ETSI GR ZSM 011 V1.1.1 (2023-02)

Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects

Disclaimer

The present document has been produced and approved by the Zero-touch network and Service Management (ZSM) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/ZSM-011_IntentDrv

Keywords
automation, autonomic networking, generic

ETSI

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

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

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

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

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

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Zero-touch network and Service Management (ZSM).

Modal verbs terminology

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

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

1 Scope

Intent-based management enables simpler, more user-friendly expressions of input information, and higher flexibility in automation. Intent is a key enabler to increase automation and make management simpler; therefore the present document investigates the potential use of intents as key enabler for enhancing autonomous network and service management within ZSM framework. It provides a formal definition of intents and a list of principles of intent-driven management, leveraging existing standardization work. Some use cases are also included in the present document to provide examples of management domains where intents are applicable and capabilities that may be needed. Intent-driven management within the ZSM framework is investigated and the concept of an intent management entity is introduced, which is responsible for the life cycle management of intents and the exchange of intents between different management domains. The present document also maps the intent management entity with the concept of closed loops that is specified in ETSI GS ZSM 009-1 [i.11]. Intent modelling is also investigated, and two different approaches are proposed. The present document defines intent life cycle phases and a state diagram, together with a set of (mandatory and optional) interface capabilities that are needed for the life cycle management of intents. Finally, additional aspects such as conflicts between intents, intent translation, and intent testing are investigated. The present document outlines potential future work based on the topics explored and the critical areas that were identified in the present document.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

[i.1] ETSI GR ZSM 005: "Zero-touch network and Service Management (ZSM); Means of Automation".

[i.2] TM Forum IG1230: "Autonomous Networks Technical Architecture v1.1.0".

[i.3] IETF RFC 9315: "Intent-Based Networking - Concepts and Definitions".

[i.4] ETSI TS 128 312: "LTE; 5G; Management and orchestration; Intent driven management services for mobile networks (3GPP TS 28.312 Release 17)".

[i.5] TM Forum IG1253: "Intent in Autonomous Networks v1.2.0".

[i.6] TM Forum IG1253A: "Intent Common Model v1.1.0".

[i.7] TM Forum IG1253B: "Intent Extension Models v1.1.0".

[i.8] TM Forum IG1253C: "Intent Life Cycle Management and Interface v1.1.0".

[i.9] TeraFlow Project: "Secured autonomic traffic management for a Tera of SDN Flows".

NOTE: Available at https://www.teraflow-h2020.eu/.

[i.10] TeraFlow Project - Scenarios.

NOTE: Available at https://www.teraflow-h2020.eu/node/161.

[i.11] ETSI GS ZSM 009-1: "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers".

[i.12] W3C® Recommendation 25 February 2014: "RDF 1.1 Concepts and Abstract Syntax".

NOTE: Available at https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[i.13] ETSI GS ZSM 007: "Zero-touch network and Service Management (ZSM); Terminology for concepts in ZSM".

[i.14] Dave Lenrow: "Intent: Don't Tell Me What to Do! (Tell Me What You Want)".

NOTE: Available at https://www.sdxcentral.com/articles/contributed/network-intent-summit-perspective-david-lenrow/2015/02/.

[i.15] ETSI GS ZSM 009-2: "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 2: Solutions for automation of E2E service and network management use cases".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS ZSM 007 [i.13] and the following apply:

autonomous entity: part of a network that is capable of making and actuating decisions within its specified degree of autonomy and area of influence

NOTE: In the ZSM Framework a Management Domain is an example of an autonomous entity.

intent: formal specification of the expectations, including requirements, goals, and constraints, given to a technical system (see TM Forum IG1230 [i.2])

intent handler: logical entity that receives intents (i.e. the intent information objects) and handles them in the domain that is responsible for that intent's fulfilment

NOTE: An intent handler is not allowed to modify and/or remove an intent but can reject to fulfil it. It fulfils the requirements and goals, based on the resources and solutions it has available once it has accepted the intent. An intent handler reports back to the intent owner regarding the intent fulfilment.

intent information object: information object that represents a specific set of requirements, goals and constraints which are structured according to the intent IOC

intent information object class: object class that describes the type, structure and relationships of the information elements that specify the requirements, goals, and constraints of an intent

intent management entity: autonomous entity in a domain that can play the role of intent owner and/or intent handler and is capable of making and actuating decisions to fulfil intents

intent negotiation: procedure involving an intent owner and an intent handler where the intent fulfilment terms are settled prior to the intent being accepted by the intent handler

NOTE 1: Alternatively, an intent negotiation could also result in a rejection of the intent.

NOTE 2: An intent handler is an autonomous entity in a domain for the aspect of intent fulfilment.

intent object instance: unique managed object instance that is instantiated at the intent handler (MnS producer) based on the information of intent requirements, goals and constraints sent to the intent handler (MnS producer) by the intent owner (MnS consumer)

intent owner: logical entity that originates intents (creating intent information objects) and is responsible for managing its lifecycle. Ideally only an intent owner is allowed to manage the intent lifecycle

opportunity cost: cost of a particular good or service compared to an alternative

NOTE: Opportunity cost is a term used in economics. When consumers or businesses make the decision to purchase or produce particular goods, they are doing so at the expense of buying or producing something else. This is referred to as the opportunity cost.

producer utility: total benefit for a producer to supply a good or service

utility: total satisfaction received from consuming a good or service

NOTE: Utility is a term used in economics. Economic theories based on rational choice usually assume that consumers will strive to maximize their utility. The utility is subjective. It depends upon the mental assessment of the consumer and is determined by several factors which influence the consumer's judgment.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS ZSM 007 [i.13] and the following apply:

CFS	Customer Facing Service
IOC	Information Object Class
MnS	Management Service
OPEX	OPerational EXpenditure

4 The concept of intent-driven management

4.1 Introduction

In an autonomous networks management framework that is purely intent-driven, all goals and expected behaviour are defined with intents. The management framework will only perform operations that relate to the fulfilment and assurance of an intent, which means that all goals - including those that may have been considered "common sense" in human-operated systems - are to be expressed as intents.

An intent in an autonomous management framework is expressed declaratively - that is, as a goal that describes the properties of a satisfactory outcome rather than prescribing a specific solution. This gives the framework the flexibility to explore various solution options and find the optimal one. It also allows the framework to optimize by choosing its own actions and decisions, e.g. exactly which service to instantiate, or configuration to make, that will ultimately maximize utility.

Unlike traditional software systems, where requirements are analysed offline to detect and resolve conflicts prior to implementation, intents are added to an autonomous framework during runtime. Adaptation to changed intent as well as conflict detection and resolution are therefore essential capabilities of an autonomous framework.

Expectations originate from contracts or business strategy and remain constant when the underlying framework is replaced or modified. Consequently, when setting up the intents, it is important that they are formulated in an infrastructure-agnostic form, so that they can be transferred across network and infrastructure implementations, i.e. vendor, technology, and operator agnostic.

As described in clause 5.2, intents can be used for interactions between management service consumer and management service producer in intent-driven management. From the management service consumer's perspective, an intent that is used in such an interaction is agnostic to the management service producer's infrastructure and the resources that are ultimately used in the intent fulfilment.

In short, the intent establishes a universal mechanism for defining expectations for different layers of network operations. It expresses goals, utility, requirements, and constraints. It defines expectations on service delivery as well as the behaviour of the autonomous management framework and the underlying managed network.

An original intent will be transformed and decomposed when transferred between different domains. At each stage it can be decomposed into several new declarative intents and also, partly or completely, transformed into various actions.

In addition, the decomposition and transformation will happen not only when the intent will be transferred between different domains but also between the different levels and layers of operational management to define for example needs, requirements, constraints, and targets.

Although purely intent-driven management frameworks are foreseen, it is more likely that intent-driven management will complement the traditional imperative (not intent-driven) management solutions.

4.2 Definition of intents

4.2.1 Introduction

ETSI GR ZSM 005 [i.1] introduced intent-based approach as a means of automation. Clause 4.3.0 of ETSI GR ZSM 005 [i.1] summarizes how the notion of intents emerged in the telecommunications industry and how some of the main academic and industry work have adopted this concept in the area of network and service management and automation. Also, the later TM Forum's IG 1230 [i.2] presents the concept of intent and the evolution of the concept over time in clause 5.7, showing mainly how intent is defined and used in different standardization organizations.

The notion of intent as a concept and its role in the telecommunication industry has evolved over time from being policy-centric towards being a means for declaration and communication of goals, requirements, and constraints to parts of a technical system, such as a management system. The setting of intent is often linked to humans' expectations and desires, but it can also be used to express goals to be exchanged between machines or within an autonomous system.

IRTF NMRG [i.3] has defined intent as "*A set of operational goals (that a network should meet) and outcomes (that a network is supposed to deliver) defined in a declarative manner without specifying how to achieve or implement them*".

In ETSI TS 128 312 [i.4], intent is defined as "a desire to reach a certain state for a specific service or network management workflow". Besides that, "*an intent specifies the expectations including requirements, goals and constraints given to a 3GPP system, without specifying how to achieve them*".

The same general definition has been adopted in TM Forum's IG 1230 [i.2] as a baseline. TM Forum's work emphasize that from the user's perspective, an intent expresses the expectation the user has with respect to the behaviour of the system. Intents should be used to convey the goals needed to ultimately fulfil humans' expectations. In this sense, according to TM Forum's IG 1230 [i.2], an intent can also be defined as "*the formal specification of all expectations including requirements, goals, and constraints given to a technical system*".

Based on the definitions presented above, there is a common understanding in the industry that an intent is a knowledge object that is used to describe the expectations to a system in a way that allows autonomous operations to be performed by the system receiving such intents. The ZSM framework should use the same definition when applying an intent-driven approach in the zero-touch networks and services management.

Therefore, the adoption of intent definition as in TM Forum IG1230 [i.2] is proposed:

> *"Intent is the formal specification of the expectations, including requirements, goals, and constraints, given to a technical system".*

4.2.2 Principles of Intent

Properties and Implications of Intent are well defined in [i.2] and some others of the previously mentioned documents.

As conclusion the following principles of intent are identified for ZSM:

1. Intent establishes machine-processable knowledge

Intents are formal specification created for a technical system, which means that they establish machine-readable knowledge to be considered by the autonomous management. In an autonomous network, where business goals and requirements can change dynamically and the operations need to quickly adapt without human intervention, the business objectives of the operators as well as the expectations of the customers and users need to be conveyed in some means of knowledge, and this is the main purpose of intents.

2. Intent is declarative, so it leaves any implementation detail internal to the solution provider

Intents define goals, requirements, and constraints, which should be provided in a declarative form. The definition excludes all imperative implementations and solutions aspects. Therefore, intents leave out the implementation of the details of how the network and service is operated to the internal management capabilities of the autonomous network. In this respect, artifacts such as workflows, policies, decision rules, etc. are still needed to realize intent-driven autonomous networks. Such tools will be used internally to the autonomous systems according to network operator's strategies.

3. Intent is focused on expectations of the results for the consumer

Intents focus on a specification of expectations, reflecting the idea that intents are expressed from a consumer's perspective. Intents can originate directly from humans, e.g. customers or operators using intents to communicate to an autonomous system their expectations. It is the job of the autonomous system to fulfil those expectations, while the intent originator may play a supervisory role. Intents can also be generated internally within the autonomous system and can be used between the autonomous entities to influence the details of the specific wanted behaviour and contribute to the overall fulfilment of human expectations (expressed by intents initially created).

4. Intent is formally expressed so it is machine-processable and readable for human

Finally, one of the most important aspects of intents is that they are formally expressed so that they can be interpreted by machines as well as by humans. The sender and receiver of intents need to agree in their interpretation; therefore, there should be no ambiguity in their meaning. The formal expression and unambiguous meaning of intents can be achieved by well-defined information models that completely define the semantics and vocabulary that is required for the operation of each autonomous system that uses intents. The intent models that can be defined in the ETSI ZSM scope are presented in clause 5.4.

5. Intent supports complete automation of intent owner-intent handler interactions as well as of intent-defined service delivery

Intents are abstract, which allows them to be formulated by intent owners without requiring intent owners to learn details of the intent handler's managed entities. This, combined with e.g. suitable intent owner-intent handler interface operations and closed loop-based instantiation and maintenance of services by intent handlers, supports complete automation of intent owner-intent handler interactions and of operations through to service delivery and maintenance.

4.2.3 Ensuring trust in intent-driven autonomy

As expressed with principle 2 above, all implementation details and insight to the intent processing are left to the particular solution of an autonomous intent handling.

To address the expectations of a guaranteed service quality and most secured processes service providers require a high level of control, which raises expectation to get more insight into the intent handling process itself. Therefore, additional capabilities can be optionally enabled.

1. Intent handlers offer optional insight to the intent handling process

Intent handlers may expose the sequence of generated actions or operations, when requested by an authorized user.

2. Intent handlers offer optional explicit intent verification

An Intent can e.g. be explicitly tested at any time if intent expectations are still met, when requested by an authorized user.

4.2.4 Additional aspects of Intent

The introduction in clause 4.1 describes intent as a key enabler in a management framework for autonomous networks. When coming to the more concrete definition how an intent can be expressed (i.e. in an intent information object), also looking at the process of intent-handling within a ZSM framework, different business goals may be considered.

Table 4.2.4-1

#	Goal	Means	Description
a)	Autonomy	Service Abstraction	Avoiding service knowledge for the consumer, express the need, not the concrete service, let the producer decide autonomously about concrete service and how to build it
b)	Human Language	Language Translator	Allows to formulate an intent with human language, to be translated into machine readable intent information object
c)	Flexible Offerings	Parameters to Negotiate	Allows to negotiate for example quality of a service
d)	Provide Best Producer	Intent Request Broker	Find the best service producer/intent handler for the purpose
e)	Time to market	May vary	Enable runbook generation at runtime, avoid coding of runbook workflows

Service abstraction to enable autonomy as in a) is the most relevant aspect of intent and the focus of the present document. An intent handler can decide autonomously on the concrete service instantiated, how this is assembled and how its quality is assured. In the simplest case, the structure of an intent information object, may be similar to a CFS (Customer Facing Service) specification, catalog based, just at an abstracted level. The handler decides on the concrete service and resource composition and its quality assurance.

The full value of an intent comes with goals b), c) or d) it may support human natural language, negotiation e.g. about the quality of service, or find the best service producer for an intent. All these goals are supportable, given the intent information object has the right structure to express the expectation of the intent consumer to the intent producer.

However, looking at the growing complexity of networks, the fast-growing number of possible service offerings, the effort to implement the processing of new intent offerings has to be seen as well as shown with goal e). Therefore, simplification to implement intent processing is another goal, to lower cost of automation and reach faster time to market. The structure of an intent information enables simplification for the consumer, but not necessarily for the producer. Hence the present document touches also implications to the intent management entity implementation.

4.3 Examples of use cases considered

4.3.1 Introduction

This clause introduces use cases where intents play an important role in the autonomous service and resource management. The list of use cases is non-exhaustive and is meant to illustrate the use of intents in different scenarios and to provide examples of management domains where intents are applicable and capabilities that may be necessary for the realization of intent-driven management.

4.3.2 Automotive use case

4.3.2.1 Description

In the automotive use case, mobile operators not only provide the connectivity to the connected cars, but also deploy MEC and cloud infrastructure along the Transport Network (TN) to host the CCAM (Cooperative, Connected, and Automated Mobility) applications. The huge scale and diversity of CCAM services impose 3 main requirements for TN services: low-latency, high-capacity, and massive flow management. For example, in Beyond-5G (B5G) networks, the objective is to collect telematics and driver behaviour data and analyse it to ensure the vehicle's performance, efficiency, and safety. It is estimated that each car will produce and send 25GB of data to the cloud every hour. MEC is deployed to distribute the functionalities between the edge and the core clouds to reduce the number of flows and the capacity required, thereby lowering the E2E latency.

TeraFlow project [i.8] (funded by the European Commission under the Horizon 2020 Programme) is working on smart connectivity beyond 5G. It aims to build a new type of secure cloud-native SDN controller by integrating current NFV and MEC frameworks and have new features for flow management at the service layer.

The TeraFlow OS is designed to unify the management of computing, storage, and networking resources; deploy integrated services (e.g. provision of cloud & edge computing resources and connectivity between them); and optimize the cloud and network resources in an integrated way.

The E2E CCAM services span multiple TN domains, each of which is deployed with a per-domain TeraFlow OS instance. In each TN domain, the physical infrastructure is virtualized to create multiple co-existing virtual TNs (VTNs) with specific QoS.

These TeraFlow OS instances cooperatively manage the E2E services composed of multiple E2E VTNs.

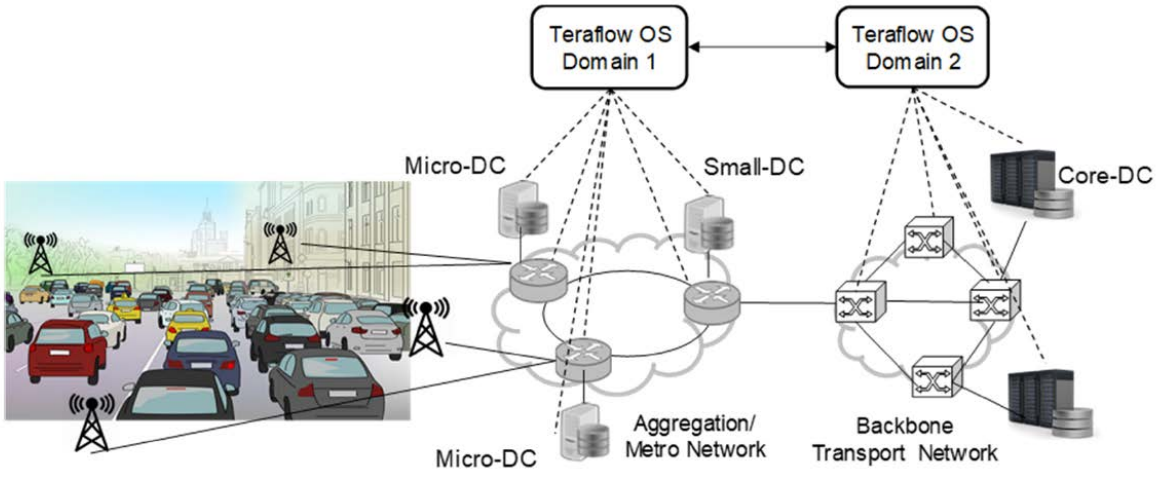


Figure 4.3.2.1-1: TeraFlow Automotive use case scenario (source: [i.9])

4.3.2.2 KPIs

The main KPIs for the CCAM services in E2E are:

- Resource efficiency.
- Multi-tenancy support.
- Latency.
- Positioning.
- Trust/privacy.
- OPEX reduction.

These KPIs cover both technology and business aspects. They are applied to both TNs and cloud infrastructure, which are administrated by different service management systems. To assure the specified KPIs, coordination is required:

- Internally: between TeraFlow OS instances to offer TN slicing services.
- Externally: with 5G network slicing management entities (e.g. ETSI ZSM, 3GPP, ETSI NFV MANO, ETSI MEC) to offer 5G slicing services.

The assurance of these KPIs require different actions. For example, QoS-based KPIs are related to network configurations. Business KPIs (e.g. OPEX reduction) are specified by vertical customers and need to be translated into the network service KPIs. KPIs on security and privacy need to be assured by deploying proper technologies, e.g. security, isolation, etc.

4.3.2.3 Intents

The above KPIs are requirements that can be represented as part of the intents provided by automotive verticals. Such intents will be sent to the customer-facing service portal (e.g. TeraFlow OS or other CSP portals), which are the ZSM framework consumers.

One important step is to interpret these intents into domain-specific intents:

- RAN domain intents.
- CN domain intents.
- MEC domain intents (if MEC is deployed).
- TN domain intents. Note that TN domain intents are defined for TN slices, which then need to be further decomposed into:
  - Access segment intents.
  - Aggregation segment intents.
  - Metro segment intents.
  - Core segments.

The domain-level intents should be technology-agnostic, e.g. TeraFlow OS will support a technology-agnostic NOS LCM, e.g. with deployment, upgrade, and migration. Then, the domain-level intents will be translated into technology-specific intents and realized by individual domains.

Two key capabilities are needed for intent-based management in this UC:

1) Intent decomposition and interpretation/translation across layers:
   a) Intent decomposition: from customer-facing intents (business-oriented and E2E-based) to network-facing intents (technology-oriented, network-domain-based, and technology-agnostic) → decompose the E2E business intents into network-domain intents.
   b) Intent translation: from network facing intents (technology-agnostic) to resource-facing intents (technology-specific).
2) Intent assurance:
   a) Certain monitoring mechanisms are demanded to monitor "intents" at all layers (customer-, network-, and resource-facing).
   b) Measurement and monitoring of intents should be supported.

NOTE: This use case deals primarily with resource management, but the KPIs, the management domains and the key capabilities listed above are also applicable for the management and optimization of managed services.

4.3.3 Cloud private line services

4.3.3.1 Description

Cloud Private Line (CPL) services connect cloud service users to edge or cloud data centres, and edge or cloud data centres to each other, with deterministic connection performance. They may represent point-to-point, point-to-multipoint, multipoint-to-point, or multipoint-to-multipoint connectivity service topologies, and may be connected or connection-oriented in nature. Data flow mapping to CPL services is port-, packet- or frame-oriented and CPL services may or may not include inspection and differential mapping of packets or frames to connection services at connection service ingress ports. Service and/or supporting technology types include Ethernet, MPLS (Multi-Protocol Label Switching), OTN (Optical Transport Network) and DWDM (Dense Wavelength Division Multiplexing), solely or in combination. Collections of CPL services may be considered as Transport "Slices" and defined and provisioned collectively; here CPL services are assumed to be defined and provisioned individually.

Figure 4.3.3.1-1 illustrates an example of a CPL service infrastructure that uses EOO (Ethernet-over-OTN) technology. User data (e.g. IP packets or Ethernet frames) are carried by Carrier Ethernet Services, which operate over Ethernet Virtual Connections (EVCs). The EVCs provide service OAM (Operations, Administration and Maintenance), while service isolation and traffic protection are offered by the underlay OTN services. Right-sizing and topological configuration of service tunnels, or groups of tunnels, is provided by reconfigurable ODUFlex and DWDM technologies.

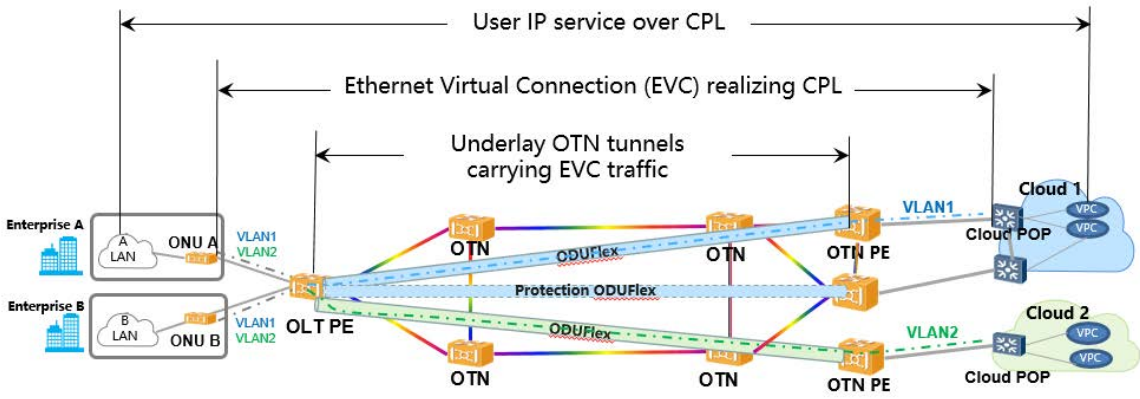


Figure 4.3.3.1-1: Illustration of a CPL service infrastructure based on EOO (Ethernet-over-OTN) technology

Data centres may be operated by the CPL service customer, by the CPL services provider, by some other service provider(s), or by any combination of these. CPL service traffic consists in machine-to-machine data flows with a range of characteristics. Some data flows are essentially continuous, may require low or medium bandwidths, and may be anywhere from relatively latency-insensitive to highly latency-sensitive (e.g. synchronous data mirroring). Other data flows may comprise block data transfers, of varying sizes and completion time requirements, may occur on varying schedules, and may require small to very large bandwidths; they may also have varying latency sensitivities. The application drivers of individual data flows may depend on a range of application circumstances that may vary in time. Even CPL service availability and restoration requirements are variable and derive from application requirements associated with particular data flows.

There is demand, in multiple market segments, for dynamically user-driven mass-customized CPL services, having deterministic connection performance, when in operation, to serve these requirements. Such a paradigm would replace both static port-to-port "large pipe" private line services and statically-configured services that rely on statistically-based sharing of transport resources among data flows that require - perhaps significant - overprovisioning of resources to prevent service performance degradation under unfavourable aggregate data flow conditions. This new paradigm is useful to both service consumers and service providers, as: services may be closely matched to specific needs; service performance is deterministic in operation, providing e.g. determinism in block data transfer times; service delivery is network resource-efficient, as resources may be allocated to closely match the minimum detailed needs of every service; and services may be better-monetized, as no service parameter needs to be "given away free". Obviously, however, a dynamically mass-customized service paradigm, operated at any reasonable scale, requires a high degree of automation of service delivery and maintenance processes.

Dynamically mass-customized CPL services are driven operationally by CPL service consumers, either semi-manually (e.g. through a user-facing provisioning portal) or - more usefully - directly by consumer scheduling software systems. Intent is obviously a useful API and service-driving paradigm to support such capability.

4.3.3.2 Intent parameters

Parameters relevant to CPL service intents include:

- Service end-points.
- Service connection topology type.
- Service ingress flow identifiers, if applicable.
- Bandwidth (e.g. minimum, range, etc.).
- Latency (e.g. maximum, range, etc.).
- Scheduling parameters (e.g. start/stop/resume schedules, overall or specified per-parameter).
- Availability (e.g. minimum time-based availability percentage).
- Restoration (e.g. maximum time-to-restore on failure).

Typically, CPL services will be delivered by one or more Transport Management Domains (MDs). In a fully ETSI ZSM-compliant framework, the CPL service Intent API would lie between a service consumer system, or some intermediating aggregator, and the provider E2E MD. Interactions between the E2E MD and the Transport MD(s) could be of any type: intent-driven, service-based or resource-based. See Figure 4.3.3.2-1.

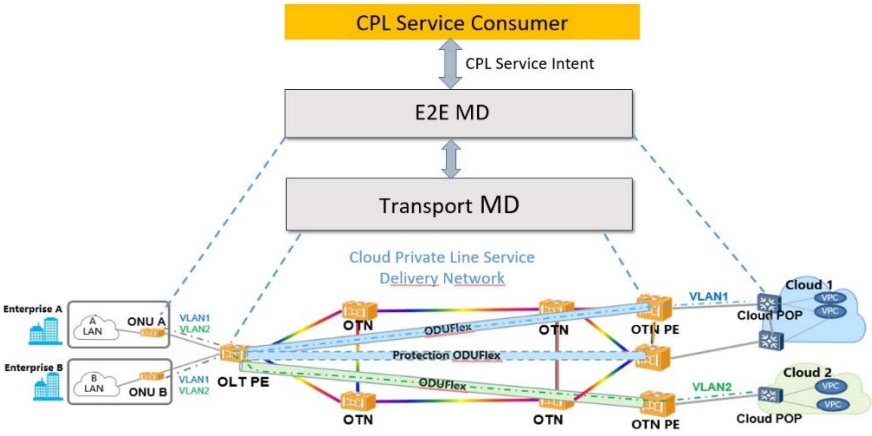


Figure 4.3.3.2-1: Illustration of a CPL service infrastructure based on EOO (Ethernet-over-OTN) technology, showing ETSI ZSM domains and APIs

Economic considerations - e.g. service pricing - could in principle be a component of an intent expression, in terms of limits expressed by the intent owner, possibilities returned in "negotiation" by an intent handler, etc.

Some existing service-based API models reflect similar parameters to those listed above, and thus may be usable - directly or with appropriate extensions - as the basis for CPL service Intent APIs. The operational difference between service-based and intent-driven systems lies in the strict separation of information regimes between owner and handler domains (per clause 4.3.3.3) and in service delivery handling, through closed loops, by Intent Management Entities (IMEs - per clauses 5.1 and 5.2).

4.3.3.3 Intent Translation

It is a core principle of intent-driven system interactions that intents are expressed solely in terms natively comprehensible to the intent owner; intent handlers (i.e. IMEs) are responsible for closing any lexical or other comprehension gaps through intent translation: see clauses 4.3.3.3 and 5.8. Therefore, all the intent parameters suggested in clause 4.3.3.2 are to be expressed in terms intrinsically known to the service consumer system. Many parameters are expressible in universally-comprehensible metrical terms, so that translation is not required. Many parameters may also be encapsulated by service "class" aliases, whose definitions may, e.g. be shared by straightforward administrative (i.e. "offline") processes. Service end-point and flow identifier definitions may or may not be intrinsically comprehensible to both owner and IME. For example, expressed service end-points may represent service consumer equipment end-points that lie outside of the IME's resource domain(s). In such cases, the IME associated with the Intent API has to identify corresponding provider end-points. Both end-points and flow identifiers may, in principle, be represented on the owner side by aliases corresponding to specific - and potentially variable - sets of end-points or flow identification parameters. IMEs are responsible for maintaining appropriate "mappings" in such cases; see clause 5.8.

4.3.3.4 CPL service delivery by Intent Management Entities

As discussed in clause 5.2 both initial and continuing resource selection and allocation to meet expressed service intent requirements is the responsibility of IMEs and their service-specific Intent Instantiation & Maintenance (I&M) closed loops. Such selection and allocation may be influenced by provider-domain intents, policies and considerations including resource efficiency maximization, notions of consumer and service relative priorities, etc. Resource allocations may be changed during service operation provided that resulting service interruptions are within the limits established by the availability and restoration components of affected service intents. Note that - assuming that reasonable numbers of services admit interruption intervals beyond resource reallocation time scales - this may represent a substantial "improvement" over traditional private line service paradigms, as it allows for ongoing service performance "upkeep" as well as ongoing use-of-resource optimization on the part of the provider.

5 Intent-driven management within the ZSM framework architecture

5.1 The role of Intent Management Entity

The ZSM reference architecture is service based and intent-driven management will require to be introduced in a cooperative manner. For that purpose, the following notions are introduced in the present document, which help to investigate how the constructs of intent management work and can be built. Since the ZSM architecture is service-based, a future normative specification would need to focus on the management services and corresponding interfaces.

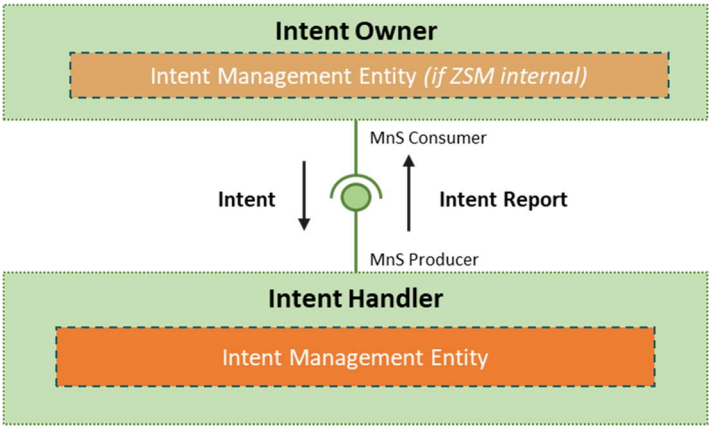


Figure 5.1-1: Notions of Intent management in the ZSM Framework

An intent owner is the role of the intent originator. It is responsible for managing the intent lifecycle, by sending lifecycle requests (as MnS consumer) to an intent handler (as MnS producer). The intent handler has the role to fulfil the intent, as expected by the intent owner. The intent handler may reject to fulfil the intent or offer alternatives. Once accepted, it fulfils the intent-based on services and resources available and reports the results back to the intent owner.

The intent management entity is the element within the ZSM framework providing the MnS for intents. As intent handler it can receive intent requests (MnS producer), as intent owner it can generate intent requests (MnS consumer) e.g. from E2ES MD level.

An autonomous system requires intents to be formally defined in a machine-readable and processable way, but the broad range of considerations involved and their abstract semantics are often difficult to structure. Techniques from knowledge management and semantic modelling enable the creation of an ontology of intent, based on an extensible metamodel. Resource Description Framework (RDF) [i.11] and RDF Schema [i.12] standards can be used for knowledge modelling.

NOTE 1: RDF is one of the possible modelling languages.

Intent specified directly by human operators would require an intuitive frontend, potentially using natural language.

The operation of services within an intent-based network also requires the introduction of intent management entities in the operations stack and functional architecture. An intent-handling entity receives the intents, decides which actions will be taken to fulfil all given intents and implements its decisions.

Intent management entities have a knowledge base that contains the intent ontology. They also have machine-reasoning capabilities to realize knowledge-driven decision-making processes.

Machine reasoning plays a key role in intent management, with its capability to understand abstract concepts from diverse domains and provide precise, specialized conclusions based on precedent and observation. Probabilistic modelling contributes quantification of risk and uncertainty, which is essential to make informed decisions when facing conflicting goals and new situations.

Figure 5.1-2 shows how the intent management entity works. While its implementation is domain-specific, its interface is generic. It receives intents that express all types of expectations. It is equipped with policies (which may include simple rules, or more advanced adaptive mechanisms) and Artificial Intelligence (AI) models that implement the capabilities needed for analysing the network state and finding optimized operational actions based on observations from the operated environment. The intent handler also reports the fulfilment and assurance status of its intents.

The Application Programming Interface (API) of the intent management entity is domain-independent. Its main objective is to manage the life cycle of intents and send reports. Intent is constructed based on a common intent meta-model and is specified according to domain-specific intent information models.

NOTE 2: The above API is northbound and has to be further investigated.

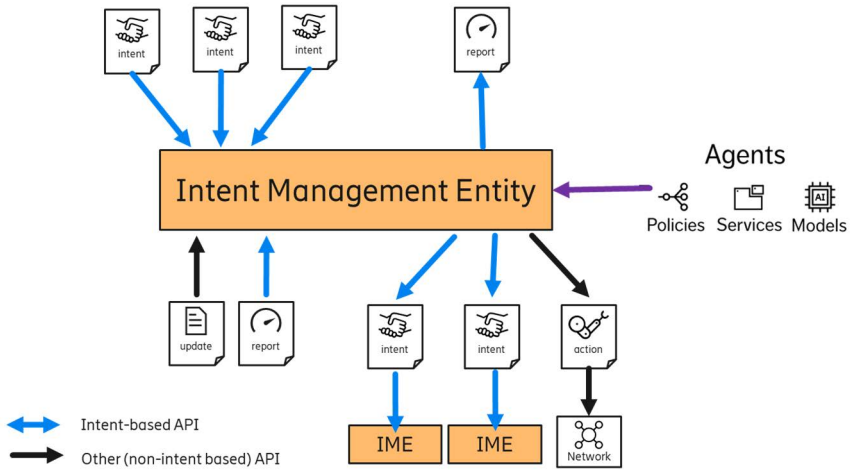


Figure 5.1-2: The intent management entity and its environment

Figure 5.1-3 shows a schematic mapping of Intent Management Entity into the ZSM reference architecture.

The E2E Service MD can either transform and convey an intent towards MDs or interpret the intent and transform it into a set of services offered by the MDs. Reciprocally, the MDs can either transform and convey an intent towards Network resources or interpret the intent and transform it into resource requests.

An MD may also reshape its offered services to match the intent.

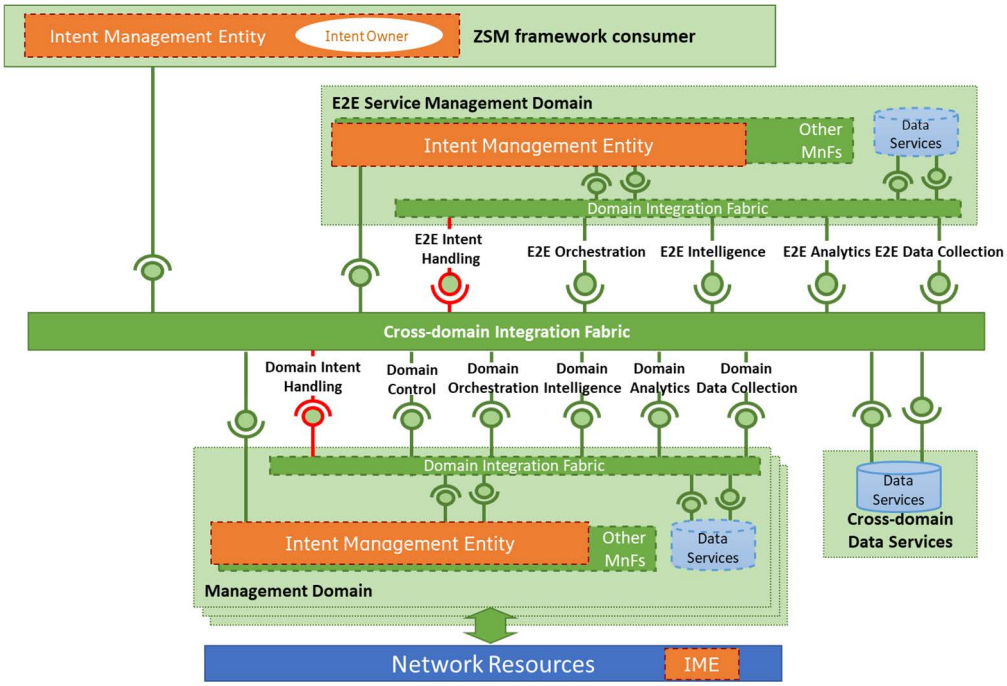


Figure 5.1-3: The intent management entity in the ZSM architecture

NOTE 3: A ZSM framework consumer is typically the intent owner, but an intent owner could, in principle, also be situated in a management domain.

5.2 Mapping of ZSM closed loop concepts to Intent Management Entity operations

In clause 5.1, an example of TM Forum's Intent Handling Entity construct - called Intent Management Entity (IME) in the present document - is used to describe intent-related operations at a high level. Those clauses illustrate the situation of IMEs within ZSM-defined management architectures and domains. They also illustrate the functional relationships both among IMEs that may exist in different domains, and between IMEFs in certain domains and domains which lack IMEs.

In this clause, the operations of IMEs are explored in greater detail, making use of, and referring to closed loop concepts defined by ETSI GS ZSM 009-1 [i.11], based on management services for monitoring, analysis, decision, execution and knowledge.

Several types of intent related control loops are identified by TM Forum. The different control loops related to intent-based operations described in TM Forum's IG1253 [i.5] are shown in Figure 5.2-1.

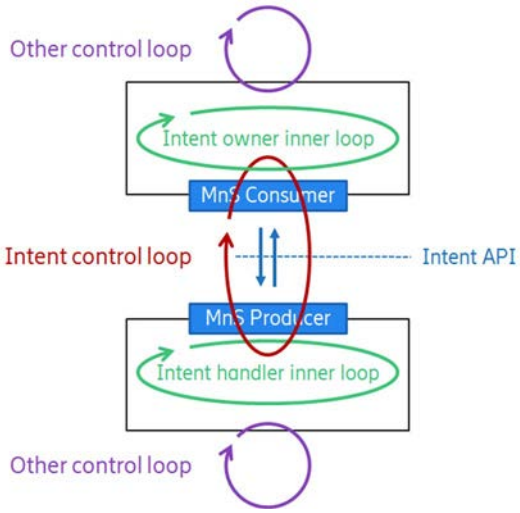


Figure 5.2-1: Control loops related to intent-driven operation
(source [i.5] - Control loops related to intent managers
(Copyright © TM Forum 2022. All Rights Reserved.)
(*source IG1253 Intent in Autonomous Networks v1.3.0, Figure 16.1*))

In Figure 5.2-1, an intent handler (whence, "Intent Handling Entity" and here, "Intent Management Entity") operates a control loop whose function is to produce and maintain the delivery of a service state or outcome, that state or outcome being defined by a service intent expression. The service intent expression is generated by a consumer - an intent owner - and delivered across an Intent API; it defines the closed loop goal per clause 8.1.4.3 of ETSI GS ZSM 009-1 [i.10]. Producing and maintaining the delivery of a service state or outcome is the core function of most IME instances.

NOTE 1: The handler and owner in Figure 5.2-1 simply represent generic domains lying below and above, respectively, a particular Intent API.

An intent owner - the originator of a service intent - operates its own closed loop for life cycle management of the intent and its related service. This involves:

- synthesis of the requirements of a service to be delivered and maintained by an intent handler;
- formulation of those requirements in appropriate intent terms;
- communication of the service intent via an Intent API to an intent handler;
- evaluation of feedback from the intent handler regarding the deliverability or status of a requested or active service; and
- determination of appropriate reactions and intent (service) life cycle staging based on such feedback, as well as on other inputs and information.

Lifecycle stages are considered in clause 5.5 of the present document.

An intent handler operates its own control loop (closed or inner loop), driving actions to deliver on the intent expectations and reports the result back to the intent owner (intent feedback). The intent handler can also be an intent owner, sending further intents to subordinate IMEs, as discussed in clause 5.1.

The "intent control loop" illustrated in Figure 5.2-1 is best understood, in ZSM closed loop conceptual terms, as linked or coordinated owner and handler closed loops. The linkages consist in inputs to closed loop operations provided by one domain/closed loop to another as suggested in Figure 5.1-1:

1) the defined intent is communicated in one direction (from the intent owner to the intent handler) to set service requirements;
2) in the opposite direction (from the intent handler to intent owner), reports (intent feedback) communicate the intent fulfilment and handling status.

Such inputs/outputs, and interactions between control loops and external entities in general, are described in clauses 8.1.5 (interactions between closed loops and external entities) and 8.2 (closed loop coordination) of ETSI GS ZSM 009-1 [i.7].

In many respects - particularly when considering e.g. intent reconciliation - it may make sense to consider that a single IME may serve multiple service intents, and thus host multiple closed loops. One possible composition of an IME is therefore as shown in Figure 5.2-2.

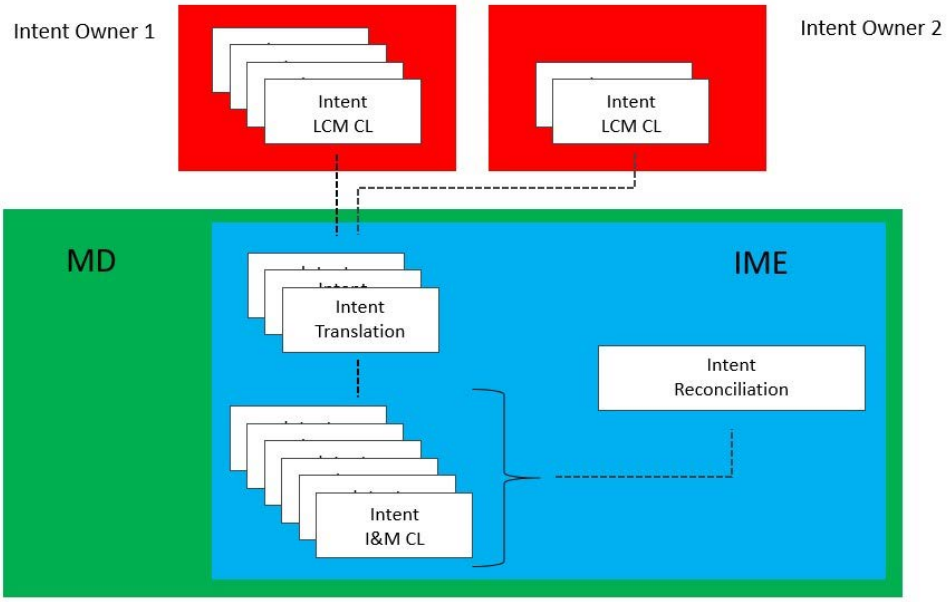


Figure 5.2-2: Key components and possible structure of an Intent Management Entity (IME)

The Intent Instantiation & Maintenance Closed Loops (I&M CLs) shown in Figure 5.2-2 are peer closed loops per clause 8.2 of ETSI GS ZSM 009-1 [i.7]. They are ephemeral managed entities - e.g. made-to-order closed loops per clause 7.4 of ETSI GS ZSM 009-1 [i.7] - associated on a 1:1 basis with service intents generated by intent owners. The functional components serving the various Intent I&M CLs - monitoring, analysis, decision, execution, knowledge - may be fully or partly shared among them, though many work and data flows among components, and governance, life cycle staging etc., may be specific to respective I&M CLs. The I&M CLs act upon managed entities - managed resources, managed services or other closed loops - per clause 7.2.1 of ETSI GS ZSM 009-1 [i.7]. Managed resources (at least) may serve all or multiple peer I&M CLs. Intent I&M CLs are considered further later in this clause.

Intent Translation Functions (ITFs) close information "gaps" between intent owner and intent handler realms, such that Intent I&M CLs have service definition targets comprehensible to them, and such that intent owners receive service performance or status feedback in terms comprehensible to them. The possible need for such functions is described by TM Forum (e.g. in [i.5] - see Figure 5.2-1); ITFs are considered in more detail in this document in clause 5.7. Multiple ITF instances may be required to the degree that multiple independent owner-realm information domains operationally intersect an IME.

The Intent Reconciliation Function (IRF) is a closed loop coordination service, as defined in clause 5.2.5 of ETSI GS ZSM 009-1 [i.10]. The role of such coordination services is well-described in the introduction to that clause:

"To address different situations and coordination needs of the different closed loops, coordination capabilities may include among others:

- Capability to align goals of individual closed loops sharing a given scope.
- Capability to identify different interaction types between closed loops such as cooperation (positive interaction), conflict (negative interaction) or dependency (neutral interaction).
- Capability to identify different types of conflicts between closed loops such as parameters conflict, metrics conflict, or indirect conflict.
- Capability to address the different interactions between closed loops with adequate mechanisms, such as conflict resolution mechanisms.

- Capability to identify before the execution of a proposed action of closed loop that such an action may cause undesired effects to other closed loops or to managed entities (e.g. pre-execution and post-execution coordination, concurrency coordination, etc.)".

The IRF detects and manages conflicts within or among service intents. For example, intrinsic conflicts may consist in irreconcilable terms within or between intent expressions; in another example, conflicts among service intents may arise through competition for fulfilment resources. Further, the IRF balances and reconciles the objectives of service intents with those of other intents - including those generated and managed at the host MD's own "level" (e.g. resource policies per clause 8.1.5.2 of ETSI GS ZSM 009-1 [i.10]). It thus seeks to drive controlled maximization of service utility among consumers as well as maximized conjoint consumer-producer utility. These concepts are discussed in greater detail in clause 5.6 of the present document.

An Intent I&M CL is created by an IME when a new service intent appears at an Intent API and is maintained as long as that intent - itself a managed entity per ETSI GS ZSM 009-1 [i.10] - is maintained by its owner in any stage of life cycle. Its operations are related to those stages as follows:

- The Intent I&M CL has to initially:
  - Identify management and/or resource domains, accessible to the IME's host domain, that are capable, collectively, of delivering the requested service.
  - Find available or create new (e.g. VNF) resources in those domains supporting most complete possible fulfilment of the requested service intent. Reservation of such resources, pending an appropriate activation or similar life cycle trigger, may or may not be prescribed.
  - Reflect the full or partial ability, or the complete inability, to fulfil the requested service, to the intent owner, in appropriate terms. Similar information has to be provided to the IRF, which may - e.g. considering consumer or service relative priorities - coordinate actions affecting other Intent I&M CLs, for example to liberate resources to support the new service request.
- Once the service is instantiated and active, the Intent I&M CL has to:
  - Attempt to maintain operation of the service per the requirements established by the service intent, by monitoring the status of the service and responding to lapses through corrective actions found and undertaken at its discretion - for example, by attempting to find alternative fulfilment resources. Coordination with the IRF is essential, as corrective actions contemplated or undertaken may have impacts on other services. Such actions and impacts may even be directed by the IRF - e.g. according to notions of consumer and service relative priorities.
  - If full service intent cannot be maintained, this has to be reported to consumers in appropriate terms. It is for intent owners to determine appropriate life cycle reactions.
  - Repeat steps already described in the event the intent owner modifies the service intent.

Changes in available resources are among the events which may lead an Intent I&M CL to make changes in the details of how it fulfils its associated service. Note that individual CLs may drive continuing processes of searching for more "efficient" - in some sense defined for or by it (e.g. as a resource policy) - fulfilment. Alternatively or also, IRFs may drive coordinated optimization actions across I&M CLs.

As discussed in clause 5.1, an IME in a given management domain may employ either intent-based or non-intent-based interactions with subordinate management domains. As described earlier in this clause, an IME's I&M CLs are responsible for continually comparing outcomes delivered by subordinate domains with the targets set for them, and for undertaking actions as may be available and required to close any gaps detected. The details of this process are, however, subtly different between the cases of intent-based interactions and non-intent-based interactions.

Where an IME employs non-intent-based interactions with a subordinate domain - e.g. using a resource - or service-oriented interface - the assessment of outcomes may be based on performance reporting, observation (e.g. instrumentation data) or similar means. Where a gap between required and actual outcomes is observed, I&M CLs may assess the shortcoming and either:

a) in the case of a resource-oriented interaction, attempt to direct a reallocation of resources capable of restoring the targeted outcome, or, if unable to fully restore achievement of the targeted outcome, to report the remaining performance gap to the corresponding intent owner;

b) in the case of a service-oriented interaction, either specify an alternative service requirement, or prompt an attempt by the subordinate domain to reallocate resources capable of restoring the originally targeted outcome. Note that this assumes that services are established in a "set-and-forget" manner by the subordinate domain, and that the subordinate domain does not operate a closed loop to maintain the service to the defined targets. If the subordinate domain cannot, finally, be made to fully restore performance of the service to defined targets, the I&M closed loop may report the remaining performance gap to the corresponding intent owner.

In these cases, the entire I&M closed loop exists within the superior domain IME.

Where IMEs and their I&M CLs use intent-based interactions with subordinate domains, those subordinate domains and their IMEs assume primary responsibility for closed loop maintenance of their targeted outcomes. Effectively, I&M closed loop operation is partially delegated to the subordinate domain IMEs. Reactions by the superior domain IME to reported departures from targeted performance by subordinate IMEs (domains), are restricted to stimulating intent life cycle management actions on the part of the corresponding intent owner. Such actions can include e.g. modifying the intent, deactivating the service, etc.

NOTE 2: According to Closed loop goal class defined in Table 8.1.4.3-1 of ETSI GS ZSM 009-1 [i.10], intent can be used as an input for the declarative closed loop goal statement. To realize a mechanism of intent in control loop, the detailed description and properties of "closedLoopGoalStatement" should be specified in future work.

5.3 Intent interactions between different management domains

The ZSM framework has a principle concerning the division of concerns that allows the integration between E2E management and management within multiple MDs, e.g. Radio Access Network (RAN), transport, and Core Network (CN). This principle implies that intents are be handled in a distributed manner between these different layers of management and between different MDs. This division of concerns creates a hierarchy of IME. Regarding this IME hierarchy (focusing on the different layers of management, i.e. vertical operations), in the ZSM framework, an E2E service intent is handled by the IME located at the E2E service MD, which will send service intents to MDs that match the concerns specified in the E2E service intent (intent delegation). Intents can originate directly from ZSM framework consumer, and then domain specific intents are derived (or decomposed) automatically and propagated downwards to specific MDs by its corresponding IME. Furthermore, for all intents that an IME receives inside each MD, it is expected to report the intent fulfilment and handling status back to the source of the intent for all of them.

The scope of operation of the IME within MDs is well-defined. Practically it refers to a set of resources and a corresponding set of operational tasks an IME is responsible for. This way, the IME is the only:

1) authorized agent that can make use of the available resources; and

2) who can make (or coordinate) actions and decisions related to the resources and functions inside this scope.

Therefore, the IME is responsible for the intent fulfilment within this scope. IMEs are restricted to their scope, and they are not allowed to directly interact and interfere with the intent operation that is in scope of another IME (an IME located at RAN domain will not directly manage resources in the transport domain, for example). The IME can however use intent (becoming an intent owner) to put additional requirements on the operation of another IME scope (becoming an intent handler), asking the respective IME to reach a state that fulfils all needs. IMEs are in this respect single points of contact for all operational request within the intent mechanism.

Figure 5.3-1 shows the following inter-domain operation steps:

1) When receiving the intent from the ZSM framework consumer, the E2E service management domain decomposes the E2E intent into multiple MD intents.

2) Delivers the MD intents to each corresponding domain.

3) After receiving the intent, each management domain uses closed-loop automation mechanisms to satisfy the intent.

4) Reports the intent fulfilment information to the E2E service management domain.

5) Based on the intent fulfilment information received, the E2E service management domain integrates the received feedbacks of each domain into E2E intent fulfilment information and reports the E2E intent fulfilment information to the ZSM framework consumer. Moreover, the E2E service management domain determines whether to re-decompose the E2E intent according to the fulfilment information and the capabilities of each domain.

The E2E service management continuously evaluates whether the intent is satisfied and re-decomposes the intent to fulfil the intent, and the E2E intent can be re-evaluated as a whole or just regarding the sub-intent for one of the domains.

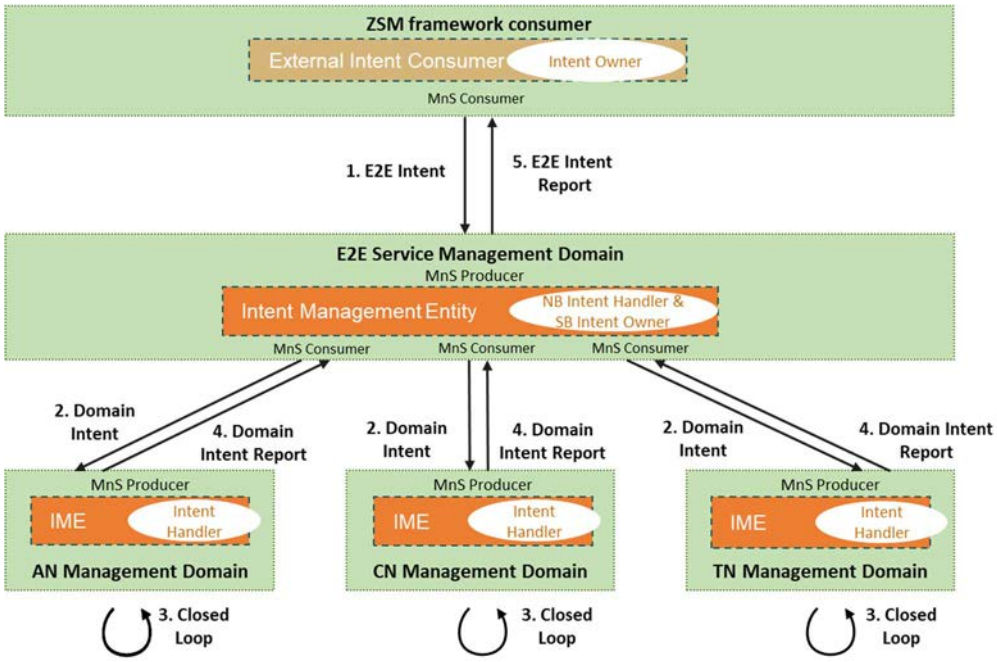


Figure 5.3-1: Inter-domain operations steps using intents and feedbacks between different management domains

5.4 Intent model federation

5.4.1 Introduction

Intents are a list of expectations, referring to operational requirements, goals, and constraints, that are created by an intent management entity and sent to another intent management entity to be considered in autonomous operations.

As defined in clause 4.2, the intent owners and intent handlers need to agree on the interpretation of intents and there should be no ambiguity. The intent models define the expressiveness of intent and the common semantics to enable to two parties to agree on the meaning of the intent.

The intent model has to provide expressiveness to cover all use cases and be understood by all management domains involved. Although the intent models require specific expressiveness to specify the requirements for different domains, incompatible models and interfaces that would require large translation efforts to be used across domains should be avoided.

All intents should have similarities in the general structure and partly also in the required expressiveness. One example is an intent that specifies a functional requirement that something should be delivered. In some cases that can be a service or a product - if such intent is used between the ZSM framework consumer and the E2E service management domain. In other cases that can be a network slice or a network function - if such intent is used between the E2E service management domain and management domain. The target of delivery depends on the scope of the intent management entities involved, but it is the same type of expectation that can be formally defined by a similar intent model. Another example is an intent that specifies a non-functional requirement setting a target based on a KPI. Such intent can specify that the service availability should not fall below 99 %, or that link delay should not exceed 10 ms. The KPI to be considered also depends on the scope of the intent management entities involved, but the different requirements are similarly expressed by a numerical value and a condition (does not fall below, does not exceed, etc.).

Based on the example above, it was concluded that intent modelling requires a common expressiveness that can be used across different management domains, and domain-specific details that can be combined with the common expressiveness. The goal is to facilitate the use of intent-driven operations across management domains and avoid the use of intent models that would diverge and require large translation efforts.

The common expressiveness that all intent information objects should have - regardless of the use case and domains involved - is called intent meta-models. The intent meta-model defines purely domain independent expressiveness, leaving domain-specific expressiveness to intent information models that extend the intent meta-models. These extensions define domain-specific vocabulary and semantics and can be developed independently by standardization bodies, vendors, operators, or any other organization interested in participating in the model federation.

In this respect, when creating a new intent information object, the intent owner has to decide the intent extensions and intent information models that are required to provide the desired expressiveness, depending on the domains involved in the intent-based interaction. The intent extensions and intent information models should be supported by both the intent owner and the intent handler so that the intent information object can be correctly interpreted.

Clauses 5.4.3 and 5.4.4 summarize two different meta-models that can be considered in the present document.

5.4.2 Criteria for selection of intent meta-models

Multiple Models have been proposed for intents and it is necessary for the present document to conclude on the most appropriate model to be developed normatively. It is key that the following criteria be met to select the right model.

- Common structure: Intents should have similarities in the general structure and the model should allow a common structure for different intents.
- Reusability in different organizations (SDOs): The model promoted by ZSM is reusable in and by different Standard Development Organizations that are likely to adopt the ZSM recommendation.
- Expressiveness in different domains: The preferred model supports common expressiveness across different management domains, a change in domain does not require use of a different model or new domain specific extensions but may instead contain attributes that allow domain specific information to be submitted as part of the intent.
- Support for different use cases: Within any one domain the model allows for support of different use cases, e.g. functional requirements on something to be delivered, requirements on setting characteristics on an object, or requirements on setting performance targets (e.g. on a KPI) of an object.
- Support for different layers: The preferred model allows intents on different network-related layers including business and operations layers.
- Minimal ambiguity: The preferred model should minimalize or completely avoid ambiguity. It should ensure that a given information elements submitted as part of the intent cannot be interpreted in two or more different ways.

5.4.3 Intent meta-model from TM Forum

TM Forum IG1253 [i.5] proposes an intent model federation approach where intent information objects can be created based on a set of distinct models that are composed by an intent meta-model and domain-specific extensions. This clause summarizes the intent meta-model defined in TM Forum IG1253A [i.6]. The full description can be found in TM Forum IG1253A [i.6].

The intent meta-model defines common intent artifacts, which are mainly represented by the intent and the intent report classes. All intent information objects are objects of the intent class, and all intent reports are objects of the intent report class.

Intents are defined as the specification of expectations. Therefore, the intent meta-model defines the expectation class, which are detailed requirements and goals expressions.

The expectation class has two properties: *target* and *parameters*. The property *target* allows referring to individual or collections of instances or artifacts to which the expectation is applicable. Examples of targets are a service, resource, or a slice instance.

NOTE: The concept of the term *target* used in TM Forum IG1253A [i.6] has similar meaning as the concept of *managed entity* defined in ETSI ZSM framework.

The property *parameters* define the properties of the targeted instance, and consequently the requirements, goals, or constraints. Examples of parameters are objects defined within a catalog, or numeric values for non-functional requirements.

The expectation class can be extended to represent different types of expectations. The intent meta-model defines a set of specialized expectations:

- Delivery expectation:
  - Expectation that allows pointing at assets that are required, which can be a service that be an application, service, or slice, that need to be delivered. Usually, these expectations are related to pre-defined catalog objects.
- Property expectation:
  - Expectation that requires a property of an artifact instance to be set to a specific value or set of defined values.
- Metric expectation:
  - Expectation that requires a metric of an artifact instance to not fall below or exceed a threshold value. KPIs are examples of such metrics, and this type of expectation can be applied for use cases with performance requirements. The intent meta-model has two metric expectation classes that allow specifying metric-based requirements: MinMetricExpectation and MaxMetricExpectation.
- Allocation:
  - Expectation that specifies requirements on the allocation of instances of related artifacts. Examples of allocation requirements can be that instances should stay in a limited geographical region, or within an administrative domain, etc.
- Coexistence:
  - Expectation that specifies requirements on the coexistence of multiple artifact instances that are used to fulfil a given intent. The coexistence can be in a common environment or context. Examples of coexistence expectation is that two different services have affinity requirements and need to stay in the same data center.
- Sharing:
  - Expectation that specifies if instances of a related artifact can be shared or not between different intents.
- Connection:
  - Expectation that specifies connections that are required for the deployment of related artifacts. Network slices, for instance, may require the connection of network functions and services that can be specified with this type of expectation.
- Reporting:
  - The intent handler is obliged to report the status of the fulfilment of all accepted intent information objects. The reporting expectation allows specifying the detailed conditions for sending an intent report.

Since the intent handler is the only entity that is aware of the progress of the intent fulfilment, reports are always pushed towards the intent owner and not pulled. The push mechanism is used to avoid unnecessary and excessive polling of reports. The intent owner can specify conditions, such as frequency or events, for triggering reports to be sent by the intent handler.

Any intent extension model can add domain-specific modelling artifacts. Examples of such artifacts are additional sub classes of the expectation class defined by the intent meta-model.

In addition to the intent extension models, there are intent information models that add vocabulary to the intent model. For example, an information model can define a set of KPIs and the identifier the KPI is referred to. This information model can be combined with the property expectation defined in the meta-model to create a domain-specific non-functional requirement.

Intent extension models extend the expressiveness of the meta-model by modelling new artifacts, such as new expectation sub classes. While intent information models are independent of the intent meta-model or the intent extension models. Intent information models, such as the definition of domain specific KPIs, do not need to extend any existing intent meta-model artifacts. Therefore, there are already existing information models that can be useful for intent-driven operations and can be leveraged by the industry.

Each intent information object may use domain and use case specific intent extensions and intent information models, which will make that given intent to be of a specific type, e.g. business intent, service intent, resource intent. However, there is no need to explicitly make this type a part of the intent model. Intent information objects can be typed solely for the purpose of documentation and human understanding, but there is no semantic benefit to add this to the intent model. Therefore, the intent and intent report classes do not have sub classes.

5.4.4 Declarative Intent Model

5.4.4.1 Intent Expectation

In the most basic form, a consumer may use an intent to express to the producer to:

"ensure that a Managed Entity O is in a specific state S".

The consumer may desire the same requirements for multiple entities of the same type, in which case the intent may be stated for a list of entities as:

"ensure that Managed Entities $\{O_1, O_2, \ldots O_N\}$ are in a specific state S".

However, the consumer may wish to state different requirements for different types of Managed Entities. It is in that case necessary to distinguish the requirements to be achieved for each type of Entity. Correspondingly, the combination of requirements, goals, and constraints for each type of entity may be termed as the Intent expectation.

5.4.4.2 Desired outcomes as Intent Targets

For a given intent expectation, the state may be expressed in terms of state attributes of the Managed Entity(s), which include the parameters of the Managed Entity(s), the measurable counters and KPIs that characterize the performance of the Managed object(s) or some abstract index that expresses the behaviour of the entity(s). A given intent expectation may state multiple requirements on the same entity or entity type. A consumer may for example require for the cell object(s) that HandoverFailureRate < 2 % and RACHFailureRate < 1 %. The state may also be context specific, i.e. the intent my require a specific state given a specific context. As such with the state as a combination of parameters, KPIs and context, the intent expectation may be stated as:

"ensure that for

Entity O,

parameter_1 is P_1, …., Parameter_n is P_n;

KPI_1 is K_1, …., KPI_m is K_m;

context_1 is C_1, …., context_k is C_k;

"

Each of the Entity State Attributes (parameters, KPIs and context) may be set to be equivalent to a specific value or constrained to a value or a range of values, e.g. as listed in Table 5.4.4.2-1. The combination of state attribute, the condition constraining the attribute and the value or value range to be achieved the attribute is the target, i.e. the target is the tuple:

target = [attribute, condition, value range]

Similarly, the context is the combination of state attribute, the condition constraining the attribute and the value or value range to be considered as described further in clause 5.4.4.4.

Table 5.4.4.2-1: Example intent targets and context for different Managed Entities

Type	Managed Entity	Object State attribute	Condition	Value range
Parameter target	Radio Network	Coverage area	Is at least	40 km radius
KPI Target	Communication Service	User throughput	Is greater than	2 Mbps
Context	Communication Service	Time of day	Is within	6:00 to 22:00 hrs

5.4.4.3 Intents and Managed Entities

The Managed entity(s) for which a given expectation is addressed may listed using their identifier. This may, however, not always be adequate or may be cumbersome for some intents. For example, it may be easier to state "all cells in city ABC" as opposed to listing the individual cells. As such it may be easier to identify the Managed entities by stating the "type of entity" together with the Managed-Entity context information that filters and identifies the desired objects. The Managed-Entity context is in form of a context list whose entries are each a tuple (attribute, condition, value range). For example, in the case of "all cells in a city" there is a single Managed-Entity context, which is the tuple "location, =, city_ABC" to be applied to "objectType=cell".

5.4.4.4 Context and filter information

Each target may be constrained to only be achieved for a very specific set of constraints. For example, the consumer may state that:

> "*ensure that handoverFailureRate < 2 % if Load > 80 %", where the target "HandoverFailureRate < 2 %" is only to be achieved only in the context "Load > 80 %*".

Similar to the target, the context is also a tuple of < attribute, condition, value range > but which the values having a different semantics.

Although contexts and targets have the same structure, to distinguish between what has to be achieved and the context which is only to be considered as required conditions, the Context has to be explicitly stated separate from the target.

EXAMPLE: If the consumer may wish that the Radio Link Failure rate (RLF) is less than 2 % when the load is more than 50 %. If the context (i.e. load > 50 %) is not explicitly stated/modelled as context, the producer could interpret the request to mean (RLF < 2 % and load > 50 %).

For a given expectation, the specific list of targets may be desired to be achieved for given combined contexts, i.e. besides the Target, an expectation may state a list of contexts which apply to all targets within the intent expectation. Similarly, there may be contexts that apply to all expectations within a given intent. Correspondingly, both Intent expectations and intents should be modelled to contain aggregate contexts that apply to all the contained sub elements.

5.4.4.5 Intent fulfilment status

The fulfilment information describes the degree to which a specific aspect of the intent (i.e. either an expectation, a target or the whole intent) has been fulfilled and the related reason(s). The degree of fulfilment is recorded into the `fulfillmentStatus` field. The possible values of the fulfilment include:

- "ONGOING": This is the default status and is the initial status right after the intent has been activated.
- "SUSPENDED": This is the status if the producer decides to suspend the fulfilment of the intent, expectation, or target for whatever reason. This is status needs to be supported by a reason such as the event(s) that were observed when fulfilment was attempted.

- "FAILED": This is the status when the producer decides that the intent, expectation or target cannot be fulfilled. This is status has to be supported by a reason such as the event(s) that were observed when fulfilment was attempted.
- "FULFILLED: This is the status if the intent, expectation or target is fulfilled as desired by the consumer that created the intent.

For some scenarios (particularly for the "suspended" and the "Failed" scenarios), the status has to be supported by extra information describing or related to the status. This extra information needs to be recorded.

5.4.4.6 Class definitions

The descriptions in clauses 5.4.4.1 to 5.4.4.5 imply that five types of information objects are required - intent, intentExpectation, IntentTarget, context and FulfillmentInfo. Objects of type intent and intentExpectation need to be instantiable so these should be classes while objects of type IntentTarget, context and FulfillmentInfo do not need to be instantiated, so these are dataTypes as shown in Figure 5.4.4.6-1.

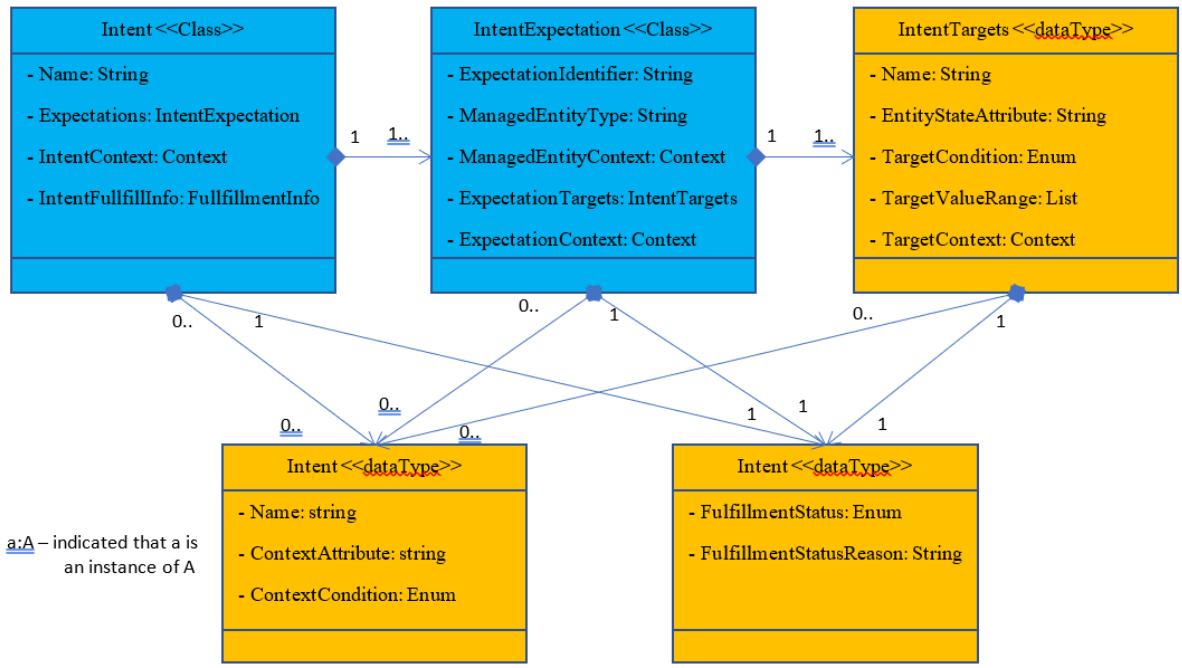


Figure 5.4.4.6-1: Potential information elements of an Intent Information Object

Each information object may be identified by a name or identifier that is of type string. The Intent class includes the list of one or more Expectations to be achieved, the IntentContexts within which the IntentExpectations should be achieved and the FulfillmentInfo Object that is written by the intent handler (MnS producer) to indicate the state of the intent. Each Expectation is an instance of type IntentExpectations while each IntentContext is an instance of generic type Context.

The IntentExpectation class needs to define a list of one or more expectation specific ManagedEntities to which the IntentExpectation applies. The expectation specific ManagedEntities list may be defined by the ManagedEntityType (e.g. a service or Network Slice, or cell) and a set of ManagedEntityContexts that filter from all available entities fitting the ManagedEntityType to define the specific list of entities to which the IntentExpectation applies. For this list of one or more managed entities, the IntentExpectation class then defines a list of ExpectationTargets to be achieved and an optional list of ExpectationContexts, which applies to all ExpectationTargets, within which the ExpectationTargets should be achieved.

Each ExpectationTarget in the IntentExpectation is a tuple of three entries - an EntityStateAttribute which is a String that characterizes the selected ExpectationEntities; a TargetCondition that states the limits within which the EntityStateAttribute is allowed; and the TargetValueRange that states the range of values that applicable to the entityStateAttribute as constrained by the TargetCondition. The ExpectationTarget may have an optional list of TargetContexts, which define the specific contexts within which that specific ExpectationTarget should be achieved.

Each Context, defined for the Intent or ExpectationEntity or IntentExpectation, or ExpectationTarget, is a tuple of three entries - an ContextAttribute which is a String that characterizes an aspect of the Intent, ExpectationEntity, IntentExpectation, or ExpectationTarget; a ContextCondition that states the limits within which the ContextAttribute is allowed; and the ContextValueRange that states the range of values that are applicable to the ContextAttribute as constrained by the ContextCondition.

Similar to the intent, fulfilment may also be tracked for any IntentExpectation and also for any specific ExpectationTarget. As the IntentExpectation and the ExpectationTarget objects need each to have a fulfilmentInfo Object. The FulfilmentInfo has FulfilmentStatus which is an enumeration of possible state values as well as a FulfilmentStatusReasons which may be a string that captures the reasons related to any FulfilmentStatus value.

Further details on the definitions, nature and characteristics of these information objects can be found in annex B.

5.5 Intent lifecycle

5.5.1 Introduction

Intents need to be actively lifecycle managed. Intents are used for interactions between two distinct intent management entities, as described in clause 5.1 and the lifecycle management of a given intent object instance is distributed between these two intent management entities.

The intent management entities can assume two distinct roles: intent owner or intent handler. An intent management entity can assume exactly one of these two roles for each intent object instance and the role will determine the responsibilities and tasks in the life cycle management of the intents.

The intent owner is the originator of the intent. It is the entity that communicates the requirements, goals, or constraints to the intent handler. The intent owner is the only entity and role that is allowed to change the intents that it has generated, and it has to actively remove them from the corresponding intent handlers if the intents are not needed anymore. In exceptional cases, an authorized intent management entity other than the intent owner may be allowed to change or remove an intent. The original intent owner should be informed about this activity.
The intent handler plays an active role in the life cycle of intents mainly during the negotiation and formal definition of the intent after an intent is defined and instantiated.

The intent owner is the main entity involved in the life cycle management.

An intent management entity becomes the intent handler of an intent when it receives that particular intent information object from an intent owner. An intent handler has to consider the requirements, goals and constraints specified in the intent information object when operating the domain and infrastructure. The intent handler is responsible for keeping the owner updated about the intent fulfilment status and progress by sending intent reports.

An intent handler can reject an intent if, e.g. it is not able to understand or not able to fulfil it. Thus, in order to avoid rejection, the intent owner can request intent feasibility or negotiate with intent handler for the best intent expression by exchanging proposals using optional operations on the intent interface, as described in clause 5.6.

There should be a policy defined by the intent owner on how the negotiation should be conducted, e.g. how many iterations are allowed.

5.5.2 Phases of the intent lifecycle

Since the intents are originated at the intent owner, their lifecycle begins even before they are received and instantiated by the intent handler. The intent owner and handler responsibilities are different at each phase of the intents' lifecycle, which are detailed below.

Detection:

In the detection phase an intent owner identifies if there is a need to define new or to change/remove existing intents to set and/or change requirements, goals, and constraints.

Every intent management entity has its own internal requirements, goals and constraints that should be fulfilled. Some of these requirements, goals and constraints need to be sent to other intent management entities because they involve other management domains. An intent management entity can send those requirements, goals, and constraints to other intent management entities by means of intents.

In the detection phase the intent owner reacts to changes in its own internal goals or to changes in the fulfilment of the intents that were sent to intent handlers. The intent reports coming from the intent handlers serve as information sources for the detection phase.

Investigation:

In the investigation phase and intent owner determines what intents are feasible.

This has two aspects: first, the intent owner needs to find the right intent handlers that have the right domain responsibilities and support the intent information that the intent owner wants to define. Intent handler capability management and detection would be used for this process.

The other aspect of investigation would be to find out if the wanted intent is realistic. This means, if the intent handler would be able to successfully reach the wanted goals and meet the requirements. This depends on the current resource situation and state of the system and can vary over time. Typically, the feasibility of intent is done through a guided negotiation process between the intent handler and intent owner (using optional operations on the intent interface, as defined in clause 5.6). The owner can explore what the handling result of a wanted intent would be, what would be the best result the handler can achieve, or what would be the most challenging requirements, the aspiring intent handler can offer to fulfil.

Definition:

At the end of the investigation phase the intent owner knows what intent information objects are feasible and which intent handlers need to be involved. By combining this information with the needs that were identified in detection, the intent owner can now define all required intents.

In the definition phase the intent owner specifies and creates the required intents that need to be sent to one or more intent handlers.

Distribution:

In this phase the intent owner distributes the defined intents. It first identifies the right target intent handler for each individual intent. This involves an intent handler registry providing information about intent handling domain scope of available intent handlers. The registry information also contains the intent handler's capabilities with respect to the intent extensions and intent information models it implements and therefore supports to be used in the intent definition. Further details about the IME registration process can be found in clause 5.6.4.

For modified intent the used intent handler does not change, but its capabilities with respect to supported information elements need to be considered to avoid rejection of the intent.

If a suitable intent handler is discovered, the intent owner informs it about changes (using the intent interface). This includes setting of new intent, modification of existing intent or removal of intent that is not needed anymore. A targeted intent handler participates according to the intent interface operations and its reporting obligations.

Operation:

The intent handler operates on its own intent handling domain according to the requirements, goals and constraints set by the received and accepted intents. The operation phase involves the intent handler performing its own analysis, decision-making, and actuation trying to fulfil the intents and keeping them fulfilled.

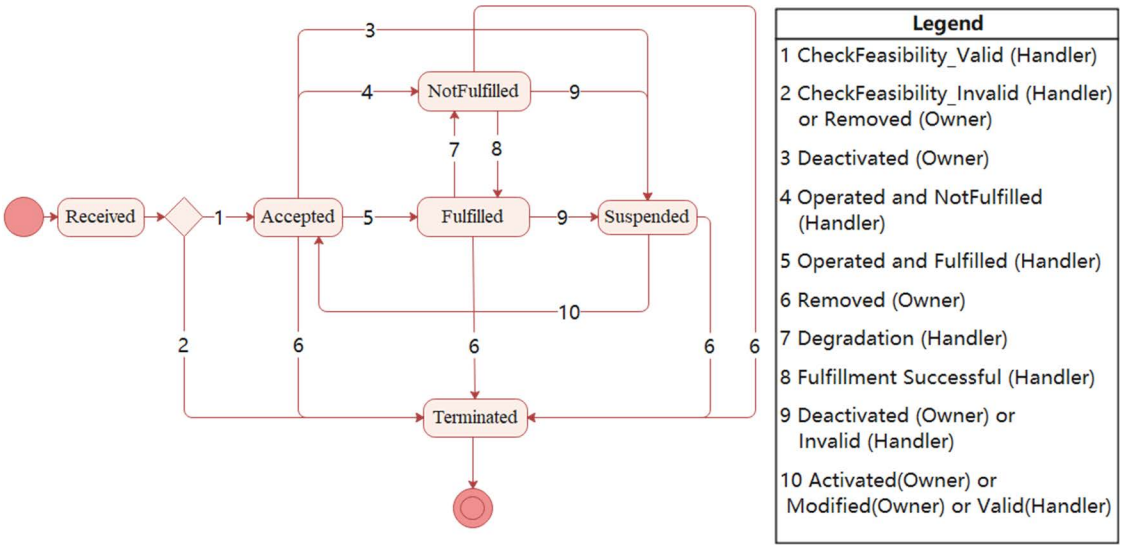
The intent handler is reporting on the state of handling and fulfilment success to the intent owner. This reporting is done individually for each distinct intent. The reporting closes the lifecycle loop as it enables the detection phase in the intent owner.

The intent handler cannot change the intent. It can only report on its success to meet the requirements to the intent owner. However, for multiple different intents a single intent management function can assume the role of intent owner for some and the role of intent handler for others.

The intent owner is responsible for detection and deletion of any loops that the chain of intents may form, including loops that may be created through interactions between several instantiations of the ZSM framework [i.13].

5.5.3 States machine of intent handling

The intent management entity may check the feasibility of each intent, to ensure the intent is in its scope, and also check potential conflicts between multiple intents. Generally, intents within an intent management entity have the following states.



NOTE: The state machine illustrated in Figure 5.5.3-1 is a high-level state machine, and the detailed state machine will be refined for normative work.

Figure 5.5.3-1: State machine of intent handling in intent management entity

Received: This is the initial handling state of an intent when it is received by an intent management entity. Next, the feasibility check may be performed, including validity/satisfaction/confliction, etc. The intent management entity accepts or rejects the intent-based on the check result.

Intent can be rejected for the reason of out of scope, unsupported models, unsupported context, etc. Intent can be abandoned by intent handler or removed by intent owner, which are implementation details.

Intent will be accepted if the check result is valid and compliant, or permitted to operate even if it does not meet the expectations completely.

Accepted: In this state, the intent management entity decides to accept the intent, if the intent should take effect immediately, the operation decisions and actions of the intent management entity try to transition the intent handling state into "Fulfilled" and keep it there until the intent is removed or suspended.

In some cases, the intent does not take effect immediately, thus it will be transited to the suspended state. For example:

- Not in the valid period.
- Not in the valid condition, but not a case of rejection (e.g. line is temporarily busy).
- Intent owner deactivated this intent (if supported).

Fulfilled: All expectations of the intent are met according to the current state of the system determined by measurements and analytics.

NotFulfilled: The intent is in this state if any of the expectation is not met.

The intent handler may try to optimize internally (e.g. when encountered with issues) to fulfil the intent. This may lead to intent handler to transition between intent Fulfilled and Notfulfilled state.

Suspended: In this state, intent should not be executed. It can be transited to Accepted state because of some reasons, for example:

- Intent owner activated this intent (if supported).

- Intent becomes valid again.
- Intent is being modified.

If intent handler judges the encountered issue cannot be solved, it will report intent owner, who will decide whether to remove the intent.

Terminated: The removal of the intent was requested, or some other exception cases occurred, and the intent management entity has stopped considering the intent in operation. The intent management entity is doing final clean-up actions.

NOTE: The intent management entity is however responsible for keeping the intent owner aware of changes of the intent handling state through intent reports, and intent owner can request to remove an intent in any state.

5.6 Intent-based interface

5.6.1 Introduction

The interface described in this clause is the interface between two distinct intent management entities, as introduced in clause 5.1. Through this interface the intent management entities manage the lifecycle of intent object instances.

The intent interface is established by introducing an intent handling management service that provides the required capabilities for the lifecycle management of intent object instances and for intent reporting. An instance of the intent management entity in the role of intent handler is the producer of this management service. Whereas the consumer of this management service is another instance of an intent management entity in the role of intent owner.

Clause 5.6.2 describes the relationship between intent handler and intent owners.

Clause 5.6.3 describes the operations on the intent interface that provide the needed capabilities for intent object instances life cycle management and for intent reporting.

5.6.2 Relationship between intent owner and intent handler

In an intent-driven management framework there are typically multiple sources of intents. Even from a single source, i.e. an intent management entity, the requirements may be expressed by several different intents. Therefore, often the intents received by an intent handler originate from multiple distinct intent owners.

Besides that, some intents, especially those expressing requirements for end-to-end services, may require the engagement of intent handlers with different scopes and responsibilities, i.e. involving different management domains. An intent owner may generate multiple intents to address the requirements for each affected domain and send these to different intent handlers. To summarize, there are in general many to many relationships between intent owners and intent handlers, as shown in Figure 5.6.2-1.

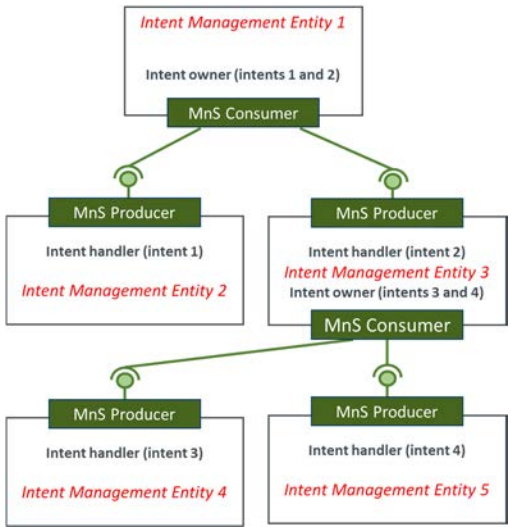


Figure 5.6.2-1: Relationship between intent owners and intent handlers

Despite the many-to-many relationship between intent owners and intent handlers, the intent object instances used between them are always unique. This means that there are never multiple intent owners for a given intent object instance instantiated at an intent handler, and an intent object instance is never instantiated at multiple intent handlers. If an intent owner needs to issue the same requirements, goals or constraints to different management domains involving different intent handlers, this has to be addressed by multiple distinct intent object instances with identical or similar content.

It is important to clarify that an intent management entity can play the roles of intent owner and intent handler for different intent object instances, as shown in Figure 5.6.2-1. For this, it is crucial that intent object instances are uniquely identifiable, and each intent object instance has a unique association with an intent owner (that originated the intent) and an intent handler (that fulfils the intent).

5.6.3 Operations on the intent interface

5.6.3.0 Introduction

There are several possibilities to provide the capabilities that are needed for the intent interface. The capabilities can be provided by operations on such an intent-based interface. The intent-based interface could be for example an:

- interface with detailed operations, e.g. "create", "activate/deactivate", "read", "update", and "delete";
- interface that aggregates some operations, e.g. an operation that covers both "create" and "activate" together;
- interface that is lightweight and offers only the set of minimally necessary operations and relies on the semantics of the intent information models to realize more detailed capabilities.

In general, the needed capabilities for intent object instances lifecycle management and intent reporting can be offered by the intent interface operations together with intent models that extend the intent interface capabilities and deliver additional information.
This could be the case, e.g. for reporting within a certain time frame triggered by an intent-based interface operation. The time frame information as a specific execution condition will be delivered based on the linked intent model and will not be specified in the respective interface operation itself.

TM Forum IG1253C [i.7] specifies a set of lightweight operations for an intent interface that is based on the mandatory operations "set", "remove", "report". Operation "set" is used to send the intent information object from intent owner to intent handler and can be used to modify the content of a given intent object instance. Operation "remove" is used to delete an intent object instance at the intent handler. And operation "report" is used to send the intent report from intent handler to the intent owner.

ETSI TS 128 312 [i.4] specifies operations "create", "modify", "delete" and "query" for the lifecycle management of intent object instances.

The present document investigates some options and then make recommendations for the subsequent normative phase. It builds upon the Create, Read, Update, and Delete (CRUD) operations pattern since this has been adopted for managing information objects in the ZSM architecture.

5.6.3.1 Mandatory operations

5.6.3.1.1 Introduction

The following interface operations can be used to enable intent object instances lifecycle management.
The operations below are listed as mandatory and should be supported by all intent management entities. Optional operations are listed in clause 5.6.3.2.

5.6.3.1.2 Create

This operation can be used for the creation of an intent object instance by an intent owner (MnS consumer at the intent interface) and to send this intent to an intent handler (MnS producer at the intent interface).

The create intent operation may result in the instantiation of the intent object instance, if the intent received by the intent handler is accepted, or it can result in the rejection of the new intent; in this case, the intent object instance is not instantiated at the intent handler.

5.6.3.1.3 Read

This operation can be used for reading the intent related information by an intent owner (MnS consumer at the intent interface).

This operation can also be used by any authorized entity.

The desired information could be requested with this operation. The desired information could be the intent content (i.e. its expectations), or an intent report with expectations fulfilment status related to that given intent object instance.

5.6.3.1.4 Update

This operation can be used for the modification of an intent object instance by an intent owner (MnS consumer at the intent interface). The only intent management entity that is allowed to use this operation is the intent owner (the one that created the intent).

5.6.3.1.5 Delete

This operation can be used for the removal of an intent object instance by an intent owner (MnS consumer at the intent interface). That means, after the successful execution of this operation the intent will no longer exist.
The only intent management entity that is allowed to use this operation is the intent owner (the one that created the intent).

NOTE: Authorized entities may also use the interface operations listed above.

EXAMPLE: The network owner may have policies which will result in any type of intervention, including executing the operations.

5.6.3.2 Optional operations

5.6.3.2.1 Introduction

An important aspect of intent-driven systems is their support of automation of intent owner-intent handler interactions. Such support is facilitated by the declarative and abstracted nature of intent information objects, in that an intent owner is not required to know anything about the intent handler's managed entity in order to formulate intents. This allows intents to be simply communicated from intent owner to intent handler, at least in cases where e.g. resource or other constraints on intent handler delivery of services are absent and valid intents may always be accepted by intent handlers. However, further intent interface operations may be created to support the negotiation - and the potential automation of negotiation - of intent-based service terms between intent owner and intent handler. For example, on receiving an intent request, an intent handler may not be able to fulfil it, or to fulfil it entirely, whereupon the intent handler might propose alternative service intents for the intent owner to choose from. Or, an intent owner may wish to discover what levels of service performance an intent handler may be able to deliver, before formulating specific service intents. The following optional interface operations help to create such operational richness in automatable owner-handler negotiations.

5.6.3.2.2 Judge

In some cases, the intent handler may find different solutions that are able to fulfil the intent expectations; one of the alternatives may be better or more optimized in one aspect and the other alternative may be better in another aspect.

In other cases, the intent handler may not be able to find any solution that is capable to fulfil all the intent expectations but may have some alternatives that are able to get better results for some of the expectations over others.

In both situations described above, while the intent handler has autonomy to decide the best approach to be taken, i.e. which solution to utilize, it may be beneficial that the intent handler and owner engage in a collaborative evaluation so that the intent owner can optimize the utility of the final intent.

The collaborative evaluation allows the intent handler to ask the intent owner to decide which out of many possible outcomes is preferred from the intent owner's perspective.

The Judge operation is initiated by the intent handler by asking the intent owner for the judgement on a set of solutions. In this procedure multiple intent reports are sent, each of them represents the expected outcome for a solution.

It is important to emphasize that the intent handler is not sending to the intent owner the details and the actions that are expected to be taken in each solution alternative, since the intent owner is not capable to understand these details. The intent handler would rather use intent reports to communicate the expected outcomes of these actions. This means the owner can judge based on effect of the actions rather than their nature.

The invocation of Judge operation (initiated by the intent handler) is followed by a response that indicates the order of the preferred solution alternatives. If a solution alternative is not desired by the intent owner (i.e. should be ignored by the intent handler), it is not included in the response. The intent owner can always decide not to consider any of the handler's alternatives. In case the intent owner cannot respond to a Judge operation for any reason, the intent handler is supposed to decide on its own what is the best alternative to be considered, following the principle that the intent handler has autonomy to decide the most appropriate actions to fulfil the intents.

5.6.3.2.3 Feasibility

In some cases, the intent owner may want to verify what expectations are actually feasible to be fulfilled by an intent handler or what outcomes can be provided by the intent handler.

This is an operation that should be used before the intent is actually considered by the intent handler as a set of requirements. This means that an intent handler is not expected to fulfil the given intent, but rather just provide the theoretic feasibility and outcomes (reports) in case the intent would be fulfilled.

5.6.3.2.4 Best

In some cases, the intent owner may be interested to know what is the best level of performance that the intent handler is able to deliver for a given set of requirements. In this situation, an intent owner creates an intent information object where one or multiple expectations are marked as for the evaluation for the best possible outcomes that could be achieved by the intent handler.

The response to a Best operation is an intent report where the expectations that were marked for evaluation indicate the best performance levels that currently can be fulfilled by the intent handler. It is assumed that any other constraints (e.g. cost budget, business relationship between owner and handler, etc.) specified as part of the intent being evaluated are taken into account in the evaluation process.

It is also possible to ask for the best possible proposal for already existing intent. This can be sensible if the intent handler fails to fulfil the intent as is and the intent owner wants to know what is the best level that the intent handler can deliver.

5.6.3.3 Optional operations to ensure trust in Intent-driven autonomy

5.6.3.3.1 Activate

The activation of an intent is a needed functionality.

This functionality can be used for example if:

- this intent was previously deactivated and should be activated again; or
- the intent was created but not yet activated for a processing by an intent handler, that means the creation of the intent does not include any activation of this intent to start the processing of it.

There are several variants to realize this activation functionality.

EXAMPLE: The following variants might be used to realize it:

- Realization as an interface operation.
- Realization via an information/data model.

An intent owner can apply the activation functionality to activate the achievement process of the intent at any time after a deactivation of this intent or when a creation of an intent is not combined with any activation of the processing of this intent by an intent handler.

In addition, there should be a possibility to define or connect additional information in the intent activation context, e.g. a start time for processing this intent by an intent handler, if needed.

5.6.3.3.2 Deactivate

The interface capability for the deactivation of an intent could be used optionally in addition to a basic operation set to support and to leverage the intent lifecycle management. This interface operation could be used when an intent/intent-based goal was previously activated by an intent owner and the running achievement process of the intent/intent-based goal should be deactivated based on any reason.

An intent owner can apply this operation to deactivate the achievement of the intent at any time after the execution of an activate operation/procedure related to this intent.

An alternative to the possible option mentioned above could be that the deactivation of an intent should be carried out through an appropriately configured attribute in a linked related model. This attribute is evaluated and based on it the corresponding deactivation steps are carried out, if requested.

5.6.3.3.3 Suspend

For a temporary suspension to achieve an intent which was previously defined and activated by an intent owner (who will be a Management Service (MnS) consumer at the intent interface). An intent owner can suspend the intent at any time.

The Suspend capability is similar to Deactivate with the difference that with a Resume interface operation the achievement of the intent could be continued, based on previous status as a valid continuation point. In contrast, the activation capability will start at the beginning of the achievement process of the intent.

5.6.3.3.4 Resume

After the reason(s) that caused the intent owner suspended the intent have been eliminated or mitigated, the achievement of the intent should be resumed.

The Resume interface capability could be applied when an intent was previously activated by an intent owner and the running achievement process of the intent is suspended for any reason.

NOTE: How to resume an intent fulfilment from a certain state of achievement of the intent goal at a point in time is for further study. In principle, this is an imperative statement that should be avoided in an intent-driven management.

In addition, that means a modification or deletion of the defined intent is not needed in these cases.

As an alternative, the suspension of an intent could be carried out through an appropriately configured attribute in a linked related mode, where this attribute is evaluated and based on the result of this the corresponding suspension of an intent should be carried out, if requested.

There are several variants to realize the Resume capability, e.g.:

- Realization as an interface operation.
- Realization with an information/data model.

An intent owner could apply this operation to resume the achievement of the intent after the execution of a related suspend operation at any time.

In addition, possibilities to define or connect additional information with the resuming of the achievement of the intent could be provided, e.g. a start time for resuming of the intent achievement by an intent handler, if needed.

5.6.3.3.5 Logging capabilities

There should be a logging service available with which for example the following activities can be logged on demand that are related to the intent-driven approach (not a complete list):

- Relevant activities/actions which are carried out during the processing of an intent (e.g. within ZSM domains).
- ZSM API/interface operations which have been applied.
- ZSM external communications e.g. with other applications, functionalities, and management functions/management services outside of ZSM.

The logging should be configurable from the respective ZSM intent owner and/or ZSM entities who are authorized for it.

Several different logging levels should be configurable, e.g. to provide different degrees of details to the authorized ZSM intent owner/ZSM entities. There should be a mechanism for the authorized ZSM intent owner as well as authorized ZSM entities to subscribe to one or more configured logging services which is used to express that the authorized ZSM intent owner and an authorized ZSM entity want such loggings provided for them.

With such logging capabilities used on demand should be ensured that relevant actions/events can be monitored and tracked during the processing and the fulfilment of an intent, if necessary.

In addition, with that it should also be possible to support the detection of security, data privacy, and legal issues.

5.6.3.3.6 Notification capabilities

There should be an event notification service to which ZSM consumers can subscribe including the several event types/notifications in the intent-based context. In addition, there should be a capability that the ZSM consumers will be able to make configurations related to such notifications.

With that, the monitoring and knowledge of what happening in the intent-based environment should be supported.

EXAMPLE: The translation of an intent into actions which several (micro-)services perform could lead to an abnormal behaviour. The related ZSM intent owner who has subscribed to a respective event type/notification will be informed and get information about this situation depending on the defined and configured notification.

Another possibility to provide such information to the ZSM consumer could be realized via the reporting as recommended by TM Forum in the intent-based context.

5.6.3.3.7 Testing

The interface capability for the testing of an intent could be used optionally in addition to a basic operation set to support and to leverage the intent lifecycle management.

The tests could be part of a negotiation phase or could be handled separately, generally before this phase or could be a mix of both.

In the following only the testing before a negotiation phase or during a negotiation phase will be considered. In the negotiation phase there can be carried out feasibility checks which could cover such testing possibilities or parts of them.

This functionality should support the intent owner to trigger tests for an intent automatically at an intent handler side. This might be realized based on a variety of information for example policies, rules, requirements and a goal belonging to this intent to find out and to check automatically e.g.:

- whether the intent could be fulfilled or not (pre-testing and/or feasibility check);
- several defined intents for the same goal and their results;
- different scenarios, e.g. what is the result of the energy consumption in connection with the fulfilment of the intent?

The intent owner should have the possibility to configure the tests functionality including the definition of one or several tests which should be realized.

There might be also predefined tests provided.

One or several configurable test reports should be possible to provide by the intent handler to the intent owner, if requested. At least any kind of confirmation concerning the test(s) should be provided by the intent handler to the intent owner.

The test reports should be made available on a streaming basis or a file basis in connection with configurable time intervals.

The following interface operation could be used as an optional operation in addition to a basic operation set:

- Test:
  - For testing an intent by an intent handler which was previously defined by an intent owner (the Management Service (MnS) consumer at the intent interface).
  - An intent owner can activate, modify, deactivate, suspend, resume, and delete the test(s) concerning his intent at any time.

5.6.3.3.8 Verification of intent outcome - optional interface capability

The purpose of this capability is to determine that an intent negotiation has resulted in a satisfactory handling of an intent, as well as note any deviations. The capability might be used for verifying the result of an intent on the part of the intent owner, or other authorized stakeholders, connected with a previously defined and executed intent after providing the associated result of this intent. This capability could be used optionally in addition to basic interface capabilities to support and to leverage the intent life cycle management in such a case where the intent owner wants to verify the result of the intent request after it has been executed by the intent handler.

There are several variants to realize such capability.

EXAMPLE: It could be realized also beyond such an interface.

In contrast to the testing capability described in clause 5.6.3.3.7, verification could be conducted after the intent negotiation has been concluded and the intent has finally been executed-in the target production environment, as well as the intent handler has sent the intent outcome back to the intent owner. The verification could be conducted by the intent owner or a dedicated procedure.

Deviations from the expected outcome should be made available to any concerned parties.

The following interface operation could be used as an optional operation in addition to a basic operation set to verify the result of the intent request after it has been executed by the intent handler:

- Verifying intent outcome:
  - For the verification of the result of an intent by an intent handler which was previously defined and requested by an intent owner (the Management Service (MnS) consumer at the intent interface). An intent owner can activate, modify, deactivate, suspend, resume and delete the verification task concerning the result of his intent request at any time.

5.6.4 Intent Management Entity registry

The IMEs that play the role of intent handlers in the management domains may need to be discovered by the IMEs that play the role of intent owners before the intents are exchanged with the operations defined in clause 5.6.3.

An intent owner may query multiple IMEs to decide which one should be the best intent handler for a given intent. The decision may be based on the supported intent models, on the supported intent interface operations, or any other decision criteria.

Every IME has a set of capabilities as well as a defined scope of management defined in the management domain where it resides. To make this information available, all IMEs may formally express its supported capabilities and scope of management by means of an IME profile.

An IME profile may have the information below:

- IME interface endpoint.
- Management domain.
- Supported intent roles.
- Supported intent optional operations.
- Supported intent models.

The IME profiles may be accessed by all authorized IMEs that are interested in such information. Such information may be managed by the different (E2E) management domain's data services, or by the cross-domain data services in ZSM architecture. The IME profiles may be kept in logical IME registry(ies) that are used for registration and discovery of IMEs.

The IMEs may decide to which IME registry(ies) it will register its profile and what information is contained in its profile. It is possible that IMEs only register its profiles within a given (set of) management domain(s) to limit its discoverability. It is also possible that only limited information is exposed in its registered profile, e.g. only IME interface endpoint and management domain. This allows flexibility for the IMEs to decide about the right level of capabilities exposure.

5.7 Handling management conflicts

5.7.1 Introduction

In most cases the Intent-driven management will co-exist with imperative operations-based management. The Intent-driven management will make service choices that are part of the total set of services that management domains will offer towards the E2E Service management domain, see Figure 5.7.1-1.

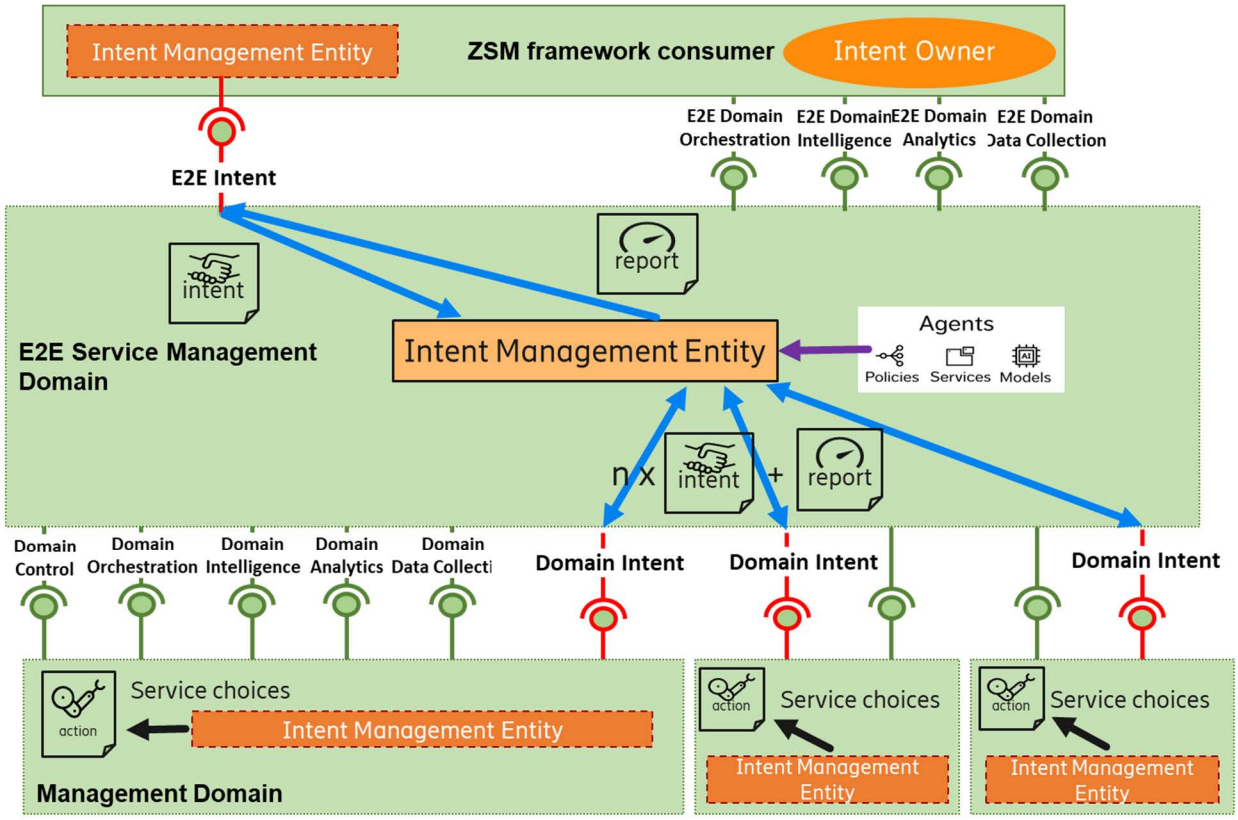


Figure 5.7.1-1: A hierarchy of Intent Management depicted in a simplified ZSM framework architecture diagram

As described in clause 4.1, an original intent will be transformed and decomposed when transferred between different domains. The original intent typically emanates from a ZSM framework consumer and will be transferred over the E2E intent interface to an Intent handler in the E2E Service Management Domain, analysed and transformed into intents that are transferred to one or several Management Domains over Domain Intent interfaces.

At each stage it can be decomposed into several new declarative intents and also, partly or completely, transformed into various actions.

The original intent does not need to be associated with an E2E service. It can also have another source than a consumer, e.g. a network function, and emanate from the E2E SMD or any of the MDs.

There are potential conflicts between different intents in the Intent-driven management, as well as, towards operations-based management.

There are potential benefits of using intent-based management in conflict detection and conflict resolution. Conflicts can be detected on a higher level, closer to the customer, by abstracting the goals to intents. The E2ES MD may assess inherent contradictions that are undetectable by the individual MDs. The abstraction opens a door to a dialogue with the consumer supported by reports provided upstream in the intent hierarchy.

One of the benefits of expressing intents as utility-level goals is that it helps the framework cope with the conflicting objectives of multiple intents. This is vital, because an autonomous framework often has to take multiple intents into account before making a decision.

For example, an autonomous framework may have one intent to deliver a service containing detailed data, while another may be to preserve data privacy. It can resolve such conflicts in a number of ways, e.g. explicitly from weights that introduce relative importance or implicitly from policies comparing different utility-level goals.

5.7.2 Categories of intent conflicts

According to the abstraction level, intent conflicts can be classified into the following categories:

Syntax-level conflict: A syntax-level conflict refers to a conflict between two or more components of an intent expression. It is characterized in that two or more components of an intent expressions describe mutually exclusive targets for the same object (e.g. the target of one intent component is to ensure the UL/DL RAN UE throughput < 5 Mbps, the target of another one intent component is to ensure the UL/DL RAN UE throughput > 8 Mbps for the same cell or base station).

Action-level conflict: An action-level conflict refers to a conflict between two or more intents on translated actions, that is, actions of the two intents cannot be executed at the same time. For example, for one intent to optimize RAN UE throughput, the action of the intent can be "turn off the power saving algorithm switch", for another intent to save energy, the action of the intent can be "turn on the power saving algorithm switch". These two actions of the two intents cannot be executed at the same time.

Impact-level conflict: An impact-level conflict refers to opposite effect between the translated actions of two or more intents, that is, actions of the intents can be executed at the same time, but the actions brings different impacts. For example, for one intent to optimize coverage performance, the action of the intent can be "adjust the antenna angle", for another intent to save energy, the action of the intent can be "turn on the power saving algorithm switch". The actions of the two intents can be executed at the same time, but the action "turn on the power saving algorithm switch" will deteriorate the coverage performance.

Note that the two or more intents referred to in the preceding paragraphs may be related to a single owner-targeted intent outcome, or to multiple such outcomes targeted by one or more owners, or a combination of both. Therefore, conflicts may arise within the context of a single closed loop (i.e. an I&M closed loop per clause 5.2), or - as described in ETSI GS ZSM 009-2 [i.15] - as conflicts arising from the concurrent actions of more than one such closed loop.

5.7.3 Intent - Non-Intent Conflicts

Intent-based Autonomous Networks, besides providing a way for the human operator to operate it via intents, may also be influenced by lower layer control operations such as policies or direct configurations targeting lower layer network functions. Intents may be translated and decomposed into sub-intents, policies, rules, and configurations that are then delegated to lower layer intent managers or network functions. The policies and direct configurations directly intervene with domain/resource controllers, network elements, and automation functions. There may be a potential conflict between an intent and a lower-layer policy. For e.g. the translated intent turns out to affect a set of configuration parameters which overlaps with the same set of configuration parameters that are targeted by the manually administered policies. It may be that there is explicit conflict of the fulfilment if the intent calls for increasing parameter A, while a manually administered policy targets to decrease parameter A. The Intent-based Autonomous Network needs to be capable of detecting and exposing (e.g. by notification, logs, traces) such conflicts towards to ZSM consumer, and resolving them inherently, e.g. using AI/ML assisted methods.

5.7.4 Potential intent conflict resolution approaches

In case of a potential conflict among multiple intents or between intents and policies, the intent-based autonomous network may support the capability to resolve the conflict inherently. A conflict may be resolved by multiple approaches depending on the preferences and requirements of the intent owner (MnS consumer). For example, intent handler (MnS producer) may set the priorities for execution among the intents under consideration after having a dialogue with the intent owner (MnS consumer). However, prioritizing certain intents over others may impact the network performance adversely, so an alternative approach could be to find a middle ground among the intents by averaging the requirements or actions produced by those intents. This approach does not favour a single intent or a group of intents, rather provides equal fairness to all. Another alternative is weighted averaging. In this case while doing the averaging, intent handler (MnS producer) may also open an additional dialogue with intent owner (MnS consumer) to put weights on certain intents which quantify their relative importance.

5.8 Intent translation

5.8.1 Intent translation: background

It is a core principle of intent-driven system interactions that intents are expressed solely in terms natively comprehensible to the intent owner. This is both desirable and necessary. Intent is declarative: the intent owner states what it wants in terms (generally) of service features or "outcomes" but does not participate to any degree in the determination of how the service is to be realized. The intent owner thus does not require any information about the intent handler's actualization resources, and, in fact, no such information is accessible to the intent owner. This has many positive consequences, including supporting portability of intents (i.e. the same intent could be offered to any handler), and maximizing handler degrees of freedom in service actualization, that are part and parcel of the utility of intent-driven networking.

However, this creates a requirement - nominally incumbent upon intent handlers (i.e. service-actuating IMEs) - to close any lexical or other comprehension gaps between owner and handler through intent translation.

In general, intents are submitted to the network as standardized information objects with a specific structure, e.g. as proposed in clause 5.4. However, intents may be provided by human (e.g. operators and users) since intent-driven management is striving to simplify the work of network management staff and human users, using intents. For human operators and users, however, it is easier to state intents in natural language. An example of intent in natural language is: *"I want to configure handovers for city of Munich for all IOT users"*. To achieve this translation, a pre-process can be used to deconstruct natural language intents submitted by humans and format these intents into the structure expected by the intent management entity according to its intent models and interfaces.

5.8.2 Intent translation: methods

It is helpful to think about intent translation in the context of conventional networking service "order entry" practices.

EXAMPLE 1: A telephone call: dialling a number identifies the service type and required parameters, as well as the service end-points. The source and target telephone numbers correspond to devices for both customer and operator; the operator has to resolve the locations (fixed or mobile) of both devices involved in the call in order to deliver the connection service between them. This involves some kind of lookup.

EXAMPLE 2: A WWW lookup: the Domain Name System (DNS) is accessed to provide translation between Uniform Resource Locators (URLs) relevant to users, and IP addresses relevant to delivery of service packets on the network.

EXAMPLE 3: A private line service: customers specify key parameters of the required service as part of an order entry process; generally, the present document will involve parameters comprehensible to both parties, such as terminal physical locations, bandwidth, etc. The process may involve feedback from the provider prior to order finalization and acceptance. Engineering of service delivery proceeds from this point, but the customer has no involvement in that.

These examples suggest means and methods that intent translation might leverage:

- An intent expression may use some terms that have universal meanings (e.g. bandwidth) and that do not require translation.
- An intent expression may use terms that are directly meaningful to both owner and provider, however each associate partly or fully disjoint different sets of information to such terms.
- An intent handler may incorporate or use a lookup or mapping service, to translate components of an intent expression.

A lookup or mapping service may require ongoing registration and correlation of "entries" from both information realms. This could involve direct communication between both owner and handler systems and the mapping service. It should be noted that effectively manual, administrative-type actions can be used to generate lookup/mapping entries: this is reasonable when entries are expected to be stable through the lifetime of a service or service class and customer relationship and may in fact represent the most practical solution in such cases.

Looking forward, it may be anticipated that the degree of information disjointness between owner and handler realms may increase over time or be larger in some use cases, with understanding of context providing a crucial backdrop to correct translation of terms. AI will be expected to play a critical role in intent translation in such cases through learning of intent term correlations between owner and handler realms, and in determining the context in which owner terms are used in a particular intent instance. Confirmation by owners of aspects of correct interpretation of terms by handlers may in some cases form a needed part of intent negotiations.

In the case of natural language translation, the translation may be based on rules or AI/ML as illustrated by Figure 5.8.2-1. The intent translation may read examples of intents and create a set of synthetic intents (i.e. variations of the input/example intent) using rules or AI/ML. The input intent to be translated can then be checked against the set of synthesized intents to identify the corresponding information objects according to the intent information model. Thereby the received natural language intent is mapped to the intent model.

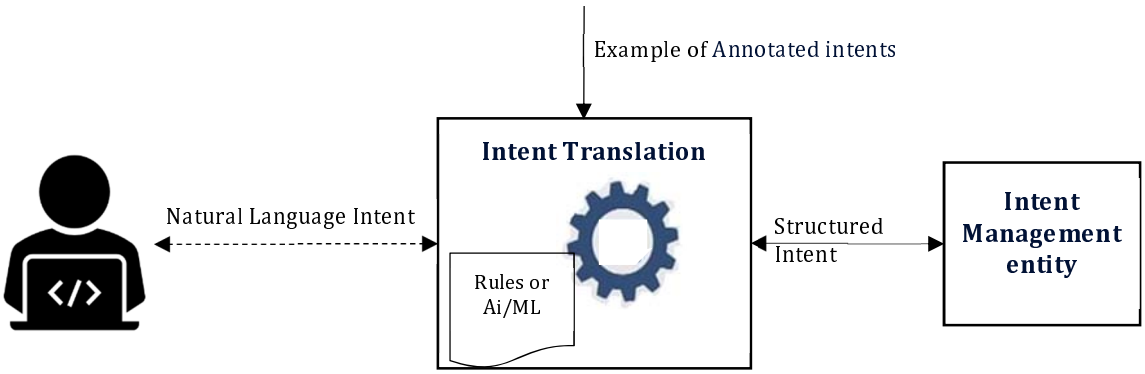


Figure 5.8.2-1: Intent Natural Language Translation

6 Next steps of standardization activities for ZSM Intent-driven autonomous networks

6.1 Summary of the present document

The potential use of intents as key enabler for enhancing autonomous network and service management within ZSM framework has been investigated in the present document, providing guidelines for generic intent-driven management interfaces, defining key operations, key aspects and concepts, principles of intents. The potential intent management conflicts and solutions to handle such problems were also listed.

6.2 Challenges faced on the present document

Given that the notion of intent as a concept and its role in the telecommunication industry has evolved over time, a significant challenge of the present document was to bridge the variety of different understandings of what intent-driven automation is.

Another challenge was to handle the coexistence between non-intent interfaces and intent-based interfaces in line with the ZSM architecture which also remains a challenge for future work.

6.3 Potential future work based on the present document

- Managing conflicts is essential to guarantee the correct behaviour of the system between entities of different domains, therefore it needs to be standardized. However, this subject calls for applying a wider view involving advanced control loops in combination with AI/ML, policies, rules, etc.

- As discussed in the present document, the process of identifying the right target intent handler for each individual intent object is essential for the whole intent-driven chain of interactions. The present document describes (on a high-level) an IME registration process that can be used for this purpose, see clause 5.6.4. More details about this registration process needs to be standardized.

- An intent handler may handle several intents in parallel and the different intents can be in different states depending on its current situation (e.g. Fulfilled, Suspended). Further details of these states need to be refined, since this was only addressed on a high level.

- When creating a new intent object, the intent owner needs to decide the intent extensions and intent information models, with which the intent is expressed, to provide the desired expression, depending on the domains involved in the intent-based interaction. The intent extensions and intent information models should be supported by both the intent owner and the intent handler so that the intent object can be correctly interpreted.

- Work on intent-driven management is ongoing in several SDOs and open-source projects and consequently several of information models are being created with a variety of prerequisites. Further work can be done to identify new models and how existing models can be combined to support end-to-end intent-driven services.

- The capabilities of the intent interface are key aspects for the intent management entities be able to manage the lifecycle of intent object instances and for intent reporting. The present document builds upon CRUD operations pattern since this has been adopted for managing information objects in the ZSM architecture. This way, the listed mandatory and optional operations and their "ways of working" needs to be standardized.

Annex A:
Examples of intents

Intents may be employed for different purposes, e.g. towards configuring the network or service, to manage the performance of the network or service, etc.

Table A-1 shows examples of intents expressed in a natural language format, specifying network and service configuration and performance management, targeting different managed entities.

Table A-1: Examples of intents expressed in natural language for network and service configuration and performance management

Intent type	Intent managed entity	Intent expression
Configuration Intent	Service	Activate service ABC
	Cell	Rehome cell x to RNC B
	Cell	Avoid Energy Saving for cells in CBD
	Cell	Restrict Energy Saving to Rural cells on week days
	Network slice	Create a network slice of type IIoT
	Transport path	Add transport capacity to the path xzy
Performance intent	Cell	Ensure for cell xyz, Busy hour load < value_V
	Sub network	Ensure for subnetwork xyz, coverage/RSRP > Some_value in CBD
	Cell	Ensure that congestion to cell xzy is < 50 %
	Transport path	Ensure that congestion on transport path xzy < 50 %
	Service	Ensure that congestion on service Serv_xzy < 50 %
	Network slice	Ensure that the end-to-end latency of user in network slice N2 < 10 ms

Figure A-1 shows an example of an intent information object that is based on RDF [i.11]. It uses the model federation described in clause 5.4 and detailed in TM Forum's IG1253B [i.7]. The language used for serialization of RDF object is TURTLE. A concise tutorial on the notations and constructs of TURTLE language that are used for this type of intent expression can be found at [i.5].

NOTE: Figure A-1 shows one example of a hypothetical intent information object with multiple expectations. The URIs presented in the figure are also hypothetical, since no such models are yet already publicly available. The example is illustrative and has not been checked for syntax conformance.

```
1. @prefix icm:  https://tmforum.org/2020/07/intent/
2. @prefix tel:  http://sdo1.org/TelecomConcepts/
3. @prefix met:  http://sdo1.org/metrics/version2/
4. @prefix sli:  http://sdo2.org/2021/03/SliceIntent/
5. @prefix slk:  http://sdo2.org/2019/SliceKPI/
6. @prefix tim:  http://sdo4.org/time/
7. @prefix cat:  http://operator.com/Catalog/
8. @prefix ope:  http://operator.com/Inventory/

9. ope:ExampleIntent2021031100002
10.   a icm:Intent ;
11.   icm:hasExpectation
12.   [ a icm:DeliveryExpectation ;
13.     icm:target _:function ;
14.     icm:params [ cat:amf ]
15.   ] ,
16.   [ a icm:DeliveryExpectation ;
17.     icm:target _:slice ;
18.     icm:params [ cat:SliceTypeA ]
19.   ] ,
20.   [ a icm:PropertyExpectation ;
21.     icm:target _:function ;
22.     icm:params [ tel:subscribers
23.                     [ icm:atLeast [ met:value 1000 ] ;
24.                   met:availablility
25.                     [ icm:atLeast [ met:value 99.9 ]
26.                ]
27.   ] ,
27.   [ a icm:PropertyExpectation ;
28.     icm:target _:slice ;
29.     icm:params [ slk:latency
30.                  [ icm:atMost [ met:value 10 ;
31.                                 met:unit met:unitMs ]
32.                ]
33.   ] ,
34.   [ a sli:LinkExpectation ;
35.     imm:target _:slice ;
36.     imm:params [ sli:connectingEndpointOf _:function ]
37.   ] ,
38.   [ a imm:ReportingExpectation ;
39.     imm:target ope:ExampleIntent2021031100002;
40.     imm:params [ imm:regularReporting tim:hourly ;
41.                  imm:eventReporting imm:intentViolation ]
42.   ] .
```

Figure A-1: Example of intent information object based on RDF using the model federation concept

Lines 1 to 8 in Figure A-1 point at all used models and assigns separate namespaces, which is the basis for combining multiple models in the model federation approach.

Line 9 starts the intent information object. Each element of the RDF triple starts with a globally unique identifier, e.g. 'ope', 'icm', 'met', 'sli', 'slk', etc. The intent object instance can be uniquely identified by the namespace 'ope' that points out to the operator inventory (see line 8).

The color code in Figure A-1 is just used to illustrate the model federation been applied.

Everything in yellow represents models created by the operator. For instance, line 18 points out to a SliceTypeA definition that is modelled in operator's catalog (see line 7).

Everything in purple represents models crated (and governed) by SDO1. For instance, lines 22 and 24 point out to models that represent the concept of 'subscribers' and 'availability'. Such concepts are provided by SDO1 (see lines 2 and 3) since they may be specific to the scope of this organization and out-of-scope of others.

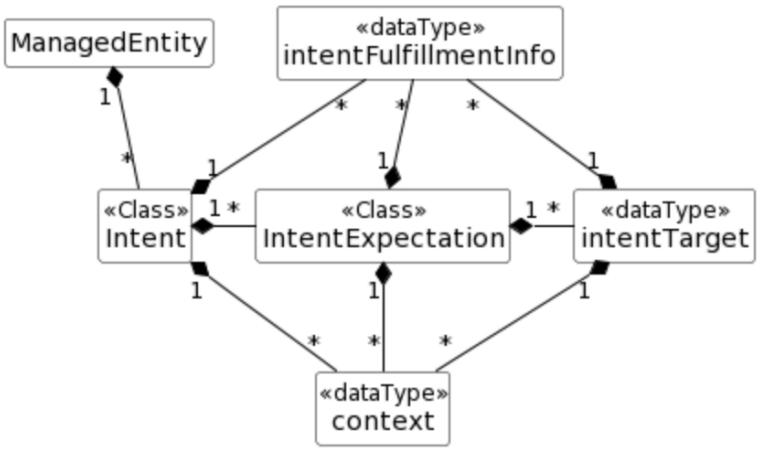
Likely, in line 29 there is a reference to a model provided by another organization, in this case SDO2.

Finally, the TM Forum's metamodel can be extended, as described in clause 5.4, allowing domain-specific expectations and other definitions. In line 34, an extension is created by SDO2 that allows the definition of a link expectation, which can specify requirements and goals related to connections between end points. Such definition is out-of-scope of TM Forum's metamodel, but is in-scope of any organization that deals with, e.g. transport management domain.

Annex B:
Required Classes of the declarative intent model

B.1 Example of declarative intent model

Figure B-1 gives an example of the declarative intent model.



NOTE 1: <<Class>> Is an object that can be instantiated, i.e. the instance can be created to leave as an independent object.
NOTE 2: <<DataType>> is a literal, i.e. it cannot be instantiated but can be assigned.

Figure B-1: Relationship UML diagram for intent

B.2 Intent <<Class>>

This Class represents the properties of an `Intent`.

The `Intent` includes the following attributes:

Attribute Name	Description
`IntentExpectation`	It indicates the expectations including requirements, goals and constraints on a given Managed Entity of the intent
`intentIdentifier`	It is the A user-friendly (and user assignable) name or identifier of the intent. It may be used to refer to the intent e.g. when deleting the intent The `intentIdentifier` is a string
`intentContext`	It describes the list of constraints and conditions that should apply for the entire intent even if there may be specific constraints and conditions defined for specific parts of the intent The `IntentContext` is of type context
`intentFulfillmentInfo`	It describes status of fulfilment of the intent and the related reasons for that status The `IntentFulfillmentInfo` is of type `fulfillmentInfo`

B.3 IntentExpectation <<Class>>

This Class represents the properties of an `IntentExpectation`.

The `IntentExpectation` includes the following attributes:

Attribute Name	Description
`expectationIdentifier`	A user-friendly (and user assignable) name of the intentExpectation.
`managedEntityType`	It describes the type of managed entity to which the given intentExpectation should apply. It is used together with the ManagedEntityContext to identify the specific entity to which the intentExpectation should apply. E.g. the intentExpectation may be stated for a network slice (type of Object) with identifier IIOT_Atomotive_2021 (identifier as context). Alternatively, the intentExpectation may be stated for a network slice (type of Entity) serving IIoT users (context 1) with network slice profile supporting automotive connectivity (context 2).
`managedEntityContext`	It describes the list of constraints and conditions to be used as filter information to identify the specific intentObject to which a given intentExpectation should apply. Note there may be other constraints and conditions defined either for the entire intent, for the specific intentExpectation or for the intentTarget of the considered intentExpectation. E.g. the intentExpectation may be stated for a network slice (type of entity) with identifier IIOT_Atomotive_2021 (identifier as context). Alternatively, the intentExpectation may be stated for a network slice (type of entity) serving IIoT users (context 1) with network slice profile supporting automotive connectivity (context 2). The `ManagedEntityContext` is of type context.
`intentTargets`	It describes the list of specific outcomes on configurations and observables related to the stated intentObject (e.g. parameters, gauges, counters, KPIs, etc.) that are desired to be realized for a given intentExpectation.
`expectationContext`	It describes the list of specific outcomes on configurations and observables related to the stated intentObject (e.g. parameters, gauges, counters, KPIs, etc.) that are desired to be realized for a given intentExpectation. The `ExpectationContext` is of type context.
`expectationFulfillmentInfo`	It describes status of fulfilment of the intentExpectation and the related reasons for that status. The `expectationFulfillmentInfo` is of type `fulfillmentInfo.`

B.4 IntentTarget << dataType >>

This dataType represents the properties of an `IntentTarget`.

The `IntentTarget` includes the following attributes:

Attribute Name	Description
`entitySateAttribute`	It describes a specific attribute of a managed entity on which an outcome may be stated, either a configuration or observable of that managed entity. The attributes may be a parameter, gauge, counter, KPI, weighted metric, etc. related to that managed entity.
`targetCondition`	It expresses the limits within which the entityStateAttribute is allowed/supposed to be. Example values: is equal to; is less than; is greater than. See note.
`targetValueRange`	It describes the range of values that applicable to the entityStateAttribute as constrained by the targetCondition.
`targetContext`	It describes the list of constraints, conditions and filter information that should apply for a specific intentTarget. Note there may be other constraints, conditions and filter information defined for the entire intent or the intentExpectation. The `targetContext` is of type context.
`targetFulfillmentInfo`	It describes status of fulfilment of the intentTarget and the related reasons for that status. The `targetFulfillmentInfo` is of type `fulfillmentInfo`.
NOTE: Others conditions like "is within the range" or "is outside the range" can be expressed in terms of these basic conditions.	

B.5 context << datatype >>

This dataType represents the properties of a `context`. A `context` describes the list of constraints and conditions that should evaluate to True when the targets are fulfilled but are themselves not to be enforced. The context may apply to the intent, the intent expectation, the intent targets or to the managed object as filter information used to identify the manged objects to which the targets are intended.

The `context` includes the following attributes:

Attribute Name	Description
`contextType`	Defines the roles for which a given context play. allowedValues: {"ManagedEntityContext", "ExpectationContext", "TargetContext", "IntentContext"}
`contextAttribute`	It describes a specific attribute of or related to a managed object or to characteristics thereof (e.g. its control parameter, gauge, counter, KPI, weighted metric, etc.) or an attribute related to the operating conditions of the managed object (such as weather conditions, load conditions, etc.).
`contextCondition`	It expresses the limits within which the ContextAttribute is allowed/supposed to be. The `ContextCondition` may be one of the following values: "is equal to"; "is less than"; "is greater than". See note.
`contextValueRange`	It describes the range of values that explicatable to the ContextAttribute as constrained by the ContextCondition.
NOTE: Others conditions like "is within the range" or "is outside the range" can be expressed in terms of these basic conditions.	

B.6 fulfillmentInfo << dataType >>

This dataType represents the properties of a specific fulfilment information.

The `fulfillmentInfo` includes the following attributes:

Attribute Name	Description
`fulfillmentStatus`	It describes the current status of fulfilment for the intent, intentexpectation or intentTarget. It is configured/written by the MnS producer and can be read by the MnS consumer. The `fulfillmentStatus` may be one of the following values: "ONGOING", "SUSPENDED", "FAILED", "FULFILLED".
`fulfillmentStatusReasons`	It describes the reason(s) for the current status of fulfilment for the intent, intentexpectation or intentTarget. It is configured/written by the MnS producer and can be read by the MnS consumer. The `fulfillmentStatusReason` is of type string.

Annex C:
Testing intent-based autonomous networks and services

In an intent-based autonomous networks, the MnS consumer (intent owner) specifies one or more intents, and the execution of those intents are managed by MnS producer (intent handler). Although using intent simplifies the operator's work, it takes away the network management control from the operator. With the current trend of using more and more automation, an operator does not know the behaviour of an MnS producer beforehand. So, before a human operator can give such control away, the operator needs to be sure that the MnS producer (intent handler) executes the intents in a way that matches with the operator's expectations. This may be achieved by testing the intent-based autonomous networks e.g. using sandbox, digital twin or part of live network.

The testing of the intent-based autonomous networks (see annex E) may support the below generic capabilities:

- It provides sample intent to the intent driven MnS producer (intent handler) which takes necessary actions to fulfil the intent (either on live or simulated environment).
- It observes and reports the changes (or lack of changes) in the network after the intent is executed by the MnS producer (intent handler) as well as any variations from the operator's expectations.

The testing may be done prior to deployment or for periodic validation of intent-based MnS. The testing may support different aspects of intent fulfilment such as intent translation, validation, and execution.

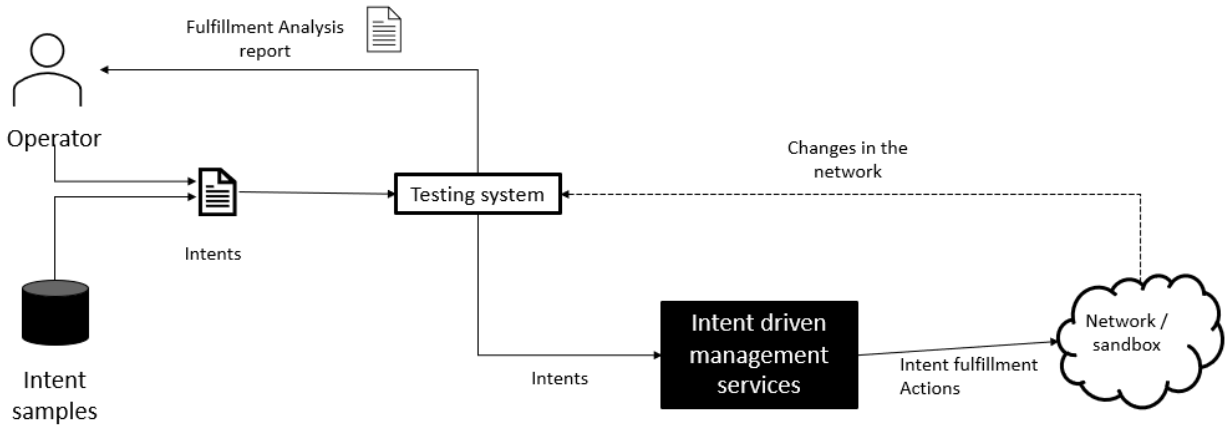


Figure C-1: Testing intent-based autonomous networks

Annex D:
Alternative Concepts of Intent modelling

D.1 List of challenges

The owner of an intent expects finally a service which fulfils his intent. The task of the intent handler is the interpretation and translation of the intent into a concrete managed service (or a set of services, or just a reconfiguration of services) of the managed network, and after creation to keep the service available and matching the given expectations (e.g. quality or quantity) by means of an autonomous closed loop (the I&M closed loop discussed in clause 5.2).

The present document puts in clause 5.4 a focus on a specific intent model format for exchange between owner and handler, it contains information that intent handler is sufficiently informed about existing context to interpret and translate the intent into concrete services. This annex discusses alternatives for intent modelling.

Effort and cost to develop the automation of a new service (or updates for existing services), are a matter of the intent handler implementation. Those may vary and are not addressed in this present document. However, the specific challenges of 5G and cloud native networks are dynamics and complexity, which triggers the discussion to simplify the intent handler's implementation.

Clause 4.2.4 identifies different business goals for which intent can support automation. The most important goals are a) autonomy (reducing cost of operations) by means of service abstraction, and e) time-to-market (reducing time and cost of developing automation for services) by capabilities of the individual intent handler. They do not necessarily need a new format of a service request, other than a higher abstraction (for example "connectivity" is more abstract than "eMBB network slice"). Other features, such as support for human language, negotiable offerings, or find the best producer will require additional information beyond just a service abstraction.

Depending on the business goals, the approach to model intent may vary. This appendix raises awareness of alternative approaches to model intents.

The present document describes the intent in clause 4.2 as a "formal specification of the expectations, including requirements, goals, and constraints". The intent owner and handler need to agree about the interpretation of an intent and there should be no ambiguity. That raises various challenges:

1) the intent owner needs to find an intent handler which may fulfil the desired intent;

2) the intent handler needs to understand the subject of the intent (e.g. what means "connectivity");

3) the intent expectations have to be interpreted and potentially negotiated with the owner (e.g. 10 GB bandwidth is the best possible); and finally

4) the intent is to be fulfilled with concrete services matching given expectations (e.g. quality or quantity).

Issues 1) to 3) require a new meta-model for intent (the intent information object class), which goes beyond the content of a classic service definition (that is a CFS or RFS specification).

Issue 4) depends on the capabilities of the intent handler to translate the intent into concrete services. Given the goal is just a) autonomy (see clause 4.2.3), a more classic CFS specification format can be used to express the intent, given it has an abstraction level and parameters to express expectations to allow the handler's autonomy in terms of how to instantiate that service in the network.

Three alternative concepts for intent modelling are identified in Figure D.1-1.

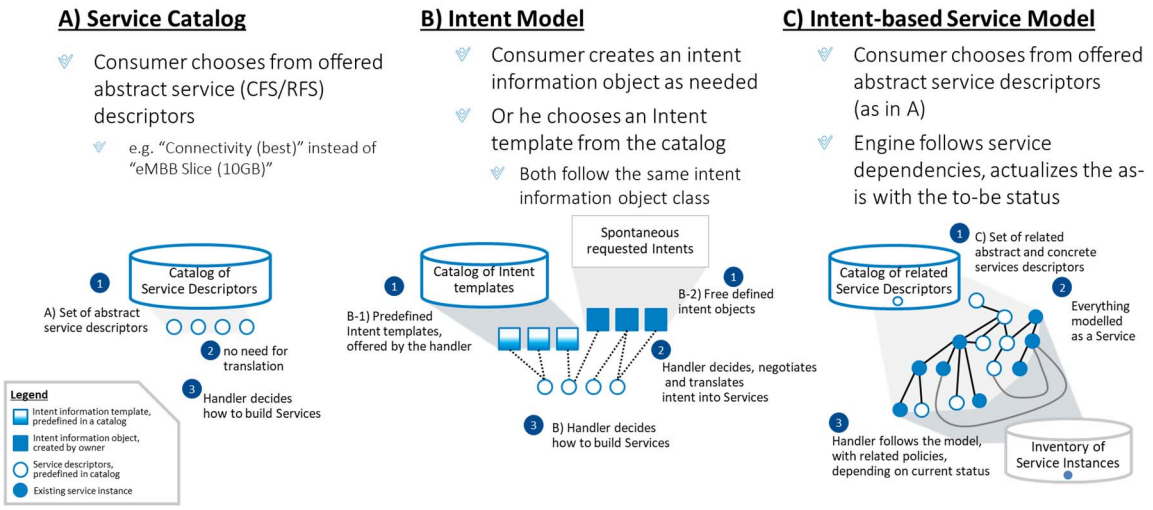


Figure D.1-1: Alternative concepts to model "intent-based"

All concepts foresee three steps to process:

1) Create the intent, either from a catalog or from scratch

2) Understand and negotiate the intent to translate it into a service (or multiple services and resources)

3) Fulfil the intent by deciding how to instantiate and operate the service autonomously (using a closed loop)

D.2 Service catalog

This is a well-known method by which a service catalog indicates offered services (CFS and RFS) by a producing domain to prospective consumers. This is compatible with the achievement of autonomy goals in an intent-driven system as long as the service description is appropriately abstracted and does not require the owner to comprehend non-native terms.

As example an abstract CFS may be called "connectivity" instead of a concrete "eMBB network slice", and specify qualifiers such as "bandwidth = best available" instead of concrete "bandwidth = 10 GB". The handler needs to find out at runtime, what concrete service with what concrete quality can provide the expected connectivity.

NOTE: Abstraction is always relative at the level of processing (e.g. E2E, Domain, Network Controller), and any service is finally an abstraction of a configuration in the network. Consumer and handler need to agree on a common understanding of the service expected by the intent.

In this approach, the consumer chooses from existing (even abstract) service offerings offered with the catalogue. That simplifies the effort of translation and interpretations and avoids wrong expectations. Still the offering is abstracted, so the owner does not need to know any specifics of the service, and the handler keeps all autonomy, to decide how to instantiate the service and assure its quality in a closed loop.

The approach still fulfils all the principles explained in clause 4.2.2, while it supports achievement of the business goal a) for autonomy as listed in clause 4.2.4.

D.3 Intent model

The second approach uses a dedicated generic meta-model for describing an intent (such as discussed in clause 5.4). It allows more flexible expectations, or nontechnical terminology with human language and enables negotiations or brokering. It supports freedom and potentially simplification of intent creation by a consumer. It is the main focus in the present document.

Without a catalog (option "B-2" in Figure D.1-1) the handler needs to understand and translate the intent, and potentially negotiate concrete and fulfillable expectations with the owner. In the last step, the handler decides what services are to be used to fulfil the intent, that means how to instantiate them and keep them matching the given expectations (e.g. quality or quantity).

The need for translation or negotiation can be simplified with a catalog of predefined intent information object templates (option "B-1" in Figure D.1-1), defined by the handler. The consumer may select one ("a la card") of the intent templates which is likely fulfillable and the handler already expects. Of course the consumer may augment or modify the predefined expectations from the template, then the handler still has to translate or negotiate the intent request.

D.4 Intent-based service model

The third approach is described in [i.5], clause 4.4 as intent-based service orchestration. It simplifies the implementation of an intent handler, hence is addresses the business goal e) for time-to-market (see clause 4.2.4).

The service catalog provides more than single abstract service descriptions: it provides a model how to compose the abstract service from other abstract or concrete services. Each service descriptor may have policies reacting on given context such as as-is status of already instantiated or configured services. The handler generates the runbook to instantiate (or update) the desired service at runtime; it avoids the need to code workflows for all possible instantiation variants, which the service model would allow.

The approach addresses Dave Lenrow's requirement "Intelligent software determines how to translate the intent into an infrastructure-specific prescription that causes the network to behave in the desired manner" (see [i.14]). The handler's behaviour is completely defined with the service model, enabling an "intent-based policy evaluation" (described in clause 4.3 of [i.1]). The precondition is the ability to express necessary dependencies and policies with the service descriptors, such as "Dynamic Service Descriptors" (DSD) (introduced in clause 4.4 of ETSI GR ZSM 005 [i.1]).

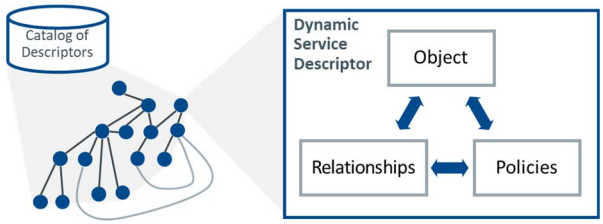


Figure D.4-1: Dynamic service descriptor concept

A single DSD comprises object characteristics (i.e. service attributes), policies and relationship. It allows a *model* of related service descriptors - every single descriptor can be seen as an atomic intent, using or used by other atomic intents. That service model represents the knowledge base that contains the ontology of multiple intents.

The approach serves the business goal of autonomy and time-to-market, while it neglects advantages of flexible negotiations, as with a dedicated intent model in case B). Like for A), the consumer can choose an abstract service from the catalog.

Avoiding the use of a catalog and serving all business goals listed in clause 4.2.4. can be achieved by a combination of B) and C) where the intent is translated into an abstract service first. Policies of the single services may take further information of the intent into account, to decide about the concrete service instantiation.

Beside the improved time to market, the approach provides further advantages for the consumer:

- Using a catalog, the owner has full clarity about the possible intents, while he is limited to those.
- The owner may skip knowledge about existing context, as the handler is aware about it (example: to request a connectivity branch, the handler already knows the relying private network).

Annex E: Bibliography

- W3C® Recommendation 25 February 2014: "RDF Schema 1.1".

NOTE: Available at https://www.w3.org/TR/rdf-schema/.

Annex F:
Change History

Date	Version	Information about changes
February 2021	0.0.1	Skeleton approved during ZSM-14b tech call
June 2021	0.0.2	Incorporated contributions: - ZSM(21)000186r2_ZSM011_-_introductory_content - ZSM(21)000192r1_ZSM011_-_Introducing_the_term__Utility__in_intent-driven_man
July 2021	0.0.3	Incorporated contributions: - ZSM(21)000238r1_ZSM011_-_Definition_of_intents
August 2021	0.0.4	Incorporated contributions: - ZSM(21)000245_ZSM011_section_4_1_introduction__addition_proposed
September 2021	0.0.5	Incorporated contributions: - ZSM(21)000270r2_ZSM011_Mapping_of_ZSM_control_loop_concepts_to_Intent_Handli - ZSM(21)000277r1_ZSM011_-_TeraFlow_Use_Case
September 2021	0.0.6	Incorporated contributions: - ZSM(21)000246r2_ZSM011_section_5_5_Intent-based_interfaces__capabilities_and - ZSM(21)000279r3_ZSM011_-_Clause_5_3_Intent_meta-model - ZSM(21)000282r1_ZSM011_Section_5_5_1_Introduction - ZSM(21)000283r1_ZSM011_-_Additional_informative_reference_ - ZSM(21)000304_ZSM011_-_Fixing_references
October 2021	0.0.7	Incorporated contributions: - ZSM(21)000258r1_ZSM011_section_5_5_Intent-based_interfaces__capabilities_and
November 2021	0.0.8	Incorporated contributions: - ZSM(21)000188r8_ZSM0011_-_Intent-driven_interactions - ZSM(21)000257r2_ZSM011_section_5_5_Intent-based_interfaces__capabilities_and - ZSM(21)000338r1_ZSM_011_-_Cloud_Private_Line_Use_Case - ZSM(21)000344r1_ZSM011_-_Solving_ENs_in_clauses_4_1_and_4_2
December 2021	0.0.9	Incorporated contributions: - ZSM(21)000318r2_ZSM011_Definition_of__Intent_IOC_IIO_MIO - ZSM(21)000346r1_ZSM011_-_Improving_Clause_5_5_-_Intent-based_interfaces - ZSM(21)000347r3_ZSM011_-_Intent_life_cycle - ZSM(21)000353r1_Adding_text_to_clause_5_6_Handling_conflicts_between_intents - ZSM(21)000368_ZSM011_Align_definition_of_intent_in_clause_4_2 - ZSM(21)000369r2_ZSM011_Add_categorizes_of_intent_conflicts_to_clause_5_6 - ZSM(21)000370r1_ZSM_011_Add_intent_interactions_between_cross-layer_domains - ZSM(21)000371_ZSM011_Clarify_description_of_intent_in_clause_5_3_1 - ZSM(21)000375r1_ZSM011_Declarative_Intent_Model - ZSM(21)000377_ZSM011___Nomenclature_update_to_section_5_2_3__IHE_to_IME - ZSM(21)000398r2_ZSM_011_-_Intent_Translation - ZSM(21)000419r2_ZSM011_Section_5_3_1_Introduction
December 2021	0.1.0	Incorporated contributions: - ZSM(21)000433_ZSM011_Section_5_3_4_Criteria_for_selection_of_Intent_Meta_M

Date	Version	Information about changes
January 2022	0.1.1	Incorporated contributions: - ZSM(22)000015_ZSM011_-_Fixes_clauses_1__2_and_3 - ZSM(22)000016r1_ZSM011_-_Solving_EN_clause_4_1 - ZSM(22)000017r1_ZSM011_-_Introduction_to_the_clauses_of_use_cases - ZSM(22)000018r1_ZSM011_-_Fixing_EN_clause_4_3_1_-_Automotive_use_case - ZSM(22)000019_ZSM011_-_Fixing_EN_clause_4_3_2_-_cloud_private_line_use_cas - ZSM(22)000020r1_ZSM011_-_Fixing_EN_clause_5_1_-_ - ZSM(22)000021r1_ZSM011_-_New_structure_to_clause_5_2 - ZSM(22)000022r1_ZSM011_-_Clause_5_3_1_Introduction_to_intent_model_federatio - ZSM(22)000023r1_ZSM011_-_Deleting_shalls_from_clause_5_3_2 - ZSM(22)000024_ZSM011_-_Removing_clause_5_4_-_MD-specific_information_model - ZSM(22)000025_ZSM011_-_Fixing_ENs_clause_5_6_1_and_5_6_3 - ZSM(22)000026_ZSM011_-_Removing_Annex_A - ZSM(22)000040_ZSM011_Modifications_related_to_section_5_5_Intent_life_cycl - ZSM(22)000042_ZSM011_Section_5_3_4_6_and_5_3_4_7_Intent_Tracking
February 2022	0.1.2	Incorporated contributions: - ZSM(22)000067r1_ZSM011_Annex_A2_and_Restructure_Section_5_4_4
March 2022	0.1.3	Incorporated contributions: - ZSM(21)000367r3_ZSM011_section_5_5_x_Testing_-_optional_interface_capability
March 2022	0.1.4	Incorporated contributions: - ZSM(22)000085r1_ZSM011_Restructuring_Section_5_4_4
May 2022	0.1.5	Incorporated contributions: - ZSM(22)000076r2 ZSM011 Adding optional operation - Judge intent - ZSM(22)000077r2 ZSM011 - Adding optional operation - Probe Intent - ZSM(22)000078r3 ZSM011 - Adding optional operation - Best intent - ZSM(22)000131r1 ZSM011 Sec 5.7.2 5.7.3 intent conflict management
May 2022	0.1.6	Fixing bug when uploading the 0.1.5 to the ETSI portal. No new content added
May 2022	0.1.7	Incorporated contributions: - ZSM(22)000088r6-ZSM011 Section 5.5.2 adding intent state machine - ZSM(22)000103r3-ZSM011 - Changes to clause 5.2 - ZSM(22)000104r2-ZSM011 - Changes to clause 5.3 - ZSM(22)000109r2-ZSM011 Sec 5.1 Updates Intro to ZSM FW - ZSM(22)000189-Combining ZSM(21)000253r4 and ZSM(21)000254r2 in ZSM011 draft - ZSM(22)000225 Change to ZSM(22)000189 - ZSM(22)000197r1-ZSM011_Sec_5_7_1_Upates_intent_conflict - ZSM(22)000198r2-Updates intent model in annex A2 - ZSM(21)000241r3 ZSM011 section 6.x Potential capabilities for intent-driven logging - ZSM(22)000106r3_ZSM011_Sec_4_2_Updates_Principles_of_Intent
June 2022	0.1.8	Incorporated contributions: - ZSM(22)000180r3 ZSM011_Mapping_on_Intent_and_DeclarativeCLGoalStatement - ZSM(22)000187r3 ZSM011 Sec 5.8 update for Natural Language translation
June 2022	0.1.9	Incorporated contributions: - ZSM(22)000083r4 ZSM011 - Changes to clause 5.6.3.1 Mandatory operations

Date	Version	Information about changes
June 2022	0.2.0	Inserted placeholders, as editor's notes, for existing contributions, which are recognized as potentially being part of the scope of the GR. In addition to these, only contributions that CLEARLY resolve a gap or something wrong will be considered. • ZSM(22)000098r1 ZSM011 5.6.z Verification of intent outcome - optional interface capability - Deutsche Telekom AG • ZSM(22)000110 -"ZSM011 Sec 5.3 Intro to Intent model"-(KRICHEL Andreas, HPE) • ZSM(22)000107-"ZSM011 Sec 4.3 Updates Use Cases Intro"-(KRICHEL Andreas, HPE) • ZSM(22)000177-"ZSM011 Sec 5.1.2 Options to implement the IME"-(KRICHEL Andreas, HPE) • ZSM(22)000178-"ZSM011 Sec 5.2 Updates for Control Loop"-(KRICHEL Andreas, HPE) • ZSM(22)000202r1 ZSM 011 Addition to section 5.2"-(JANZ Christopher, Huawei) • ZSM(22)000223 ZSM011 Sec 3.1 Terms update - Hewlett-Packard Enterprise • ZSM(22)000192r2 ZSM011 Sec 5.9 Intent-system Testing Framework Nokia Germany • ZSM(22)000200r3 ZSM011 Sec 4.x Purpose of Intent - Hewlett-Packard Enterprise • ZSM(22)000224r1 ZSM011 Appendix: Alternative Intent based concepts Hewlett-Packard Enterprise • ZSM(22)000235 ZSM011 - Clause 5.6.x IME registry - (GOMES Pedro Henrique, Ericsson) • ZSM(21)000242r1 ZSM011 section 6 Potential capabilities for intent-driven notification capabilities - Deutsche Telekom AG
July 2022	0.2.1	Incorporated contributions: - ZSM(22)000192r3_ZSM011_Sec_5_9_Intent-system_Testing_Framework - ZSM(22)000200r4_ZSM011_Sec_4_x_Purpose_of_Intent - ZSM(22)000223r2_ZSM011_Sec_3_1_Terms_update - ZSM(22)000235r1_ZSM011_-_Clause_5_6_x_IME_registry - ZSM(22)000202r3_ZSM_011_Addition_to_section_5_2 - ZSM(22)000143r2_ZSM011_Changing_section_5_5_life_cycle - ZSM(22)000080r2_ZSM011_Adding example of intent in RDF to Annex A
July 2022	0.2.2	Incorporated contributions: - ZSM(22)000280 ZSM011, 5.6.6 Intent Management Entity registry; Text change proposal regarding consistency of the description (DT AG) - ZSM(22)000282 ZSM011, 5.6.6 Intent Management Entity registry; Text change proposal regarding clarity of the description (DT AG) - ZSM(22)000287 ZSM011_Editorial_changes on clause 2 (Ericsson LM) - ZSM(22)000289r1 ZSM011_Editorial_changes on clause 4 (Ericsson LM) - ZSM(22)000290 ZSM011_Editorial_changes on clause 5 (Ericsson LM) - ZSM(22)000291r1 ZSM011_Move clause 6.1 and removing clauses 6 and 7 (Ericsson LM) - ZSM(22)000292 ZSM011_Editorial_changes on Annex A2 and A4 (Ericsson LM) - ZSM(22)000302 ZSM011 5.4.1 Change intent handling entity to intent management entity (ZTE Corporation) - ZSM(22)000224r4 ZSM011 Appendix: Alternative Intent based concepts (HPE) - ZSM(22)000300r1 ZSM011_Changes on clause 5.6.3 (Ericsson LM) - ZSM(21)000242r3 ZSM011 section 6 Potential capabilities for intent-driven notification capabilities (DT AG) - ZSM(22)000288r2 ZSM011_Editorial_changes on clause 3
September 2022	0.2.3	- ZSM(22)000333 ZSM011 Addition to Sub-Section 4.2.1 (JANZ Christopher, Huawei) - ZSM(22)000307r3 ZSM011 Additions to Sub-Sections 4.2.1 and 4.2.3 (JANZ Christopher, Huawei) - ZSM(22)000286r2 ZSM011_Changes on clause 1 (Ericsson LM) - ZSM(22)000306r1 ZSM011 Addition of Introductory Clause to Sub-Section 5.6.3.2 (JANZ Christopher, Huawei) - ZSM(22)000301r2 ZSM011 Inconsistency in term usage (ZTE Corporation)
October 2022	0.2.4	- ZSM(22)000296r2 ZSM011_Conclusions of the study (Ericsson LM) - ZSM(22)000098r3 ZSM011 5.6.z Verification of intent outcome - optional interface capability (Deutsche Telekom AG, Ericsson LM)
October 2022	0.2.5	- ZSM(22)000355 ZSM011 Alignment of the description for the responsibility regarding modifying and deletion of an intent (KLOTZ Michael, DT) - ZSM(22)000335r2 ZSM011 Change intent object to intent information object - (CHU Junsheng, ZTE)
November 2022	0.2.6	Editorial changes to solve drafting rules infringements.

Date	Version	Information about changes
November 2022	0.2.7	- ZSM(22)000378 ZSM011_modifying_informative_ietf (Huawei Technologies France) - ZSM(22)000396 ZSM011_review_comments (CHEN wenjing, DOCOMO Communications Lab.)

History

Document history		
V1.1.1	February 2023	Publication