
sandbox in future networks including IMT-2020**

Recommendation ITU-T Y.3181

ITU-T



ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	Y.1300–Y.1399
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Computing power networks	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	Y.3000–Y.3499
CLOUD COMPUTING	Y.3500–Y.3599
BIG DATA	Y.3600–Y.3799
QUANTUM KEY DISTRIBUTION NETWORKS	Y.3800–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	Y.4400–Y.4549
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Y.3181

Architectural framework for machine learning sandbox in future networks including IMT-2020

Summary

Recommendation ITU-T Y.3181 provides an architectural framework for machine learning (ML) sandbox in future networks including IMT-2020. More precisely, it describes requirements and high-level architecture for ML sandbox in future networks including IMT-2020.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Y.3181	2022-09-29	13	11.1002/1000/15058

Keywords

High-level architecture, IMT-2020, machine learning, requirement, sandbox, simulator.

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2022

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

	Page
1 Scope	1
2 References.....	1
3 Definitions	1
3.1 Terms defined elsewhere	1
3.2 Terms defined in this Recommendation.....	2
4 Abbreviations and acronyms	3
5 Conventions	3
6 Introduction	4
7 Requirements	5
7.1 Simulated ML underlay requirements	5
7.2 Operational requirements	6
7.3 Communication requirements	7
7.4 Metadata requirements	7
8 High-level architecture	8
8.1 ML sandbox within the high-level ML architecture.....	8
8.2 Components of the ML sandbox	10
8.3 APIs	12
8.4 Sequence diagrams	16
9 Security considerations.....	23
Bibliography.....	25

Recommendation ITU-T Y.3181

Architectural framework for machine learning sandbox in future networks including IMT-2020

1 Scope

This Recommendation provides an architectural framework for the machine learning (ML) sandbox in the context of integrating machine learning in future networks including IMT-2020. This Recommendation provides requirements and high-level architecture of the ML sandbox. Architectural components along with corresponding reference points and application programming interfaces (APIs) are specified.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T Y.2701] Recommendation ITU-T Y.2701 (2007), *Security requirements for NGN release 1*.
- [ITU-T Y.3101] Recommendation ITU-T Y.3101 (2018), *Requirements of the IMT-2020 network*.
- [ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*.
- [ITU-T Y.3173] Recommendation ITU-T Y.3173 (2020), *Framework for evaluating intelligence levels of future networks including IMT-2020*.
- [ITU-T Y.3174] Recommendation ITU-T Y.3174 (2020), *Framework for data handling to enable machine learning in future networks including IMT-2020*.
- [ITU-T Y.3176] Recommendation ITU-T Y.3176 (2020), *Machine learning marketplace integration in future networks including IMT-2020*.
- [ITU-T Y.3179] Recommendation ITU-T Y.3179 (2021), *Architectural framework for machine learning model serving in future networks including IMT-2020*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 machine learning model [ITU-T Y.3172]: Model created by applying machine learning techniques to data to learn from.

3.1.2 machine learning pipeline [ITU-T Y.3172]: A set of logical nodes, each with specific functionalities, that can be combined to form a machine learning application in a telecommunication network.

3.1.3 machine learning sandbox [ITU-T Y.3172]: An environment in which machine learning models can be trained and their effects tested on the network evaluated.

3.1.4 machine learning function orchestrator [ITU-T Y.3172]: A logical node with functionalities that manage and orchestrate the nodes in a machine learning pipeline.

3.1.5 machine learning marketplace [ITU-T Y.3176]: A component that provides capabilities facilitating the exchange and delivery of machine learning models among multiple parties.

NOTE 1 – Examples of parties include suppliers and users of ML models. Capabilities provided to users of ML models include functionalities to find, learn about, deploy (or download), and use ML models. Capabilities provided to suppliers of ML models (e.g., data scientists) include functionalities to share (on-board, upload), describe (learn about), and market their ML models.

NOTE 2 – A network operator may use a machine learning marketplace deployed internally and/or externally to the network operator's administrative domains. Internal and external marketplaces differ only in the deployment perspective. A marketplace that is internal to a network operator may act as an external marketplace to another network operator and vice versa.

3.1.6 machine learning model metadata [ITU-T Y.3176]: Information that describes the characteristics of a machine learning model.

NOTE – Machine learning model metadata includes, but is not limited to, the name of the ML model, ML model's author, version of the ML model, license information of the ML model, description of the data inputs and outputs of the ML model, and runtime environment of the ML model.

3.1.7 machine learning model serving [ITU-T Y.3179]: A process of preparing and deploying machine learning models in different deployment environments to enable the application of model inference to machine learning underlay networks.

3.1.8 network intelligence level [ITU-T Y.3173]: Level of application of the automation capabilities including those enabled by the integration of artificial intelligence techniques in the network.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 evaluation ML pipeline: Chaining of pipeline nodes and simulated network functions (NFs) with served machine learning (ML) models whose goal is to evaluate a particular ML use case.

3.2.2 simulation component metadata: Data describing the characteristics of a particular simulation component.

NOTE – Examples of simulation component metadata are capabilities of the simulated NFs, configurable parameters, performance indicators, monitored parameters and interfaces.

3.2.3 simulation environment metadata: Data describing the characteristics of a particular simulation environment.

NOTE 1 – Simulation environment metadata can contain information such as installation/execution requirements, simulation component metadata, performance indicators, connections and maturity indicators (e.g., alpha/beta versions).

NOTE 2 – Examples of formats for representing simulation environment metadata are JavaScript object notation (JSON) [b-IETF RFC 8259], comma-separated values (CSV) [b-IETF RFC 4180], or extensible markup language (XML) [b-XML].

3.2.4 simulation profile: A list of parameters and their values which describe the machine learning (ML) use case to be trained, evaluated, or tested at the ML sandbox.

NOTE – The list of parameters and their values may be derived from the ML intent [ITU-T Y.3172] and the simulation environment metadata.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AF	Application Function
AI	Artificial Intelligence
AP	Access Point
API	Application Programming Interface
DBr	Data Broker
DH	Data Handling
DM	Data Model
GAN	Generative Adversarial Network
KPI	Key Performance Indicator
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
NF	Network Function
RAN	Radio Access Network
RL	Reinforcement Learning
SL	Supervised Learning
UE	User Equipment
UL	Unsupervised Learning
uRLLC	Ultra-Reliable Low-Latency Communication
V2X	Vehicle-to-Everything
WLAN	Wireless Local Area Network

5 Conventions

In this Recommendation:

- The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.
- The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.
- The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option, and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.
- The colour "solid blue" is used in Figure 2 and Figure 3 to indicate components and interfaces that are newly defined in this Recommendation.
- ML pipeline – In this Recommendation, in alignment with the conventions of [ITU-T Y.3172] when the symbol shown in Figure 1 is used, this denotes a subset (including proper subset) of nodes in an ML pipeline. When this symbol is used in a figure, the symbol stands for the subset of an ML pipeline's nodes not explicitly shown in that figure.

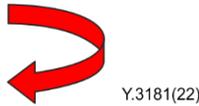


Figure 1 – Symbol used to denote a subset of nodes in an ML pipeline

6 Introduction

The integration of artificial intelligence (AI) and machine learning (ML) has been identified as one of the key features of future networks. However, network operators have the challenge of maintaining the operational performance and associated key performance indicators during or after this integration. In addition, the introduction of ML techniques to IMT-2020 networks may raise concerns regarding the transparency, reliability, and availability of the ML methods, techniques and data.

Studying the trade-offs, advantages, and disadvantages while integrating various ML mechanisms is important to understand their impact on the network. For example, reducing the generalization error is the main concern in applying any kind of supervised learning (SL) approach, which can be high even if the test error is kept low (this phenomenon is commonly known as *overfitting*). Similarly, unsupervised learning (UL) aims to find patterns from data without any guidance (unlabelled data) and hence lacks validation. On the other hand, reinforcement learning (RL) is based on the learning-by-experience paradigm. RL has been shown to be of great utility for single-agent approaches in controlled scenarios, however notable adverse effects can appear as a result of the competition raised by multiple systems sharing the same resources (e.g., while providing heterogeneous services using common network resources).

Thus, all kinds of learning can lead to unexpected and/or undesired behaviour in live networks. Even if the performance of networking systems can be improved by ML techniques in the long term, it is safe to assume that the system will unavoidably experience certain performance degradation during a transitory regime. In some situations, this degradation of key performance indicators (KPI) may be unacceptable for network operators, especially for demanding requirements of certain network-oriented applications such as ultra-reliable low-latency communication (uRLLC) applications. In other cases, the network may change quickly and may not reach a stable, long-term regime that is expected to optimize the network's performance.

NOTE – The transitory regime precedes the stability phase of an ML model when applied to a network. Performance degradation can result from potential delays in serving models in the network, or from trying suboptimal configurations during exploration periods in online learning.

Given the instability that ML methods can generate in communications systems, which can be particularly exacerbated in online mechanisms including exploration phases, the sandbox subsystem [ITU-T Y.3172] emerges as a promising solution for training, testing, and evaluating the performance of ML models before being deployed in live networks. The ML sandbox is an isolated environment in which machine learning models can be evaluated. The ML sandbox is therefore meant to reproduce the behaviour / operation of live networking systems, thereby improving the robustness and resilience of future ML-enabled networking systems. ML sandbox includes a managed test network (e.g., a testbed) or a software-based environment (e.g., using a simulator or an emulator). Software-based network environments can be particularly useful to overcome the limitations of limited training data sets and laboratory-based testbeds. For instance, simulators can be used to frame cases that have not been noticed before (i.e., anomalies), which would contribute to enabling failure prediction, anomaly detection and self-healing.

Through the management subsystem, network operators can manage the ML sandbox and thereby address the challenges posed by ML-driven solutions for networks. The interfaces between the machine learning function orchestrator (MLFO) and the ML sandbox allow the manageability of the replicated network environment (e.g., simulation), the execution of test cases and the evaluation of ML models.

7 Requirements

The requirements for the ML sandbox's architectural framework are divided into the following categories:

- Simulated ML underlay requirements
- Operational requirements
- Communication requirements
- Metadata requirements

7.1 Simulated ML underlay requirements

REQ-ML-SANDBOX-001: The ML sandbox is required to simulate heterogeneous sources of data (SRCs) and sinks (SINKs) of an ML output.

NOTE 1 – SRCs and SINKs simulated in the ML sandbox include those within the IMT-2020 network as well as application functionalities hosted in the network slices. Examples of application functionalities hosted in the network slices are vehicle-to-everything (V2X) applications, Industry 4.0 applications and emergency applications.

REQ-ML-SANDBOX-002: The ML sandbox is required to support the dynamic instantiation of new simulated SRCs and/or SINK nodes.

NOTE 2 – Instantiation of new simulated SRCs and SINK nodes is managed by the MLFO.

REQ-ML-SANDBOX-003: The ML sandbox is required to consider policy inputs from the operator while configuring the simulated ML underlay networks [ITU-T Y.3172].

NOTE 3 – Examples of policy inputs are those related to conflict resolution and resource management.

REQ-ML-SANDBOX-004: The ML sandbox is required to enable coordinated time synchronization of operations executed in the ML sandbox as required by the specific use case.

NOTE 4 – The time synchronization may be coordinated by the MLFO by controlling the sequence of operations executed in the ML sandbox. The sequence of operations triggered by the MLFO may be according to the synchronisation requirements of the specific use case. An example of a sequence of operations triggered by the MLFO is the generation of data by a radio access network (RAN)-specific simulator which is input into the corresponding ML model as an SRC, followed by analysis in the ML model, and finally the application of an ML inference into specific simulators for the SINK.

REQ-ML-SANDBOX-005: The ML sandbox is recommended to consider the quality of data needed for ML models (training or testing) while generating the simulated data.

NOTE 5 – The quality of data depends on the use case requirements. The requirements on the quality are input in the ML intent. Examples are alignment and similarity with live networks, including user equipment (UE) capabilities, the granularity of reported UE measurements, frequency of channel measurements, accuracy of measured parameters, etc.

REQ-ML-SANDBOX-006: The ML sandbox is recommended to support demand mapping [ITU-T Y.3173] for configuring and updating the simulated ML underlay networks.

NOTE 6 – Demand mapping is achieved by continuous, run-time, matching of the ML intent with the configuration options provided by the simulated ML underlay network. The configuration of the simulated ML underlay networks may be continuously adjusted based on demand mapping.

NOTE 7 – Demand mapping may be implemented through the analysis of data patterns, ML pipeline output and the corresponding optimization of the simulated ML underlay networks.

REQ-ML-SANDBOX-007: The ML sandbox is required to provide sanity checks to assess the correct operation of the simulated ML underlay networks.

7.2 Operational requirements

REQ-ML-SANDBOX-008: The ML sandbox is required to support multiple evaluation ML pipelines, which may be chained and interfaced with simulators from different levels of the network.

NOTE 1 – Network levels are defined in [ITU-T Y.3172].

REQ-ML-SANDBOX-009: The ML sandbox is required to support the monitoring and evaluation of ML pipelines and simulation components according to the specifications in the ML intent.

NOTE 2 – Examples of monitoring and evaluation output may include threshold-based asynchronous notifications from the ML sandbox (to the MLFO), post-processing of the ML output, metering, security threat analysis, etc. Other outputs may include updated metadata which reflects the results of the evaluations of the models in the ML sandbox.

REQ-ML-SANDBOX-010: The ML sandbox is required to support the testing and evaluation of multiple ML pipelines at the same time, including aggregated impacts on the network due to the multiple pipelines.

NOTE 3 – For example, different ML pipelines may use different types of models (e.g., based on the RL and the SL). The type of model may be decided by the MLFO based on the use case. Simultaneous evaluation of the different ML pipelines may be triggered for addressing an ML use case. The outputs of these ML pipelines may be compared to make an optimal decision.

REQ-ML-SANDBOX-011: The ML sandbox is required to support training and testing ML models that combine simulated and real data from the network.

NOTE 4 – The choice of data to be used is managed by the MLFO [ITU-T Y.3172].

NOTE 5 – The combination of simulated and real data may also include augmented data.

REQ-ML-SANDBOX-012: The ML sandbox is required to support dynamic resource management for ML pipeline nodes instantiated in the ML sandbox.

NOTE 6 – The instances of ML pipeline nodes in the ML sandbox (e.g., simulated SRC node) may need resource management mechanisms like the dynamic resource allocation. The ML sandbox may use various request handling mechanisms like load balancing towards the ML pipeline nodes (e.g., ML model) in the ML sandbox.

REQ-ML-SANDBOX-013: The ML sandbox is required to enable granular evaluation of the ML test cases.

NOTE 7 – In the case of batch jobs (combined test cases) which are triggered by the ML sandbox, isolation of problems found in the evaluation stage require granular information on the specific test case that has failed. The ML sandbox is needed to enable such granular evaluation.

REQ-ML-SANDBOX-014: The ML sandbox is required to support monitoring and evaluating the network intelligence level.

NOTE 8 – See [ITU-T Y.3173] for monitoring and evaluating the network intelligence level.

REQ-ML-SANDBOX-015: The ML sandbox is required to support testing techniques to enhance the robustness of the ML pipelines.

NOTE 9 – Examples of testing techniques include regression and/or integration of testing techniques for testing the ML models, data generation techniques for ensuring quality and augmentation of simulated data, simulation of failure scenarios, or rare scenarios for ML model training.

REQ-ML-SANDBOX-016: The ML sandbox is required to produce the output of simulations, tests and evaluations in a machine-readable format.

NOTE 10 – Metadata corresponding to the model may be updated with the results of the evaluations. Such updated metadata may be used by the MLFO in future selections of models.

7.3 Communication requirements

REQ-ML-SANDBOX-017: The ML sandbox is required to support data handling (DH) reference points toward technology-specific simulated ML underlay networks.

NOTE 1 – Data handling reference points are defined in [ITU-T Y.3174].

REQ-ML-SANDBOX-018: The ML sandbox is required to support the transfer of trained models across the different ML pipelines in the sandbox as well as to other subsystems in the ML overlay.

NOTE 2 – Application and reuse of trained models from the ML sandbox for many use cases are examples of scenarios where the transfer of trained models across different ML pipelines in the ML sandbox is required. The transfer and deployment of trained models in live networks to enable specific use cases is an example of a scenario that requires the transfer of trained models from the ML sandbox to other ML overlays.

REQ-ML-SANDBOX-019: The ML sandbox is required to support the transfer of data for training or testing models across different ML pipelines in the sandbox as well as to other ML overlays.

REQ-ML-SANDBOX-020: The ML sandbox is required to support interfaces with the ML marketplaces to transfer ML models and the corresponding metadata.

NOTE 3 – See clause 8.2.2.2 reference point 13 in [ITU-T Y.3176] for the interface between ML marketplaces and the ML sandbox. This interface serves both in the downlink (e.g., download models) and the uplink (e.g., update models).

NOTE 4 – An example of metadata is the outcome of applying an ML model in a live or test network, which can be used to enhance trust and confidence in an ML model available in the marketplace.

REQ-ML-SANDBOX-021: The ML sandbox is required to support data handling mechanisms including metadata storage, communication interfaces with data models and ML underlay networks, and data storage.

NOTE 5 – See [ITU-T Y.3174] for data handling mechanisms.

7.4 Metadata requirements

REQ-ML-SANDBOX-022: The ML sandbox is recommended to reuse the ML metadata store across different ML underlay networks to allow the interworking between the evaluation ML pipelines and the simulated ML underlay networks.

NOTE 1 – API-g is stored in the management subsystem to allow the training, testing, and evaluation of ML models in the simulated ML pipeline [ITU-T Y.3174].

NOTE 2 – Data models (DMs) and the corresponding API-s used in the simulated ML underlay network are stored in the management subsystem to allow the interworking between the data broker (DBr) and the simulated NFs [ITU-T Y.3174].

REQ-ML-SANDBOX-023: The ML sandbox is recommended to derive the simulation profile from the ML intent inputs, from the MLFO along with the simulation environment metadata and use it to configure and update the simulated ML underlay networks.

NOTE 3 – The simulation profile may include a list of parameters and their values which describe the ML use case to be trained, evaluated, or tested at the ML sandbox. The MLFO can provide the ML intent inputs offline or at runtime, based on the triggers.

NOTE 4 – The simulation environment metadata describes the parameters of each simulator. This is provided by the simulation designer.

REQ-ML-SANDBOX-024: The ML sandbox is recommended to use the ML model metadata from the ML marketplace to adjust the simulated ML underlay networks and the evaluation scenarios.

NOTE 5 – For instance, the limitations of the algorithms in terms of the amount of data (e.g., unsupervised learning) should be input as the amount of data to be generated (e.g., the number of access points (AP) to be simulated, the total simulation time, the minimum number of events, etc.).

REQ-ML-SANDBOX-025: The ML sandbox is required to support the simulation environment metadata.

NOTE 6 – Simulation environment metadata can be provided to the serving framework for considering the deployment environment while creating an inference engine (see clause 8.1.3 in [ITU-T Y.3179]).

NOTE 7 – Simulation environment metadata includes the data models used by the simulated NFs and APIs to access these data.

NOTE 8 – Simulation environment metadata can be used by the data handling to select the type of storage of data (see clause 7.1.1, REQ-ML-DH-011 in [ITU-T Y.3174]).

REQ-ML-SANDBOX-026: The ML sandbox is recommended to support the isolation between different instances of the evaluation ML pipelines (instantiated for different ML underlay networks).

NOTE 9 – Examples of reasons for isolation are security, data privacy reasons and support for slicing.

8 High-level architecture

The high-level architecture of the ML sandbox is described here in the context of architecture frameworks described in [ITU-T Y.3172], [ITU-T Y.3174], and [ITU-T Y.3179]. Interactions between the components of the ML sandbox subsystem and other components of the architecture framework are elaborated with a specific focus on modifications to reference points. The components of the ML sandbox subsystem and their functionalities are described.

8.1 ML sandbox within the high-level ML architecture

To simulate ML underlay networks, the ML sandbox includes the simulated network functions (NFs), application functions (AFs), and ML pipeline(s) whose elements are managed by the MLFO [ITU-T Y.3172]. The ML sandbox is particularly useful to address the dynamic networking systems since it allows validating the effect of the ML-based optimizations before being deployed in the production environments. Besides, because of the potential limitations of the data coming from live networks (insufficient amount, privacy issues, etc.), the ML sandbox can be used to generate synthetic data as a complement to a given training procedure.

Figure 2 provides the high-level architecture showing the main involved components and the ML sandbox, which are intended to fulfil the requirements specified in clause 7.

NOTE 1 – See clause 8.2 for further details regarding the ML sandbox architectural components shown in Figure 2.

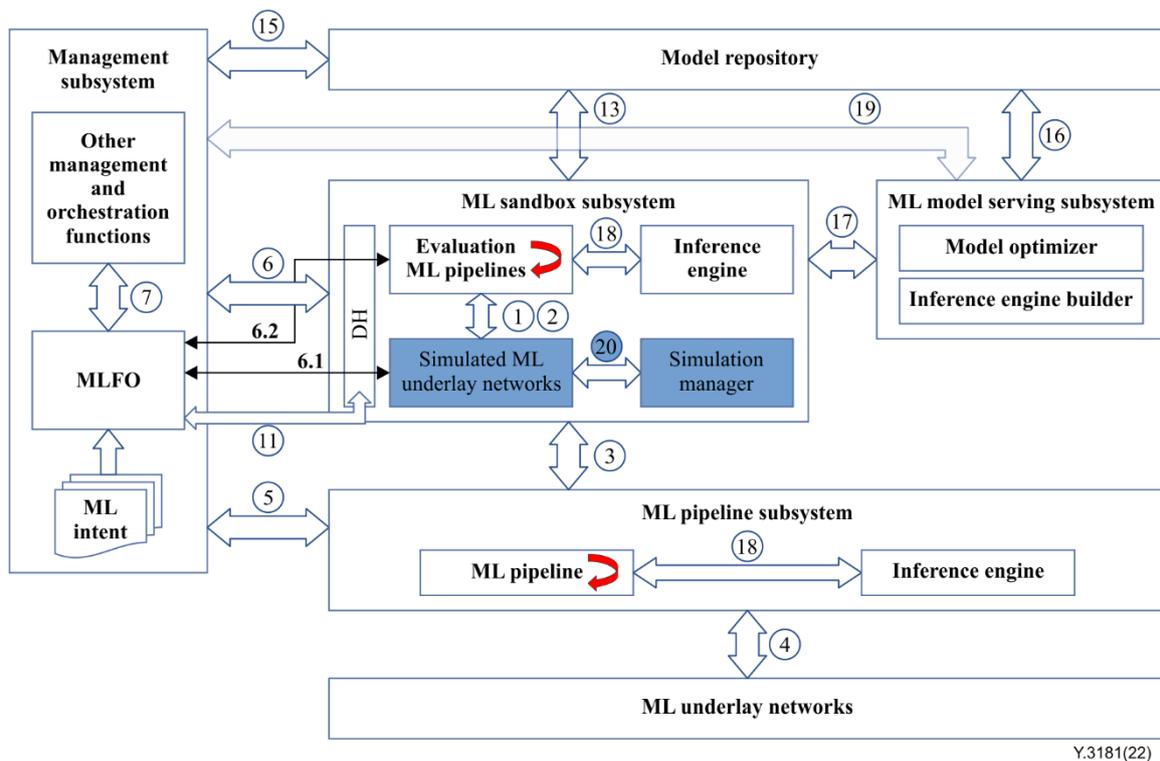


Figure 2 – ML sandbox within the high-level ML architecture

Figure 2 showcases the ML sandbox subsystem and its main components in the context of the high-level architectural framework defined in [ITU-T Y.3172]. It extends the high-level architecture for the ML model serving [ITU-T Y.3179] with specific architecture components of the ML sandbox and their corresponding interactions.

The reference points shown in Figure 2 are as follows:

Reference points 1 and 2 act as internal reference points within the ML sandbox subsystem, between the simulated ML underlay networks and the evaluation ML pipeline and are used unmodified, as defined in [ITU-T Y.3172], for training and update of ML models at the ML sandbox subsystem.

Reference point 3 is the reference point between the ML sandbox and ML pipeline subsystems [ITU-T Y.3172]. It allows the ML pipelines to interface with the ML sandbox subsystem for training and update of ML models. It is used only as a model management interface, as described in [ITU-T Y.3179].

Reference point 4, as defined in [ITU-T Y.3174], is the interface between the ML pipeline subsystem and the ML underlay network. It is used for the transfer of data between the ML underlay network and the (evaluation) ML pipeline instantiated in the ML sandbox (see clause 8.2 in [ITU-T Y.3174]). Data from the ML underlay networks and/or the simulated ML underlay networks may be used to train the ML models in the ML sandbox subsystem.

Reference point 5, as defined in [ITU-T Y.3172], is the interface between the management subsystem and the ML pipeline subsystem.

Reference point 6 is used for the management subsystem to manage the models applied to the ML sandbox [ITU-T Y.3172], including monitoring and evaluating network intelligence levels [ITU-T Y.3173]. Reference point 6 has two parts:

- Reference point 6.1 [ITU-T Y.3174] is the interface between the management subsystem and the simulated ML underlay network of the ML sandbox subsystem to orchestrate and manage the simulated ML underlay networks.

- Reference point 6.2 interfaces the management subsystem with the evaluation ML pipeline to orchestrate and manage the evaluation ML pipeline.

NOTE 2 – Data from the ML underlay networks and/or the simulated ML underlay networks may be used to train ML models in the ML sandbox subsystem.

Reference point 7 is the interface between MLFO and other management and orchestration functions of the management subsystem, used unmodified as defined in [ITU-T Y.3172].

Reference point 11 is the interface between the MLFO and the data handling (DH) components in the ML overlay, used unmodified as defined in [ITU-T Y.3174].

Reference point 13 is the interface between the ML marketplace and the ML sandbox subsystem, used unmodified as defined in [ITU-T Y.3176].

Reference point 15 is the interface between the management subsystem and the ML marketplace, used unmodified as defined in [ITU-T Y.3176].

Reference point 16 is the interface between the ML model serving subsystem and the model repository, used unmodified as defined in [ITU-T Y.3179].

Reference point 17 is the interface between the ML sandbox subsystem and ML model serving subsystem, used unmodified as defined in [ITU-T Y.3179].

Reference point 18 is the interface between evaluation ML pipelines and the inference engine, used from [ITU-T Y.3179].

NOTE 3 – The evaluation ML pipeline referred to here is the same as the ML pipeline in the ML sandbox subsystem in [ITU-T Y.3179].

Reference point 19 is the interface between the management subsystem and the ML model serving subsystem, used unmodified as defined in [ITU-T Y.3179].

Reference point 20 is the interface between the simulated ML underlay networks and the simulation manager used for managing the simulated ML underlay networks. See clause 7.1 for more details.

8.2 Components of the ML sandbox

The ML sandbox contains the components defined in the following subclauses. The detailed architecture of the ML sandbox subsystem is illustrated in Figure 3.

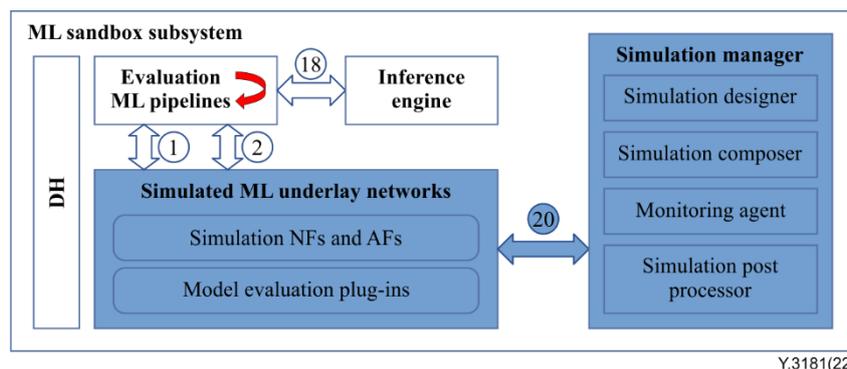


Figure 3 – Detailed architecture of the ML sandbox subsystem

8.2.1 Simulated ML underlay networks

The simulated ML underlay networks component is reused from [ITU-T Y.3172]. As explained in [ITU-T Y.3172], the ML sandbox can use data generated from the simulated ML underlay networks (obtained via reference points 1 and 2), and/or live networks (obtained via reference point 3), for training or testing of the ML models.

In this Recommendation, two subcomponents of the simulated ML underlay networks are introduced, simulated NFs and AFs, and model evaluation plug-ins.

NOTE – An example of a simulated NF is a third-party simulation tool such as an ns-3 [b-Riley-ns3].

8.2.1.1 Simulated NFs and AFs

As explained in [ITU-T Y.3174], simulated NFs and AFs provide the ability to support heterogeneous sources (SRC) of data and SINK functionality. In this Recommendation, these SRC and SINK are used for training and testing the evaluation ML pipelines.

8.2.1.2 Model evaluation plug-ins

Model evaluation plug-ins are responsible for evaluating the performance of the ML models as per the requirements defined in the use case. The plug-ins interact with the simulated NFs and AFs using technology-agnostic interfaces, which would enable the interaction with heterogeneous third-party applications such as network simulators and the collection of ML model evaluation parameters.

NOTE – Examples of ML model KPIs are model accuracy, recall and precision. Other parameters evaluated could be inference latency and memory footprint. These parameters are to be specified in the use case description provided in the ML intent [ITU-T Y.3172].

8.2.2 Simulation manager

The simulation manager manages the simulated ML underlay networks, specifically consisting of the following subcomponents: simulation designer, simulation composer, monitoring agent, and simulation post-processor.

Based on inputs from the MLFO, the simulator manager takes into account metadata and policy inputs from the operator while managing the simulated ML underlay networks. The simulation manager is responsible for achieving demand mapping [ITU-T Y.3173] while configuring and updating the simulated ML underlay networks. The simulation manager provides the dynamic resource management for the ML pipeline nodes instantiated in the ML sandbox.

8.2.2.1 Simulation designer

Based on the input from the MLFO regarding the simulation requirements for the use case, the simulation designer prepares the set of simulation resources that compose the simulated ML underlay, corresponding to the ML use case.

NOTE 1 – Inputs from the MLFO may include time-synchronization of the operations executed in the ML sandbox as required by the specific use case.

NOTE 2 – The information to design the simulated ML underlays can come from the use case (ML intent) or data gathered by the live ML underlay.

NOTE 3 – As an example, in a traffic steering use case, ML models may be applied to predictively manage the resource allocation in the network. The simulation designer arrives at the simulation needs for this use case which may include data generation and simulated resource management mechanisms and corresponding parameters and configurations. Another example is, as a result of the network dynamics, some path-loss parameters used by the ML sandbox subsystem may vary over time. To address this issue, the MLFO keeps track of those changes and provides feedback to the simulation designer to update the necessary simulation parameters.

8.2.2.2 Simulation composer

The simulation composer uses the design from the simulation designer and identifies the specific simulation components to use for the use case. It takes as input the configurations and KPIs as specified in the use case. The simulation composer then deploys, installs, and instantiates the simulated ML underlay components (e.g., NFs, AFs and model evaluation plug-ins) to be used for simulating different types of network underlays and evaluating various types of ML models. The simulation composer may chain and interface simulators from different levels of the network [ITU-T Y.3172] with multiple ML pipelines.

NOTE – Based on the specified role and requirements of the simulation (derived from use cases), the simulation composer may indicate the best simulation tool (e.g., a specific implementation of a RAN simulator). Specific testing techniques like robustness testing may be applied.

8.2.2.3 Monitoring agent

The monitoring agent monitors and evaluates the simulations in the ML sandbox, including the evaluation ML pipeline and the simulated ML underlay network. The monitoring agent enables the granular evaluation of ML test cases by the MLFO.

In addition to the use-case-specific parameters obtained from the MLFO, the following five dimensions are considered [ITU-T Y.3173]: demand mapping, data collection, analysis, decision and action implementation.

The monitoring of data collection, action implementation, and analysis is done by the monitoring agent (see clause 8.3 of [ITU-T Y.3173]). This may include monitoring the quality of data needed for the ML models (training or testing), while generating the simulated data and sanity checks to assess the correct operation of the simulated ML underlay networks.

8.2.2.4 Simulation post-processor

The simulation post-processor provides an interface whereby data from the simulated ML underlays are post-processed and presented in a standard-compliant manner to the MLFO, which performs model evaluations and/or (re)training. This step is critical to handle heterogeneous sources of information.

NOTE – For instance, once the handler gets the raw logs generated by a simulator (e.g., a CSV file), the post-processor extracts the relevant information to be used by the MLFO.

8.2.3 Evaluation ML pipeline

The evaluation ML pipeline is used for model evaluation in the ML sandbox environment as described in [ITU-T Y.3172]. This component supports the transfer of evaluated and tested models and supports interfaces with the ML marketplaces to transfer the ML models and the corresponding metadata, in coordination with the MLFO.

NOTE – The evaluation ML pipeline is similar to the ML pipeline defined in [ITU-T Y.3172], except that it uses simulated ML underlay networks instead of live ML underlay networks. For example, both the evaluation ML pipeline and live ML pipeline use reference point 4 and reference point 5.

8.2.4 Data handling

Data handling (DH) provides the functionality for the storage of data models and data for simulated ML underlay networks, used unmodified as defined in [ITU-T Y.3174]. Components of DH as defined in clause 8.2 of [ITU-T Y.3174] are instantiated in the ML sandbox subsystem. Reference points 1 and 2 are reused from [ITU-T Y.3172] and [ITU-T Y.3174] between evaluation ML pipelines and the simulated ML underlay networks. The ML sandbox utilizes DH to dynamically instantiate new simulated SRCs and/or SINK nodes.

NOTE – DH is shown here for completeness. The role and interactions with DH remain the same as defined in [ITU-T Y.3174], with the only difference of addressing evaluation ML pipelines with respect to what is covered in [ITU-T Y.3174].

8.2.5 Inference engine

The inference engine provides ML model inference capability for ML pipeline(s), used unmodified as defined in [ITU-T Y.3179].

8.3 APIs

Reference points 6.1 and 6.2 are shown in Figure 2 and introduced in clause 8.1. The realization of the requirements of the ML sandbox necessitates interaction between the ML sandbox and various

other components of the high-level architecture. Reference points 6.1 and 6.2 enable the APIs which are used for such interactions.

The specific APIs that correspond to each reference point are described below.

NOTE – Interactions between the components using the APIs defined in this clause are depicted in the sequence diagrams in clause 8.4.

8.3.1 Reference point 6.1

8.3.1.1 Capability discovery request API (Capability_Discovery)

API description: Using reference point 6.1 and complementary external interfaces with simulation capabilities, the Capability_Discovery API discovers the third-party simulation components that can be used to perform use-case-specific simulations in the ML sandbox. According to the ML use case, the MLFO finds and selects the candidate simulation environments from a list of updated capabilities.

Capability_Discovery-Request

Direction: MLFO → ML sandbox subsystem

Table 8-1 describes the information elements of Capability_Discovery-Request.

Table 8-1 – Capability_Discovery-Request information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Request identifier	Integer	Mandatory	Identifier of the request, indicating "capability discovery"
ML profile	<Attribute, value> array	Mandatory	Includes metadata defining policies, requirements, constraints, etc.

Capability_Discovery-Response

Direction: ML sandbox subsystem → MLFO

Table 8-2 describes the information elements of Capability_Discovery-Response.

Table 8-2 – Capability_Discovery-Response information elements

Information element	Type	Mandatory/Optional /Conditional	Description
List of simulation components	<Attribute, value> array	Mandatory	Updated capability list of the available simulation components
Simulation environment metadata	<Attribute, value> array	Mandatory	Metadata can contain information such as installation / execution requirements, capabilities of simulated NFs, performance indicators, configurable parameters, maturity (alpha / beta), etc.

NOTE 1 – As an example of the third-party NF, ns-3 can be selected to simulate a specific deployment of IEEE 802.11ax Wireless Local Area Networks (WLANs) [b-IEEE 802.11].

NOTE 2 – Based on the use case requirements, the list of potential NFs is narrowed. For instance, NFs can have associated information (via simulation environment metadata) such as "running time", "billing aspects", "accuracy", etc.

8.3.1.2 Status reporting API (Monitor_Reporting)

API description: Using reference point 6.1, the status reporting API reports the status of the simulation components (e.g., health status), so that the MLFO can consider taking healing actions.

Monitor_Reporting (periodical or responsive)

Direction: ML sandbox subsystem → MLFO

Table 8-3 describes the information elements of Monitor_Reporting.

Table 8-3 – Monitor_Reporting information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Notification identifier	Integer	Mandatory	Identifier of the request, indicating "monitoring report"
Status	Integer	Mandatory	Code indicating the current status of the simulation components, e.g., health status indicated by green, yellow, orange and red
Severity	Integer	Optional	Code indicating the severity of the potential anomalies identified (Critical, major, minor, normal or clear)
Monitoring logs	String list	Optional	Raw data resulting from monitoring
Alerts	<Attribute, value> array	Optional	Threshold-based alerts
Suggested action points	String list	Optional	List of suggested action points to fix the potential reported issues

NOTE – Monitoring is carried out based on a continuous flow of data generated by the simulation components, including simulation data (SRC and SINK nodes), regression tests, reporting from simulation modules, etc.

8.3.1.3 Input/Output validation reporting API (Report_IO_Validation)

API description: Using reference point 6.1, the Report_IO_Validation API is used to report the status of input/output data used/generated at/by the ML sandbox. This information can be used by the MLFO to generate new data sets, re-train ML models with different configurations, update simulation components, etc.

Report_IO_Validation

Direction: ML sandbox subsystem → MLFO

Table 8-4 describes the information elements of Report_IO_Validation.

Table 8-4 – Report_IO_Validation-Response information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Notification identifier	Integer	Mandatory	Identifier of the request, indicating "input/output validation result"
Validation type	Integer	Mandatory	Indicates the type of validation performed (e.g., input data to configure simulation parameters or output training data validity)
Result of validation	Integer	Mandatory	"Success" or "fail"
Warnings	String list	Optional	Detailed information regarding the potential issues or misbehaviours observed from the current input/output data
Error details	String list	Conditional	Detailed information regarding the errors thrown during the validation procedure

NOTE – The data to be validated includes input data (e.g., to check that demand mapping can be fulfilled at the simulated ML underlay) and simulation output data (e.g., to assess the feasibility of the trained models, the accuracy of generated data, etc.). For instance, testing techniques such as equivalence partitioning or centroid positioning [b-Zhang] can be applied to validate the diversity and the quality of the data generated by the simulators (e.g., as for validating the synthetic data).

8.3.1.4 Sandbox asynchronous messages API (Sandbox_Async)

API description: Using reference point 6.1, the Sandbox_Async API is used for the sandbox asynchronous messages defined in clause 8.4.5.

Sandbox_Async

Direction: ML sandbox subsystem → MLFO

Table 8-5 describes the information elements of Sandbox_Async.

Table 8-5 – Sandbox_Async information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Message identifier	Integer	Mandatory	Identifier of the message, indicating "Sandbox asynchronous message"
Message code	Integer	Mandatory	Code of the asynchronous message type
Additional information	String list	Conditional	Additional information related to the message type

8.3.2 Reference point 6.2

8.3.2.1 MLFO-triggered operations API (MLFO_Trigger)

API description: Using reference point 6.2, the MLFO_Trigger API is used for the MLFO-triggered operations defined in clause 8.4.4.

MLFO_Trigger-Request

Direction: MLFO → ML sandbox subsystem

Table 8-6 describes the information elements of MLFO_Trigger-Request.

Table 8-6 – MLFO_Trigger-Request information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Message identifier	Integer	Mandatory	Identifier of the message, indicating "ML-triggered operation"
Operation code	Integer	Mandatory	Code of the operation to be performed
Policies and requirements	<Attribute, value> array	Conditional	Metadata including policies and requirements.
Simulation environment metadata	<Attribute, value> array	Conditional	Metadata including simulation, configuration, available resources, time constraints, etc.

MLFO_Trigger-Response

Direction: ML sandbox subsystem → MLFO

Table 8-7 describes the information elements of MLFO_Trigger-Response.

Table 8-7 – MLFO_Trigger-Response information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Message identifier	Integer	Mandatory	Identifier of the message, indicating "ML-triggered operation"
Response code	Integer	Mandatory	Code of the operation response (OK, bad request, error, etc.)
Response data	(variable)	Conditional	Depending on the request type, different response data types can be provided (e.g., training data set, trained ML model and validated ML model)

8.4 Sequence diagrams

This clause provides sequence diagrams that result from the ML sandbox operation. The sequence diagrams are derived from the requirements in clause 7, the architectural framework defined in clause 8.1 and the APIs in clause 8.3.

8.4.1 Capability discovery

Simulation components provided by third parties can be used to perform use case-specific simulations in the ML sandbox. This procedure enables the discovery of such simulation components stored in third-party repositories. The sequence diagram is shown in Figure 4.

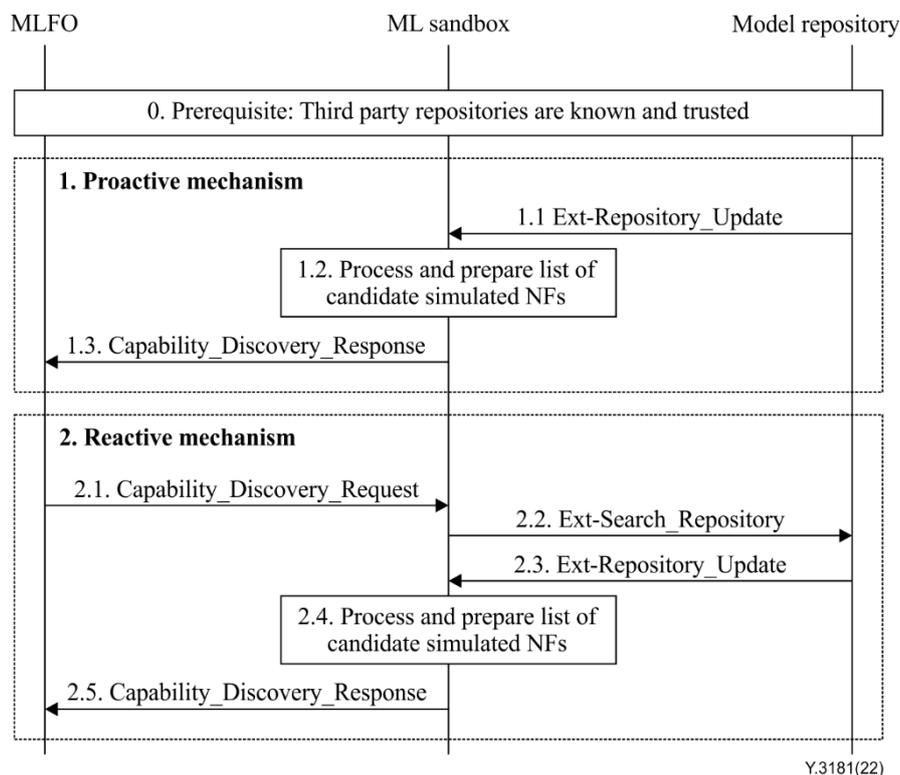


Figure 4 – Capability discovery for third-party simulation components

Prerequisite: MLFO knows the list of third-party repositories, and the offline configured, trusted, secure channels. Simulation components in the repositories are described using simulation environment metadata.

Two mechanisms (*proactive* and *reactive*) are considered according to the nature of the capability discovery notification. The steps in Figure 4 are explained below.

Proactive mechanism

This includes the following steps, as shown in Figure 4:

- 1.1 Third-party simulator repositories update their simulation components to the ML sandbox. These are to be evaluated in the ML sandbox in combination with the model evaluation plug-ins. Further component updates can also be provided to the ML sandbox as and when a third-party simulation repository (e.g., ns-3) releases specific features (e.g., MIMO support).
NOTE 1 – This step is done using an external interface, referred to as Ext-Repository_Update.
- 1.2 Information from the update message is processed to prepare an updated list of candidate simulated NFs for evaluation in the ML sandbox.
NOTE 2 – This step is done by the simulation designer (see clause 8.2.2.1).
- 1.3 The corresponding updated simulation components are published to the trusted MLFOs.

Reactive mechanism

This includes the following steps as shown in Figure 4:

- 2.1 MLFO queries the ML sandbox for simulation capabilities based on the simulation environment metadata.
- 2.2 The ML sandbox sends the query to the trusted third-party simulation component repositories.
NOTE 3 – This step is done using an external interface, referred to as Ext-Search_Repository.
- 2.3 Repositories respond with a list of simulation components matching the query.

NOTE 4 – This step is done using an external interface, referred to as Ext-Repository_Update.

2.4 Information from the response message is processed to prepare an updated list of candidate simulated NFs for evaluation in the ML sandbox.

NOTE 5 – This step is done by the simulation designer (see clause 8.2.2.1).

2.5 The corresponding updated simulation components are published to the trusted MLFOs.

Based on the use case requirements and the result of the capability discovery mechanism, MLFO arrives at the candidate simulation environments (list of NFs, simulation components, corresponding configurations, connections and data handling adaptors). From the candidate simulation environments provided by the MLFO, an operator selects an optimal configuration and deploys it in the ML sandbox.

In addition, data handling and other underlay changes are also applied based on the selected configuration.

NOTE 6 – Simulation resources in the repositories are described using the simulation environment metadata.

NOTE 7 – An example of a third-party simulated NF is ns-3, used to simulate a specific deployment of IEEE 802.11ax WLANs.

NOTE 8 – Based on the use case requirements, the list of potential NFs is decided. NFs can have the associated information (as part of the simulation environment metadata) such as "running time", "billing aspects", "accuracy", etc.

8.4.2 Health monitoring

Health monitoring is meant to ensure the proper behaviour of the simulation components in the ML sandbox. The sequence diagram is shown in Figure 5.

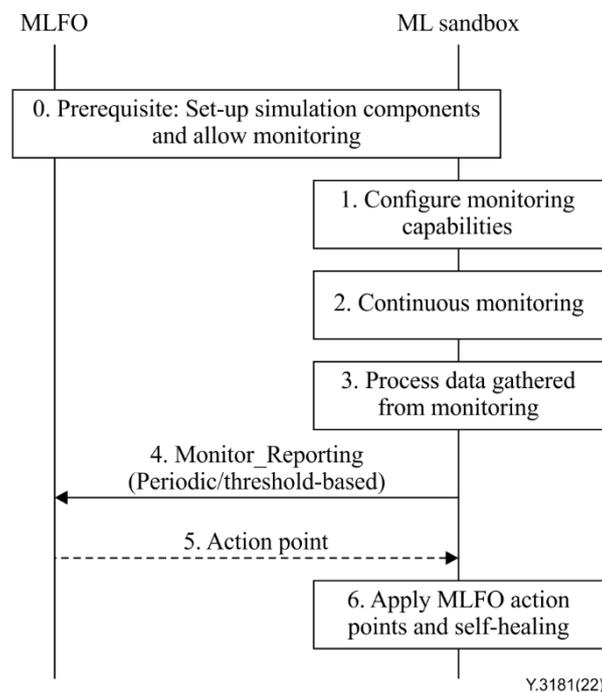


Figure 5 – Health monitoring

Prerequisite: Simulation components have been set up and the ML intent allows for monitoring simulation components.

1) ML sandbox sets up the resources for monitoring simulation components.

NOTE 1 – This step is done by the simulation designer (see clause 8.2.2.1).

- 2) Monitoring is carried out based on a continuous flow of data generated by simulation components, including simulation data (SRC and SINK nodes), regression tests, reporting from simulation modules, etc.

NOTE 2 – This step is done by the monitoring agent (see clause 8.2.2.3).

- 3) Data gathered from monitoring is processed and delivered in the form of reports.

NOTE 3 – This step is done by the simulation post-processor (see clause 8.2.2.4).

- 4) Periodic reports are generated and sent to the MLFO, according to the ML intent specification. Alternatively, threshold-based alerts can be activated when undesired events occur, which are also reported to the MLFO. MLFO may take action after processing the periodic reports or threshold-based alerts (see clause 8.4.4) and optionally send action points to the ML sandbox.

- 5) The ML sandbox applies action points (if any) suggested by the MLFO and/or self-healing actions.

NOTE 4 – The simulator composer may take action for redefining the simulation environment (e.g., switch to a more computation-intensive but accurate tool) if certain indicators of quality are not met.

8.4.3 Validate input/output data

In this scenario, input data and simulated output data are validated. This includes checking the feasibility of training models, the accuracy of generated data, etc. For instance, testing techniques such as equivalence partitioning or centroid positioning can be applied to validate the diversity and the quality of the data generated by simulators.

NOTE 1 – An example of validation of input data is to validate the synthetic data generated by generative adversarial networks (GANs) [b-Castelli-GANs].

The sequence diagram is shown in Figure 6.

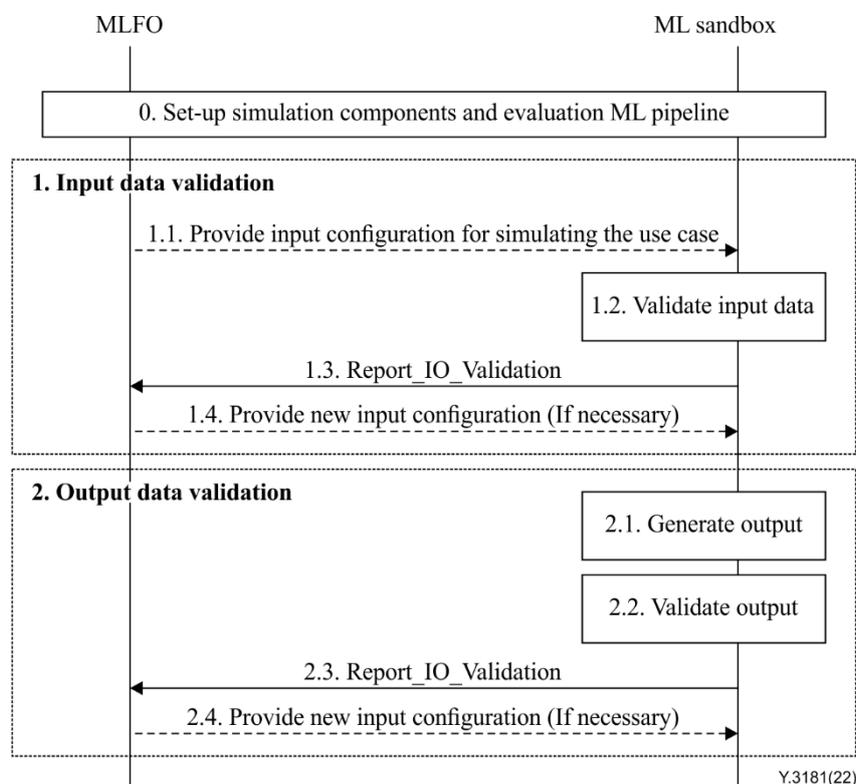


Figure 6 – Validate input/output data

Prerequisite: Simulation components and evaluation ML pipeline have been set up.

For input data:

- 1.1) ML sandbox receives the input configuration from the MLFO to configure the input data generation from the simulation environment.
- 1.2) Input data generation is validated.
NOTE 2 – This step is done by the simulation designer (see clause 8.2.2.1).
- 1.3) A response with the validation result is provided to the MLFO.
- 1.4) MLFO may provide updated inputs (if necessary) to configure the simulation environment.

For simulated output data:

- 2.1) ML sandbox receives the output data from the ML use case (training data set, trained ML model, evaluation of the ML model).
- 2.2) Output data is validated.
NOTE 3 – This step is done by the simulation post-processor (see clause 8.2.2.4).
- 2.3) The result of the validation is sent to the MLFO.
- 2.4) MLFO may provide updated inputs (if necessary) to configure the simulation environment.

8.4.4 MLFO-triggered operations

The MLFO-triggered operations are generally defined in Figure 7.

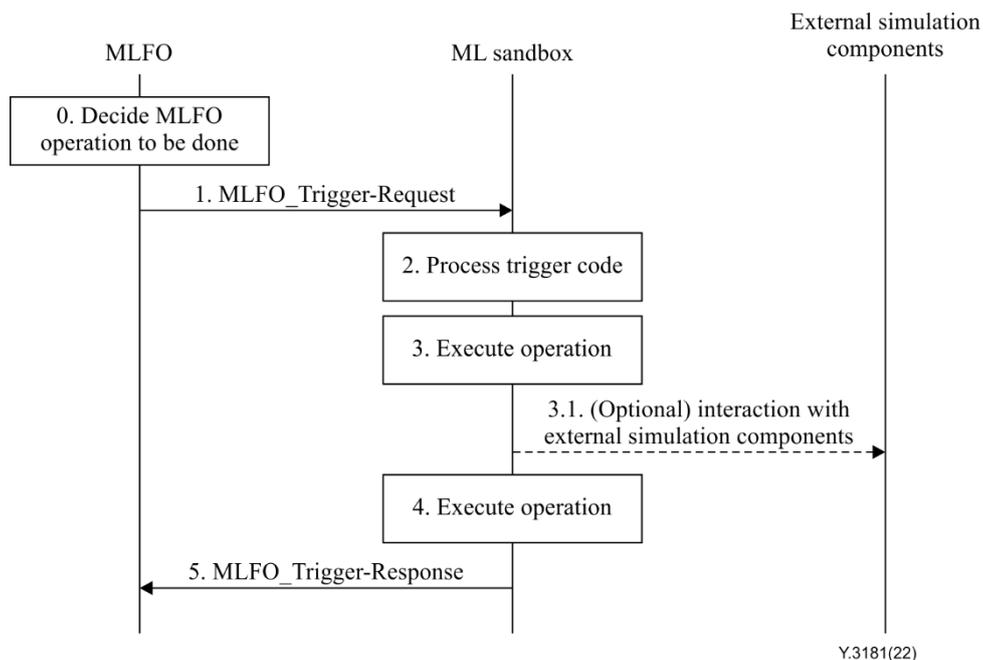


Figure 7 – MLFO-trigger operations

The MLFO_Trigger API requests and provides a response to the following set of operations:

Setup environment for the ML use case (SANDBOX-TRIGGER-001)

Prerequisite: Capability discovery is done (see clause 8.4.1).

- 1) The MLFO sends a request to the ML sandbox to set up the environment for the ML use case in the sandbox. This request contains the selected candidate simulation environment, as a result of the procedure in clause 8.4.1.
- 2) ML sandbox prepares the environment via the simulation designer (see clause 8.2.2.1):
 - a) This step can optionally include the download and install of the simulation components.

- b) Data handling for the simulated ML underlay includes the integration with the data brokers and data storage units [ITU-T Y.3174]. A plugin to interact with third party applications can also be employed for the setup.
- 3) A response is provided by the ML sandbox to the MLFO.
NOTE 1 – Environment setup may use the policies from the MLFO and demand mapping (e.g., ML intent may require isolation of resources). The ML underlay may also be configured based on the use case specification and based on the features extracted from the live ML underlay to be simulated.
NOTE 2 – Setting up the evaluation ML pipeline may include the download of ML models from the marketplaces [ITU-T Y.3176] and the ML model serving [ITU-T Y.3179] to create evaluation ML pipelines.
NOTE 3 – Environment setup may include the installation of third-party applications or setting up interfaces, establishing connections via sockets, etc.

Validate environment for ML use case (SANDBOX-TRIGGER-002)

Prerequisite: Setup environment for the ML use case is done (see clause 8.2.2).

- 1) MLFO requests to run sanity tests on a specific simulation environment for the ML use case.
- 2) The simulation designer selects and executes the test suite (interaction with third-party applications can be done through the evaluation plugin).
- 3) The output of the test suite is processed and the output to the MLFO.
NOTE 4 – For instance, installing a third-party simulation tool may output a set of traces (or logs) indicating the result of installing the submodules. This information should be post-processed to assess that the final installation procedure is successfully accomplished regarding the use case requirement. For example: if the installation of the LTE module fails, but the use case is meant for Wi-Fi (where the module was successfully installed), then the result of the installation is satisfactory.
- 4) The MLFO decides whether the validation results are acceptable or not (may include some basic tests and KPIs).

Manage simulated ML underlay (SANDBOX-TRIGGER-003)

- 1) The MLFO sends a trigger for modifying / updating the simulated ML underlay. Information on changes is included in the updated ML profile.
- 2) The simulation driver configures the simulated ML underlay accordingly (e.g., adapt configuration parameters, specify desired output, etc.). Besides, there is an information exchange between the live and the simulated ML underlays (mimic purposes).
- 3) The ML sandbox responds to the trigger with the information related to the changes done in the simulated ML underlay (OK/NOK, changelog, etc.).

NOTE 5 – For example, as a result of network dynamics, some path-loss parameters used by the ML sandbox subsystem may vary over time. To address this issue, the MLFO keeps track of those changes and provides feedback to the simulation designer to update the necessary simulation parameters.

Evaluate output of ML model (SANDBOX-TRIGGER-004)

Prerequisite: The environment has been set up in the ML sandbox for the ML use case.

- 1) The MLFO sends a request to the simulation composer in the ML sandbox to run the simulated ML underlay.
- 2) The simulation composer uses the "evaluator plugin" to input the output of the ML model. The plugin can also be used to interact with the third-party application (i.e., translate commands from the MLFO to the simulator-oriented instructions).
- 3) Evaluate the output of an ML model: the evaluation platform provides a report which is processed by the post-processor.

- 4) The processed report is sent back to the MLFO. The report can also include synthetic data for training.

NOTE 6 – The information extracted from the simulated ML underlay needs to be post-processed according to the desired output specified in the use case and the characteristics of the simulated ML underlay. The evaluation result can be an OK/NOK message, a percentage of reliability, a set of KPIs gathered from the evaluation procedure, etc.

NOTE 7 – Post-processing for third-party simulation tools may include the conversion of raw data into meaningful information (e.g., average or deviation on the KPIs).

ML model training (SANDBOX-TRIGGER-005)

Prerequisite: The ML model is served, and the evaluation blocks are ready.

- 1) The MLFO sends a trigger for training an ML model in the sandbox.
- 2) Model training is performed at the simulated ML underlay through the simulator composer.
- 3) The trained ML model is post-processed (e.g., compressed, pruned) according to the use case and the policies and capabilities (time constraints, link capacity for exchanging information, storage capabilities, etc.).
- 4) The trained model is included in the response sent to the MLFO.

Table 8-8 describes the operation codes used in the MLFO-triggered operations.

Table 8-8 – Definition of the MLFO-triggered operation codes (MLFO trigger types)

MLFO trigger type	Parameters	Time sync / dependencies	Description
SANDBOX-TRIGGER-001	Request type, ML profile	Time sync = yes (evaluation blocks)	Request to prepare the simulation environment (both simulated ML underlay and evaluation ML pipeline) to train, test, and evaluate the ML models in the ML sandbox.
SANDBOX-TRIGGER-002	Request type, validation type, acceptance criteria	Time sync = yes (sanity checklist, test suite, etc.)	Request to validate (sanity check) the deployed simulation environment. Used by the MLFO to determine whether to take action to fix potential deployment issues, proceed with the ML model evaluation in the sandbox, etc.
SANDBOX-TRIGGER-003	Request type, update type, updated ML profile	Time sync = yes (updated simulation components)	Request to modify / update the simulated ML underlay according to the updates in the policies, changes in the live ML underlay, potential failures of previous simulated functions, etc.
SANDBOX-TRIGGER-004	Request type, ML profile, ML model output	Time sync = yes (evaluation results)	Request to evaluate the impact of the output of an ML model in the simulated ML underlay, so that some insights can be provided before applying the output to the live ML underlay.
SANDBOX-TRIGGER-005	Request type, ML profile	Time sync = no	Request to train an ML model in the ML sandbox.

8.4.5 Sandbox asynchronous messages

ML sandbox asynchronous messages are generally defined in Figure 8.

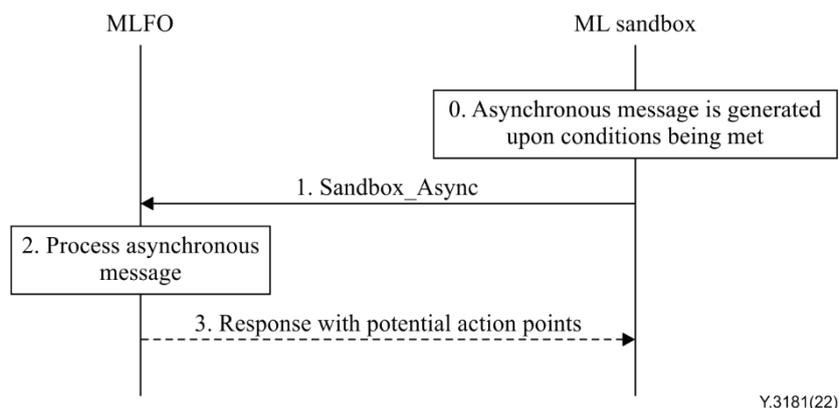


Figure 8 – ML sandbox asynchronous messages

Prerequisite: Asynchronous messages are generated upon certain conditions that are met, which are specific to the different types of events.

- 1) Upon meeting the trigger conditions, the ML sandbox sends an asynchronous message to the MLFO.
- 2) The message is processed and potential action points are defined.
- 3) The MLFO sends a response to the ML sandbox.

The sandbox asynchronous message codes are defined in Table 8-9.

Table 8-9 – Definition of the ML sandbox asynchronous message codes

Code	Parameters	Conditions	Description
SANDBOX-ASYNC-001	Status code / Error code / detailed report (conditional)	Threshold-based alert is fired / misbehaviour is detected / keep-alive message	Report the health status of the simulation components
SANDBOX-ASYNC-002	Update type / changelog / additional information on implications of the update	Update on the simulation components is notified /discovered to/by the ML sandbox	Report an update on security, accounting and licensing requirements of simulation components
SANDBOX-ASYNC-003	Alert type / forecasted results / additional information on the potential failure points	Threshold-based alert based on trend analysis (e.g., service at risk)	Proactive behavior trend identification of simulation components and sandbox resources
SANDBOX-ASYNC-004	Report type / Updated simulation component metadata	Change in simulation component is noticed	Report an update on the simulation component metadata
SANDBOX-ASYNC-005	Report type / Updated level of intelligence of simulation components	Change in the intelligence level of a simulation component	Report the simulation environment intelligence level

9 Security considerations

This Recommendation describes an architectural framework for the ML sandbox in the context of integrating machine learning in future networks including IMT-2020: therefore, general network security requirements and mechanisms in IP-based networks should be applied [ITU-T Y.2701] [ITU-T Y.3101].

It is required to prevent unauthorized access to, and data leaking from, the ML sandbox, whether or not there is a malicious intention with the implementation of appropriate mechanisms, such as those for authentication and authorization and external attack protection.

Bibliography

- [b-IEEE 802.11] IEEE 802.11 WIRELESS LOCAL AREA NETWORKS (2021), The Working Group for WLAN Standards.
<<https://grouper.ieee.org/groups/802/11/>>
- [b-IETF RFC 4180] IETF RFC 4180 (2005), *Common Format and MIME Type for Comma-Separated Values (CSV) Files*.
<<https://datatracker.ietf.org/doc/html/rfc4180>>
- [b-IETF RFC 8259] IETF RFC 8259 (2017), *The JavaScript Object Notation (JSON) Data Interchange Format*.
<<https://www.rfc-editor.org/rfc/rfc8259>>
- [b-Castelli-GANs] Castelli, M., Manzoni, L., Espindola, T., Popovič, A., & De Lorenzo, A. (2021). *Generative adversarial networks for generating synthetic features for Wi-Fi signal quality*. PloS ONE 16(11): e0260308.
<<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0260308>>
- [b-Riley-ns3] Riley, G. F., & Henderson, T. R. (2010). *The ns-3 Network Simulator. Modeling and Tools for Network Simulation* pp. 15-34. Springer.
<https://link.springer.com/chapter/10.1007/978-3-642-12331-3_2>
- [b-XML] W3C Recommendation (2008), *Extensible Markup Language (XML) 1.0 (Fifth Edition)*.
<<https://www.w3.org/TR/xml/>>
- [b-Zhang] Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2019). *Machine Learning Testing: Survey, Landscapes and Horizons*.
<<https://arxiv.org/pdf/1906.10742.pdf>>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems