ETSI GS ZSM 016 V1.1.1 (2024-10)

Zero-touch network and Service Management (ZSM); Intent-driven Closed Loops

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
DGS/ZSM-016_IntentDrvCL

Keywords
closed loop control, intent management

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

Modal verbs terminology

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

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

1 Scope

The present document specifies capabilities to support the combination of closed-loop automation with intents originating from ZSM consumers, focusing on intent-driven governance and coordination of closed loops. The present document includes use cases and additional requirements related to intent-driven aspects of ETSI GS ZSM 009-1 [2]. The present document creates a normative specification covering stages 1 and 2. It also identifies and describes additions to ETSI GS ZSM 002 [1] and ETSI GS ZSM 009-1 [2], as needed. Related work in ETSI, other SDOs and open-source projects are considered and used where applicable.

2 References

2.1 Normative 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.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

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 necessary for the application of the present document.

[1] ETSI GS ZSM 002 (V1.1.1): "Zero-touch network and Service Management (ZSM); Reference Architecture".

[2] ETSI GS ZSM 009-1 (V1.1.1): "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers".

[3] ETSI TS 128 312 (V17.3.1): "LTE; 5G; Management and orchestration; Intent driven management services for mobile networks (3GPP TS 28.312 version 17.3.1 Release 17)".

[4] ETSI GS PDL 011: "Permissioned Distributed Ledger (PDL); Specification of Requirements for Smart Contracts' architecture and security".

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 011 (V1.1.1): "Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects".

[i.2] ETSI GR PDL 004: "Permissioned Distributed Ledgers (PDL); Smart Contracts; System Architecture and Functional Specification".

[i.3] ETSI GR ZSM 009-3: "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 3: Advanced Topics".

[i.4] ETSI GS ZSM 018: "Zero-touch network and Service Management (ZSM); Network Digital Twin for enhanced zero-touch network and service management".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

intent handler: logical entity that receives intents and handles them in the domain that is responsible for that intent's fulfilment

intent owner: logical entity that originates intents and is responsible for managing intents lifecycle

operational intent: intent related to operational goals (e.g. operational goals of a customer-facing services)

NOTE: Example of operational intent and its operational goal is, e.g. minimize global power consumption.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3GPP	3G (mobile) Partnership Project
5G	5nd Generation (mobile networks)
BSS	Business Support System
CL	Closed Loop
CLG	Closed Loop Governance
CPU	Central Processing Unit
CRUD	Create Read Update and Delete
CSP	Communication Service Provider
DL	Distributed Ledger
E2E MD	End-to-end Management Domain
E2E	End to End
eMBB	enhanced Mobile Broad Band
IdCL	Intent-driven Closed Loop
IdCLG	Intent-driven Closed Loop Governance
IME	Intent Management Entity
KPI	Key Performance Indicator
LCM	Lifecycle Management
MD	Management Domain
mIoT	Massive Internet of Things
MnS	Management Service
NDT	Network Digital Twin
PDL	Permissioned Distributed Ledgers
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
SDO	Standard Defining Organization
SLA	Service-Level Agreement
TMF	TM Forum
URLLC	Ultra Reliable and Low Latency Communications
VNF	Virtualised Network Function

4 The concept of Intent-driven Closed Loops

4.1 Background

4.1.1 Introduction

Intents may be used as an abstraction among autonomous entities that need to work towards a common goal (the term autonomous entities is defined in ETSI GR ZSM 011 [i.1]). As defined in ETSI GR ZSM 011 [i.1], intents define goals, requirements, and constraints in a declarative form. Thereby, intents guide an autonomous network regarding expectations from the customer on service or network behavior.

The E2E MD and MDs may support intent-based services as mentioned in ETSI GS ZSM 002 [1], clause 4.2.10. MDs that support intent handling operations shall use an Intent Management Entity (IME) (ETSI GR ZSM 011 [i.1]) for all intent-related management. The IME shall act as an intent owner and/or intent handler. The IME as an intent owner manages the intent life cycle.

In the present document, the features/capabilities of IME functionality are identified, discussed and specified.

IMEs shall leverage Closed Loops (CLs) functionalities, e.g. specify a goal to a CL instance, as part of the intent Life Cycle Management (LCM). CLs are key enablers for automating the configuration of networks and steering them towards the desired state, since a CL has the objective of achieving a specific goal by monitoring and regulating a set of managed entities.

The present document focuses on combining the two concepts (intents and CLs). Both concepts are essential to realizing intelligent and autonomous networks by enabling the usage of intent LCM and closed loops to fulfil intents within the scope of the ZSM framework and enable intents to interact with CLs.

The information exchanged between two IMEs is:

1) the intent requirements (defined by the IME in the role of intent owner) communicated to another IME (in the role of intent handler); and

2) the intent fulfilment report sent back by the latter.

4.1.2 LCM and closed loops to fulfil intents in ZSM

The IME (in the role of an intent owner) is aware of the system state through measurements, analytics, and other information available. The IME shall execute actions using CLs to handle the network resources to fulfil the requirements expressed by the intents. These actions shall be performed through conventional management interfaces or intent-based interfaces. Regarding intent-based interfaces, the IME shall define its requirements through consecutive intents and interact with IMEs between different MDs using intent LCM operations. CLs may exist in any of the management domains of the ZSM architecture and the IME is responsible for managing such CLs, including their LCM.

Figure 4.1.2-1 shows an example of using intent LCM for communication between different MDs within a management system based on the ZSM framework. The interaction starts with the E2E service MD receiving an intent from the ZSM framework consumer. Based on fulfilment purposes, the E2E service MD may decompose the intent into multiple intents and send these decomposed intents to different MDs (RAN, Core, Transport, etc.). By doing that, the E2E service MD shall assume the following roles:

1) **intent handler** of the intent originated at the ZSM consumer; and

2) **intent owner** of the decomposed intents sent downwards to the specific MDs.

After receiving the intent, each MD may use closed-loop automation mechanisms to fulfil it and report the fulfilment status to their corresponding intent owner, thus closing the interaction loops.

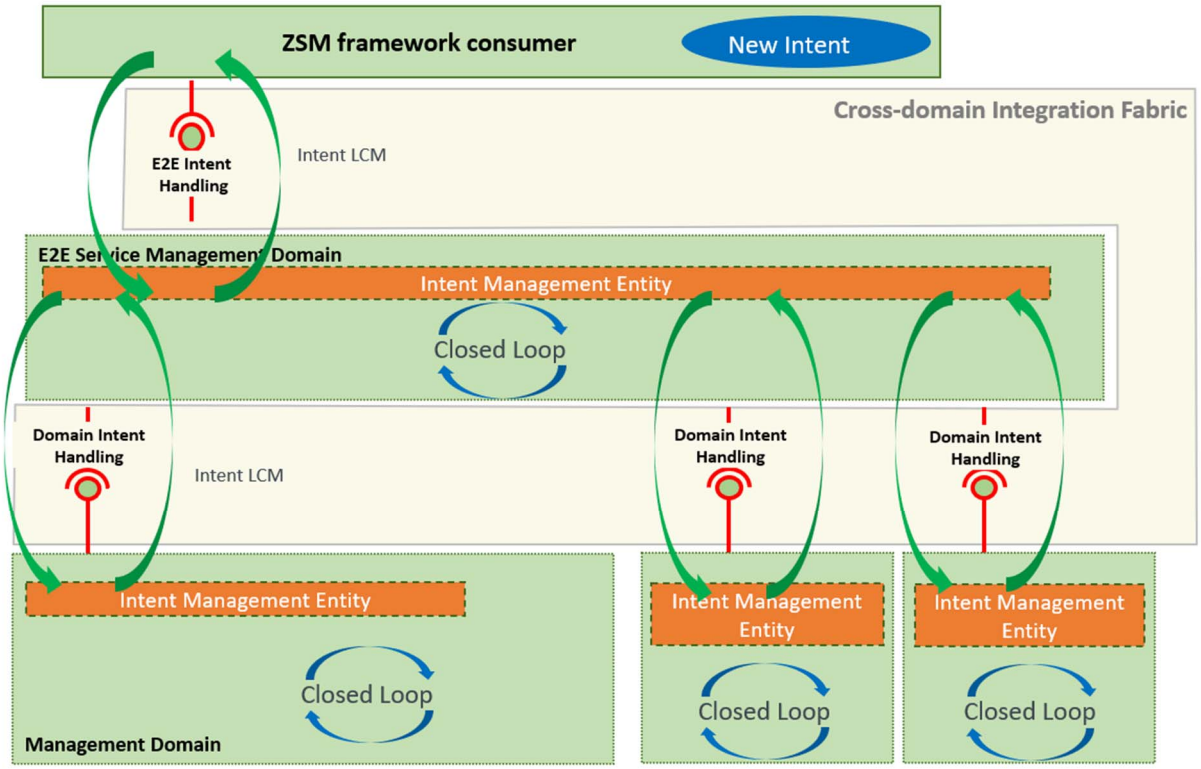

Figure 4.1.2-1: Intent management entities interactions among different management domains in the ZSM architecture

The implementation of IMEs and the use of closed loops may be realized in different ways. However, a general challenge is how the IMEs shall interact with the specific logic within a management domain, in order to translate an intent, expressing expectations, into detailed technical configurations.

4.2 Examples of use cases

4.2.1 Intents for eMBB slices

4.2.1.1 Description

In the era of 5G and "5G-Advanced", which 3GPP started to study and specify from Release 18, it is expected to enhance 5G specific capabilities such as eMBB, URLLC and mIoT. In regards to eMBB utilizing 5G network, CSPs can provide more high-speed communication environment not only for downlink communications, but also for uplink ones. This enhancement may make it easier for CSPs to provide eMBB network slices with customers more widely than current 5G.

In those situations, it is also expected that CSPs experience unknown anomaly status in the operator's network due to e.g. the drastic increase of traffic. To avoid failing to meet customers' and operator's expectations for eMBB service, i.e. "intents", intent-driven closed loops can help CSPs detect those incidents promptly and identify the root cause and appropriate actions to be taken.

4.2.1.2 Use case details

This use case can be done by the following procedure and closed-loop is adopted with intent-based:

- [Pre-condition 1] A CSP is providing broadband network services to end users utilizing eMBB network slice(s).

- [Pre-condition 2] SLA between a CSP and customers may be transformed as intents within a ZSM framework. In addition, CSP may give intent handler(s) their operational requirements (e.g. wants to reduce operational cost as much as possible without increasing power or to reduce the virtual servers when the number of active users are decreasing, or wants to increase customers' satisfaction by replying to their claims as soon as possible, etc.) as intents:

    1) While monitoring metrics (e.g. traffic data or QoE on users), E2E Service MD detects violation of SLA. [Monitoring stage of a CL].

    2) By conducting intent-based analysis, an anomaly status such as a congestion in a MD is identified as a root cause of the violation. [Analysis stage of a CL].

    3) Based on the analysis, some candidates of resolution (e.g. scaling out of resources, switching the route of users' traffic to more efficient one, etc.) are derived. [Analysis stage of a CL].

    4) An intent handler within the MD decides which action should be taken to meet the intent most properly. [Decision stage of a CL].

    5) Adopted execution is done. [Execution stage of a CL].

    6) Upon the action is executed, the status of fulfilment of the intent is evaluated as defined in clause 6.2.1.3.5 of ETSI TS 128 312 [3].

4.2.2 Intent-based closed loops in cross-domain management

This use case describes autonomous network operations based on the ZSM architecture involving an end-to-end management domain and different management domains. Operations begin with an intent expressing abstract business requirements.

NOTE: For illustration purpose, throughout this use case, it is considered the delivery of a cost-effective video service to a group of users. However, this use case is generic enough to be applicable to other types of services and other types of user requirements.

The business-level intent specifies the type of video service, high-level preferences, such as cost-effectiveness, and the user group as the context. This business-level intent may be received by a BSS or any other system that is outside the scope of ETSI ZSM. This business-level system would translate the intent into a service-level intent that can be forwarded to an Intent Management Entity (IME) that is under the scope of ETSI ZSM and is running at E2E MD level. This translation would involve some contextualization by interpreting and transforming the abstract business requirements into service-specific requirements and goals.

The ETSI ZSM framework allows any external authorized ZSM consumer to discover available IMEs and forward an intent to them. An IME in the E2E MD receives the service-level intent, and its primary objective is to deliver a service instance that satisfies that intent. In the illustrative example, a key aspect of this delivery process is the best distribution of application and network components across the network and cloud infrastructure.

There are many possible ways of processing, fulfilling, and assuring the received intent; one common mechanism is with closed loops. The number of closed loops and their types employed by the IME is implementation-specific. One example of such intent-driven closed loop is described in ETSI GR ZSM 009-3 [i.3], clause 6.5.

During the process of finding a solution to fulfil the service intent, the closed loops employed by the IME (as the MnS producer) at the E2E MD will break down the requirements on the service into domain-level requirements per network component that is necessary to be deployed. For example, a network function would be required at a particular location with latency and bandwidth goals. This would help deliver the required user experience.

Executing the solution at the service level typically involves a service orchestrator that coordinates the execution of distributed actions. Some of these actions may involve sending decomposed intents to individual MDs that are intent-aware and have a domain-specific IME. The ETSI ZSM framework allows IMEs at different MDs (including the E2E MD) to exchange intents, that are related to customer intents.

In this example the focus is on intent-based management of a cloud-native function. The closed loops dealing with the service intent determine a solution that is primarily concerned with identifying all deployment functions and allocating to a data center. Deployment function candidates are still technology-agnostic, but it is assumed that there is a direct mapping from each function to some deployment artifact that can be understood by the orchestrator or virtualization manager of the management domain.

Ultimately, the decomposition of the business-level intent provides a service instance design that is broken down into locations (such as data centers and transport paths), deployment artifacts (represented by Helm diagrams, for example), and initial resource allocation in terms of bandwidth, QoS class, storage, and so on.

Once the service is instantiated according to the preferred solution, information becomes available that enables service monitoring. This means that the intent-based closed loop(s) that was initially aiming at service fulfilment will be transformed into (or replaced with) specific closed loops with assurance goals that will be responsible to take corrective action when the intent requirements do not match the measured system state.

4.2.3 Intent conflict detection and resolution

4.2.3.1 Description

Intent related topics are being studied and specified by ZSM and different SDOs, such as TMF or 3GPP. One relevant aspect in an intent-driven system is that it is expected that conflicts between intents will happen, and they need to be detected and resolved. Intent conflict needs to be detected if any of an intent's requirements is conflicting with any requirement within the intent or with another intent. ETSI GR ZSM 011 [i.1], clause 5.7.2 defined three different levels of conflict: syntax-level, action-level, and impact-level. A syntax-level conflict refers to a conflict between two or more components of an intent expression. 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. 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.

An intent handler detects conflicts and tries to resolve them with available information and means. An intent handler may, to increase information and options available and useful in conflict resolution, request that an intent owner provide explicit priorities, or ordered lists of priorities, among intents, intent expectations and/or targets. After detecting the conflict and informing the intent owner about its intent degradation as part of intent report (in case the intent handler failed to solve the conflict), the intent handler may recommend to the owner to retry operation after "x" minutes based on which the intent owner may retry the operation after the stipulated time.

NOTE: When deriving solution, priority information needs to be taken into consideration if it is provided by intent owner. The priority information for intent, intent expectations, expectation targets may be provided by the intent owner with different means (e.g. an ordered list, etc.). The priority given to the intents is only relevant when dealing with intents coming from the same intent owner. An intent owner is not able to require priority over another intent from a different owner.

Intent handler is responsible for reporting, intent degradation, detecting and undertake the solutions to resolve intent conflicts.

4.2.3.2 Use case details

The present clause describes the sequence on how the Intent conflict may be detected and resolved in the ZSM framework:

1) While monitoring metrics (e.g. fulfilment of an intent), the E2E Service MD or other MD detects that the intent owner's intent is not fulfilled. [Monitoring stage of a CL].

2) By conducting intent-based analysis, anomaly status, such as two intent expectations conflicting with each other, is identified as a root cause of intent conflict. [Analysis stage of a CL].

3) Based on the analysis, the intent handler proceeds to either one of the steps stated below:

   a) Intent handler resolves the conflict and fulfils the intent by using the intent management service as defined in clause 7.1. Then, proceed to step 6. [Decision & Execution stage of a CL].

   b) If the intent handler cannot fulfil the intent, it shall report non-fulfilment status to the intent owner. If, however, in addition, the intent handler finds alternatives to the detailed intent that it would be able to fulfil, it may report these to the intent owner, using the Judge operation, for consideration by the intent owner. Then, proceed to step 4. [Analysis stage of a CL].

4) The intent owner sends the response to the Judge operation by choosing the preferable outcome, provided in step 3 if it is provided, etc. Then the intent handler takes the decision leveraging this information. [Decision stage of a CL].

5) Intent handler fulfils the intent by using the intent management service as defined in clause 7.1. based on the decision of the intent owner. [Execution stage of a CL].

6) Upon the action being successfully executed, the status of the fulfilment of the intent is evaluated. The information and the status of intent fulfilment is reported to the intent owner. If the fulfilment of the intent is not fulfilled, return to step 2 for re-analysis.

4.2.4 Intent-driven composition and configuration of CL (and stages)

4.2.4.1 Description

This use case describes possible methods of composition and configuration of an intent driven closed loop (CL). On this aspect, two types of scenarios can be considered which can be either made-to-order closed loop or ready-made closed loop as defined by ETSI GS ZSM 009-1 [2], clause 7.4:

Scenario#1: the pre-existing CL is used for intent fulfilment.

- This is the default scenario, where the CL required to fulfil the intent is pre-existing in the ZSM framework.
- The intent handler may use a new instance of the CL.

Scenario#2: the made-to-order closed loop is created dynamically by intent handler based on the intent by utilizing the services in the ZSM framework.

NOTE: The intent handler, based on the intent expectation, may decide to build the made-to-order CL inline with the description of ETSI GS ZSM 009-1 [2].

As an example, consider a scenario where the intent expectation is to configure a new network slice to cater to certain geographic area:

- This intent expectation can be fulfilled with a CL capable of fetching the sites serving the specified geographic area and provisioning the network parameters required for the new slice to the identified sites.
- To realize this, the intent handler prepares a made-to-order CL involving:
  1) fetching the sites involved in serving the geographic area in the scope (monitoring stage);
  2) analysing the needed configuration (analysing stage);
  3) deciding on the configuration parameters required for the new slice (decision stage); and
  4) provisioning the configuration parameters to the network (execution stage).

4.2.5 Intent-driven composition and coordination of nested closed loops

4.2.5.1 Description

In certain scenarios, in order to fulfil an intent from an intent owner, the intent handler needs to invoke certain closed loop which may need to coordinate with one or more subordinate CLs to fulfil the intent. When multiple level of such CLs are involved, this is called nested/ hierarchical CL as described in ETSI GS ZSM 009-1 [2], clause 8.2.3. Different possible scenarios on the hierarchical CL is described in ETSI GR ZSM 009-3 [i.3] as shown in Figures 4.2.5.1-1 and 4.2.5.1-2. As depicted in the figures and described in ETSI GR ZSM 009-3 [i.3], the different stages of the superior CL may interact with stages in subordinate CL to control or coordinate with the subordinate CL. Similarly, stages in subordinate CLs may interact with stages in other subordinate CLs.

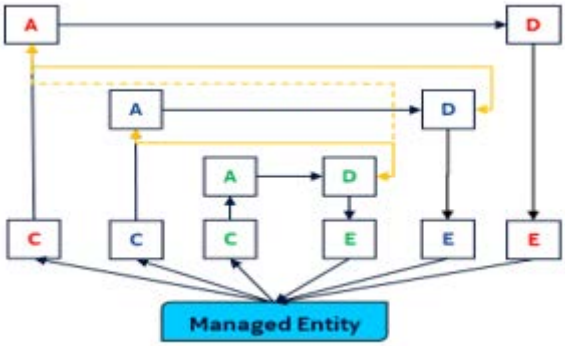

Figure 4.2.5.1-1: Solution 1
(Source: clause 6.3.2.1 of ETSI GR ZSM 009-3 [i.3])

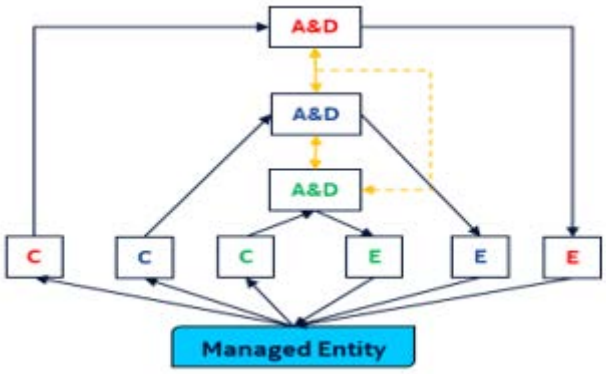

Figure 4.2.5.1-2: Solution 2
(Source: clause 6.3.2.2 of ETSI GR ZSM 009-3 [i.3])

This use case describes the intent driven composition and coordination of nested (hierarchical) CLs where the intent handler may compose (using ZSM framework capabilities) the superior and subordinate CLs for the nested CLs in order to fulfil an intent from an intent owner (see the example below). In this use case the intent handler may use ZSM capabilities to coordinate the interaction between the superior and subordinate CLs. The intent handler may also dynamically compose the CL with different stages required for the closed loop and with utilizing ZSM services supporting interaction between different stages of superior and subordinate CLs.

One of the examples for this use case is cloud resource auto-scaling where the resource autoscaling can be classified into vertical autoscaling, horizontal autoscaling and node autoscaling. In vertical autoscaling, the CPU resources are automatically scaled up/down from an already instantiated Virtualised Network Function (VNF) based on the workload requirements. In horizontal autoscaling, new VNFs are automatically instantiated, or existing VNFs are automatically released as per the workload requirement. In node autoscaling, the number of nodes in a given node pool is automatically resized based on the demands of the workloads. Based on the intent from the intent owner on auto-scaling, intent handler can compose and instantiate nested CLs for resource autoscaling, in which, the Innermost CL is for vertical VNF autoscaling, Intermediate CL is for horizontal VNF autoscaling and Outermost CL is for node autoscaling. However, one limitation in most virtual/physical infrastructure managers is that both vertical VNF auto-scaler (entity which handles autoscaling) and horizontal VNF auto-scaler objects cannot be used together for resource autoscaling due to possible conflicts in their actions. In this case, the intent handler will be able to coordinate the dependency between the different auto-scalers by handling the composition and coordination of the closed loops responsible for the respective auto-scalers in such a way that the CLs can coordinate with each other.

5 Requirements for intent-driven Closed Loops (CLs)

5.1 Requirements

Req-1: The ETSI ZSM framework shall expose intent-driven management services to authorized ZSM consumers.

Req-2: The ETSI ZSM framework shall provide discoverability of the intent-driven management capabilities offered to authorized ZSM consumers.

Req-3: The ETSI ZSM framework shall provide the registration service for intent management entities to register their intent management capabilities.

NOTE: Examples of intent management capabilities are, among others, the scope of management, the supported intent modelling, intent-based interface operations.

Req-4: The ETSI ZSM framework shall support the capabilities to process, fulfil and assure intents.

Req-5: The ETSI ZSM framework shall support the capabilities to allow ZSM consumers to state intents as a set of expectations, including requirements, goals, and constraints.

Req-6: The intent handler should be able to execute a pre-existing Closed Loop (CL) to fulfil an intent.

Req-7: The intent handler should be able to compose, activate and execute a made-to-order Closed Loop (CL) to fulfil an intent.

The following requirements need to be fulfilled by the ZSM framework to handle the conflicts in intent driven closed loops:

Req-8: The intent owner should be able to send to the intent handler its utility information using intent.

Req-9: The intent handler shall be able to report to the intent owner about possible intent degradation.

Req-10: Intent handler should be able to register its capability to support for utility through registration and discovery MnS (as defined in ETSI GS ZSM 002 [1]).

Req-11: The ZSM framework shall support capability for the intent handler to detect conflicts between different intents coming from same or different intent owners.

Req-12: The ZSM framework shall support capability for the intent handler to detect conflicts within an intent.

Req-13: The ZSM framework shall support the intent handler capability for composition of interdependent CLs, superior CLs and subordinate CLs based on the intent it receives from intent owner.

Req-14: The ZSM framework shall have the capability for coordination between interdependent CLs, superior CLs and subordinate CLs.

Req-15: The ZSM framework should support the capability for the intent handler to fulfil intents based on priorities specified by an authorized intent owner for the intents.

5.2 Intent owner/handler system expected behaviours

5.2.1 Intent owner

1) The intent owner is expected to create an intent expressing its requirements, goals, and constraints with properly constituted intents.

NOTE: Properly constituted intents are any intents the intent handler can interpret without ambiguities, which also means they need to agree on the models used.

2) The intent owner is expected to change or remove the active intents as it may see fit.

3) The intent owner is expected to specify the occurrence and conditions of the intent reports through reporting expectations in the intent.

4) The intent owner is expected to find the most suitable intent handlers to deal with the intent requirements.

5) The intent owner is expected to send the preferred solutions when receiving a trigger for the judge operation.

5.2.2 Intent handler

1) The intent handler is expected to find strategies and plan actions that would achieve what the intent is asking for until the intent owner removes the intent (in case the intent is fulfilled).

2) The intent handler is expected to register its capabilities to be discovered by any intent owner and keep this information up to date.

3) The intent handler will not respond to create, modify, or remove intent instructions from any source other than the owner of that intent.

4) The intent handler is expected to send reports containing the expected result of a intent or the most challenging requirements when receiving a trigger for an intent feasibility check.

5) The intent handler is expected to send reports containing the best result the handler could achieve when receiving a trigger for the best intent.

6) The intent handler is expected to send an intent report (informing acceptance or rejection, as well as providing the reason for rejection) to any communication from the owner about the setting, modification, or removal of intent.

7) The intent handler is expected to be able to periodically send individual intent reports (only on the relevant requirements defined by the owner) containing the status and success of intent handling to the intent owner (as determined or modified by the intent owner). If any intent degradation occurs, report it to the owner immediately.

6 Governance and Coordination of intent-driven Closed Loops

6.1 Introduction

Intent-driven Closed Loop Governance (IdCLG) inherits all capabilities provided in ETSI GS ZSM 009-1 [2], clause 8.1.

The intent-driven Closed Loop Governance (IdCLG) capabilities may include:

- Management of the Closed Loops lifecycle using intents, e.g. to provide the capability for an IME to manage CLs using the intent-related information extracted:
  - From intents.
  - During the intent LCM operations.
- Converting the information collected from the CLs to intent report format.

To make use of the intent-driven Closed Loop Governance, an IME shall have the capability to deal with intent operations (see clause 6.3).

6.2 Governing an intent-driven closed loop

6.2.1 Introduction

As defined in ETSI GS ZSM 009-1 [2], CL Governance provides the capabilities to manage and configure the CL models and the life cycle of CLs. Besides, it also provides capabilities to retrieve information from the CL status and performance. Regarding IdCLs, for each intent received, the intent handler may use existing CL instances or (in case a new service is needed) use CL models to instantiate a new CL to decide the most appropriate actions to fulfil the intent.

Intent-driven CLs may exist in any of the management domains of the ZSM architecture. A management domain that supports intent handling operations shall use IdCLG services (see clauses 7.1 and 7.2).

After receiving the intent(s) from an intent owner, the intent handler may take this intent-related information to perform CL lifecycle management operations (e.g. the CL LCM operations defined in ETSI GS ZSM 009-1 [2]) on one or more CLs to achieve the intent targets. These operations may involve assigning their goal(s) for intent fulfilment besides instantiating new CL instances. After manipulation of the intent-related information, the intent handler may use CL models from CL Governance MnS to set the goal(s) on existing (or new) CLs. ETSI GS ZSM 009-1 [2] specifies that goal(s) shall be stated in declarative or imperative forms; the former in a level of abstraction closer to intents and the latter with measurable service levels specifications. The conversion process may involve converting intents into measurable KPIs to be monitored by the CL instance.

As defined in ETSI GS ZSM 009-1 [2], clause 5.2, the CL stages should be able to report their outcomes to authorized entities. This way, reporting information shall be obtained using CL status information and (or) performance information capabilities from the CL Governance MnS. This obtained information shall be converted into intent reports before being sent back to the intent owner that originated the intent(s). An example of this collected information from the CL is the health status of the CL.

After a CL is associated with an intent (or a set of intents), the intent handler needs to report the fulfilment status and progress to the entity that originated that intent(s). Figure 6.2.1-1 shows how intents may interact with CLs using the CL Governance MnS extended capabilities, e.g. using the IdCLG.

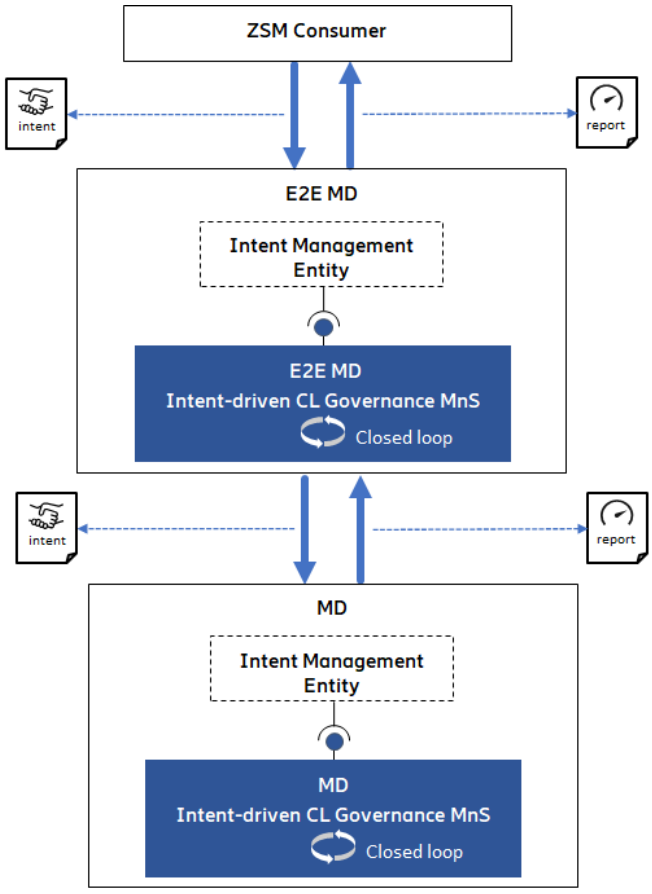

Figure 6.2.1-1: Intent-driven Closed Loop Governance interaction

6.2.2 Smart Contracts

Distributed Ledger (DL) offer the ability to store any kind of data as a consensus of replicated, shared, and synchronized digital records distributed across multiple sites and domains, without depending on any central administrator. Together with their properties regarding immutability (and therefore non-repudiation) and transparency (implying multi-party verifiability), they open a wide range of applications in what relates to the verification of the fulfilment of service agreements of any nature.

In this context, smart contracts appear as an interesting solution for the formalization and automation of governance mechanisms. A smart contract is a computer program stored in a DL system, wherein the outcome of any execution of the program is recorded on the DL [i.2]. Smart contracts are executable code that lives on a DL and inherit their properties such as transparency and immutability. They are also auto-executable, which means once recorded, a smart contract can perform tasks without human intervention. Indeed, a constructor initializes them in the first place; however, subsequent clauses are executed automatically with pre-programmed conditions. Smart contract specifications (ETSI GS PDL 11 [4]) require the adhesion to the lifecycle of smart contract proposed in ETSI GR PDL 004 [i.2] (see Figure 6.2.2-1).

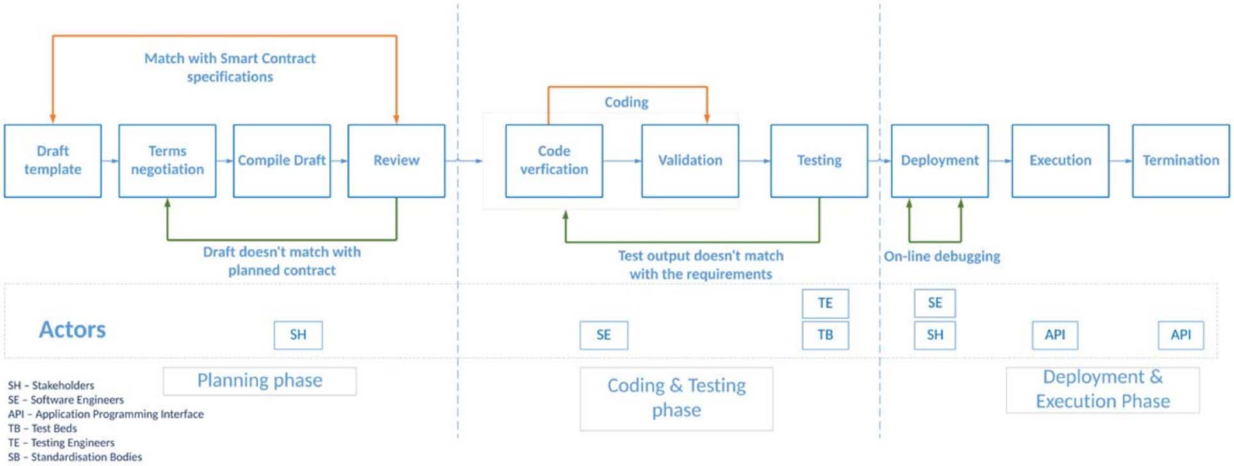

Figure 6.2.2-1: Lifecycle of a smart contract
(Source: ETSI GR PDL 004 [i.2])

The use of smart contract helps in building formal and traceable mechanisms for intent-driven closed loop governance. On the one hand, intent declaration is formalized through the smart contract code (conditions or trigger, and actions or code), i.e. the intent language is captured in the smart contract itself. On the other hand, the formalized intent is used to create and supervise the required closed loops over the applicable services.

The application of smart contracts to intent-driven closed loop governance is performed as follows:

1) The intent owner discovers available intent handlers. The owner may want to know the capabilities supported by each intent handler, including information on syntax rules for intent specification.

2) The intent owner can use this information to select the proper intent handler to express the intent.

3) The intent owner formulates the intent according to the syntax rules of selected intent handler and sends the intent provisioning request to this intent handler.

4) Upon receiving such a request, the intent handler proceeds with intent feasibility check:

   a) If feasible, the intent handler signs the associated smart contract, together with the intent owner, and registers it in the DL. Once signed and registered, the smart contract will provide a transparent and immutable representation of the intent.

   b) Otherwise, an intent negotiation procedure can start between the intent owner and intent handler.

5) The intent handler fulfils the intent using one or more CL instances and uses the smart contract to govern the LCM of those instances. This governance relies on the triggers/actions which are captured in the smart contract.

NOTE: In a service provider-customer relationship, the smart contract is an auto-executable code which includes the necessary triggers and actions.

6.3 Coordination among intent-driven closed loops

6.3.1 Closed Loop Coordination for Intent Conflict Resolution

Clause 4.2.3 considers types and sources of conflicts that can arise within and among intents, or in the attempt to deliver customer-facing services governed by intents. The present clause considers methods and means of detecting and resolving such conflicts, in particular with respect to the coordination of closed loops.

Detection of intent conflicts can sometimes be accomplished through semantic or logical inspection of intents. This may be the case, for example, if firmly contradictory objectives form part of the same effective intent, e.g. "*apply maximum security to purchasing department user operations*" and "*apply medium security to all enterprise user operations*". Such cases amount to errors in intents which, if detected outright, can be communicated as such to intent owners during intent negotiation or re-negotiation operations. These conflicts are referred to in clause 4.2.3 and ETSI GR ZSM 011 [i.1] as syntax level conflicts.

In other cases, however, the existence of a conflict is a matter of degree and circumstance.

EXAMPLE 1: "*apply bandwidth not less than X to connections*" and "*do not exceed connectivity service cost Y*"; these objectives tend toward contradiction, in the sense that more bandwidth uses more network resource while the use of more network resource generally costs more. However, it usually cannot be clear *a priori* whether these specific objectives are strictly irreconcilable.

In such cases, it is in the attempt to find fulfilment solutions satisfying all intent-defined objectives that the effective existence and degree of conflicts are discovered. This is true of conflicts arising in respect of:

1) Components of an intent governing a single intent and its corresponding assurance closed loop: such conflicts are referred to in clause 4.2.3 and ETSI GR ZSM 011 [i.1] as action level conflicts.

2) Among assurance closed loops that:

   a) govern the assurance of multiple customer-facing services; or

   b) govern the simultaneous assurance of intents governing both customer-facing services and operational objectives. Such operational objectives may be set by the network operator, e.g. minimize global power consumption.

   These conflicts are referred to in clause 4.2.3 and ETSI GR ZSM 011 [i.1] as impact level conflicts.

Assurance that customer-facing services are operating in compliance with their defining intents is achieved provided by assurance closed loops that are operated and managed by intent handlers. Ongoing adherence to intent-defined goals, of operating customer-facing services, may differ among those services and has to be determined separately, based on data, for each one.

If the managed resources used to fulfil customer-facing services are mutually independent, then analysis, decision-making and action may be undertaken independently in respect of each customer-facing service. In this case, a complete and independent closed loop may be operated in respect of each customer-facing service, and these closed loops are effectively peers in the sense of ETSI GS ZSM 009-1 [2], clause 8.2.2 and as depicted in ETSI GR ZSM 011 [i.1], clause 5.2. A basis for operation of a closed loop coordination service, per ETSI GS ZSM 009-1 [2], clause 8.2.5 and referred to as an intent reconciliation service in ETSI GR ZSM 011 [i.1], clause 5.2, could be to ensure independence of managed resources allocated for prospective use by each assurance closed loop, perhaps on a dynamic and needs-driven basis. This may be considered a pre-execution coordination action per ETSI GS ZSM 009-1 [2], clause 8.2.5.4. In this scenario, operational intents might be handled by adding some version or derivative of their objectives to the various intents defining customer-facing services.

EXAMPLE 2: If there is a total managed resource power consumption operational intent imposed on an intent handler, an allowable managed resource power consumption limit could be set independently for each customer-facing service -perhaps dynamically - by a closed loop coordination service. Those limits would become components of the intents governing the respective customer-facing services.

Where managed resources allocated to intent-defined customer-facing services are not necessarily kept independent, then actions determined and undertaken in respect of such services are not assured to be mutually independent: conflicts may arise. The analysis, decision-making and action operations of closed loops governing such services, has to be either coordinated or merged. Among interdependent peer closed loops, the action plan conflict detection and action plan selection processes, described in ETSI GS ZSM 009-1 [2], clause 8.2.5.4 for a pre-execution coordination service (a type of closed loop coordination service), are applicable.

Such closed loop coordination depends on the generation of conflict-free action plans by the analysis and decision-making processes of the various closed loops. Those processes has to be influenced so as to converge to such plans. An alternative is described in ETSI GR ZSM 009-3 [i.3], clause 6.4, wherein analysis (leading to prospective solutions), decision-making (resolving solutions for action) and action operations - in respect of what otherwise would be independent closed loops - are merged. In effect, solutions are sought that satisfy at once, or best satisfy at once, all intents impinging on use of the shared set of managed resources. Such intents may be of both types (2.a) and (2.b) above. "Best satisfy" means, in the event that no solution may be found that strictly satisfies all intents, a solution that comes closest to doing so, where closest is defined in some rigorous - ideally quantitative - sense. Utility functions, considered in clause 6.3.2, are a quantitative mechanism well-suited to this purpose. Utility functions may also be used for controlled balancing of competing objectives within a single composed intent, per (1) above.

The utility of Network Digital Twins - NDTs (ETSI GS ZSM 018 [i.4]) - in this context is quite clear. The evaluation - per Figure 6.4.1-1 in ETSI GR ZSM 009-3 [i.3], clause 6.4 - of prospective solutions passes through their *virtual actuation* on the NDT. Virtual measurements and issue identification, based on the NDT-generated outcomes, enable new analysis and prospective solution identification. This continues until a satisfactory - e.g. "best" per above - solution is found; only then does actuation of the solution on the physical network occur.

Closed loops of type (2.b) above may be nested with/around customer-facing service delivery and assurance closed loops - type (2.a) above - in hierarchical fashion, per the discussion in ETSI GR ZSM 009-3 [i.3], clause 6.2. For example, a closed loop attempting to minimize the total power consumed by resources delivering a set of customer-facing services, may be nested with/around the various assurance closed loops governing those services, using the mechanisms described in ETSI GR ZSM 009-3 [i.3], clauses 6.2.2 to 6.2.4. Those mechanisms, it should be noted, require the transmission of "insights and workflows" among components of the different closed loops in the hierarchy. In other words, as in the case of coordination of interdependent peer closed loops, analysis and decision-making processes has to influence each other across closed loops, in order to drive convergence toward a globally satisfactory or best solution. Again, merged analysis and decision-making, per above, offers an alternative realization path.

Prioritization is another possible mechanism for intent conflict resolution. Intent owners may specify relative fulfilment priorities among their intents, or among different components of individual intents. Utility functions are one means of specifying relative priorities, but absolute priorities may also be specified. All such prioritization information represents information that handlers may use to attempt to resolve or ease conflicts. Network operators may also impose relative priorities among customers, customer-facing service classes, their own independently set objectives, etc. Further, prioritization may be requested, by handlers, of owners of customer-facing service intents among which conflicts are detected.

6.3.2 Utility information in conflict resolution

During the operation phase of a given set of intents, the associated closed loop may find conflicting actions while trying to fulfil the intents, as described in clause 6.3.1. This indicates that the intent handler has to find solutions that may prioritize one intent over the other, which means, in this case, degrading the other conflicting intent. The degradation of the intents would only occur if the handler does not have the means to satisfy all intents at the same time. Then, it may prioritize one intent maximizing the intent handler's global utility (the intent handler global utility is the aggregation of all intent utilities scores into one). In any case, the degradation would be communicated to the intent owners to allow them to act accordingly. The utility is provided by an owner and processed by a handler. The use of the utility with intents is optional.

The intent owner may communicate its utility to the intent handler, by adding the utility description to its IME capability profile as defined in clause 6.5, or it may also add the utility information to the intent object sent to the intent handler during negotiation phase. This information helps the intent handler to interpret the provided utility information correctly and allows an intent handler to perform more informed decisions during intent fulfilment and resolve conflicts by selecting the strategy that will maximize its utility. It may also be necessary to the intent handler to perform normalization of the utilities in case the utilities are in different data units or scales (e.g. one utility ranging from 1 ms to 100 ms and another utility ranging from 1 000 kbps to 10 000 kbps). The following example shows the theoretical steps of using utility information in conflict resolution:

1) During the distribution phase, two intent owners (owner A and owner B) set intents (intent A and intent B) for an intent handler with specific requirements on KPIs, each requirement has an associated utility that indicates how well the requirement is fulfilled, as shown in Figure 6.3.2-1. A higher utility value of the respective solution implies that the intent is better fulfilled. The intent handler wants to maximize the handler's utility, that is, for example, internally calculated as a sum of all the intent's utility or by means of other functions. A desired feature is the ability to balance the requirements from different intents by using the handler's global utility as a decision threshold.

2) During the operation phase, the intent handler propose actions that may conflict with intents A and B.

3) The intent handler may prioritize the fulfilment of intent A given that if the KPI_A goes above x, the utility drops very quickly, whereas the utility drops only gradually when KPI_B goes above y. As a result, if there are no actions to keep the values below x and y, the intent handler may solve the conflict by allowing more degradation of intent B compared to intent A.

4) The intent handler will not provide any details of the conflicts and it will also not ask any explicit guidance to the intent owner (except for the cases where the JUDGE operation is applied) to solve the conflict, instead, it will only communicate to the intent owner B the degradation of its intent so the intent owner may take action accordingly.

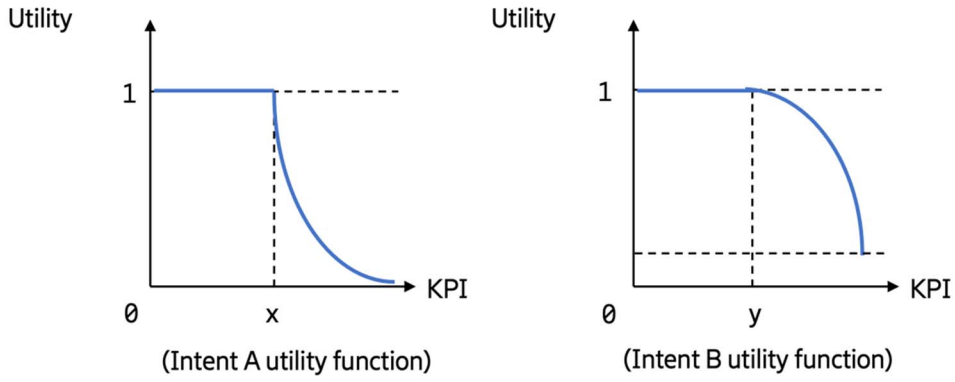

Figure 6.3.2-1: Two examples of intent utility functions for different intents

6.4 Intent management operations

6.4.1 Mandatory Operations

6.4.1.1 Introduction

The following operations shall be used to enable intent object instances lifecycle management and intent-driven Closed Loops.

The operations below are listed as mandatory and shall be supported by all intent management entities.

6.4.1.2 Create an intent

This operation shall be used for the creation of an intent object by an intent owner and to send this intent to an intent handler.

The create intent operation may result in the creation of the intent object at the intent handler, if the intent received is accepted, or it may result in the rejection of the new intent; in this case, the intent object is not created at the intent handler.

6.4.1.3 Read intent attributes

This operation shall be used for reading the intent related information by an intent owner.

This operation shall also be used by any authorized entity acting as an intent owner.

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.

6.4.1.4 Update an intent

This operation shall be used for the modification of an intent object by an authorized intent owner. The only intent management entity that is allowed to use this operation is the intent owner (the one that created the intent).

6.4.1.5 Delete an intent

This operation shall be used for the removal of an intent object by an authorized intent owner. That means, after the successful execution of this operation the intent will no longer exist.

The only authorized entity that is allowed to use this operation is the intent owner (the one that created the intent).

6.4.2 Optional Operations

6.4.2.1 Introduction

An important aspect of intent-driven closed loop-based systems is their support of automation of intent interactions. Such support is facilitated by the declarative and abstracted nature of intent information objects, in that intent owner is not required to know anything about the intent handler underlying system in order to formulate intents. However, further intent interface operations may be created to support the negotiation - and the potential automation of negotiation - of intent-based service terms between both intent owner and 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 intents for the intent owner to choose from. Or the intent owner may wish to discover about a 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 intent-driven closed loops system negotiation.

6.4.2.2 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. An example is an intent with two expectations: one expectation is latency < 10 ms, and the second one throughput > 1gbps. However, the intent handler only has actions that fulfil only one of them.

In both situations described above, while the intent handler has autonomy to decide the best approach to be taken, i.e. which action to utilize, it may be beneficial that the intent handler and intent owner engage in a collaborative evaluation so that the intent owner may improve the final intent result. The collaborative evaluation allows the intent handler to ask the intent owner to decide which out of many possible outcomes is preferred from intent owner perspective.

The intent owner sends the intent to the intent handler. After accepting the intent, the intent handler starts the Judge operation by asking the intent owner for the judgement on a set of results. In this procedure multiple intent reports are sent, each of them represents the expected result for an action.

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 action alternative, since the intent owner is not capable to understand these details. The intent handler would rather use intent reports to communicate the expected results of these actions. This means the intent owner can judge based on effect of the actions rather than how it was done.

The invocation of Judge operation initiated by the intent handler is followed by the indication of 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 intent handler 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.

The owner may use the identifier of each presented outcome report to send a list of them back to the intent handler. The list refers to reported results that would be further analysed and selected by the owner. The list is sorted with the most preferred outcome first. After receiving the preferred list, the intent handler takes the appropriated action based on the preferences and reports back the results to the intent owner.

6.4.2.3 Best Intent

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

After accepting the Best intent MnS capability, at the end of evaluation, the intent handler sends back a response to the Best MnS capability in an intent report format 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. After the intent handler reply, the intent owner can delete the Best intent, receive the confirmation and send another intent based on the evaluations provided by the intent handler. In case the outcomes provided by the intent handler is not the best option for the intent owner, it may send another Best intent changing the configuration.

It is also possible to ask for the best possible proposal for already accepted 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 performance that the intent handler can deliver.

6.4.2.4 Intent Feasibility Check

In specific 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 (ETSI GR ZSM 011 [i.1]).

This is an MnS capability that may 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 reports in case that intent would be fulfilled.

Also, any intent sent using this MnS capability is actively life cycle managed by the intent owner, e.g. it is also possible to UPDATE the feasibility check intent to explore different options (alternative intents with different expectations) to find out how the intent handler and its underlying system would react. The intent handler has to provide reports with the outcomes for those alternative intents.

The intent using this MnS capability could also be removed using the DELETE operation, as well as the intent created using the CREATE operation.

This MnS capability is part of the investigation phase, and it is initiated by the intent owner. Based on its needs, the intent owner formulates an intent to get an estimation about the expected outcomes in case that this intent would be used. The feasibility check intent is sent to an intent handler using the Intent Feasibility Check MnS capability. The intent handler would start sending reports representing its estimate of handling results if the intent would be created and handled.

The intent handler would continue sending reports until the probing intent is removed by the owner. It is also possible to modify the feasibility check intent in order to test multiple intent configurations (i.e. an intent with different expectations) or in cases where the expected outcomes do not meet (or partially meet) the intent owner requirements. In this case, a new negotiation of this feasibility check intent is required.

Ultimately the intent owner would proceed by removing the feasibility check intent. This means, based on the feasibility check results it can decide which intent configuration to use.

6.5 Intent Management Entity registry

The intent handlers in the management domains may need to be discovered by the intent owners before the intents are exchanged with the operations defined in clause 6.4.

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, e.g. on the supported intent models, on the supported intent interface operations, or any other decision criteria.

Every IME has a set of capabilities, which are formally expressed by an IME profile. An IME profile may have the information below:

- IME interface endpoint.
- Management domain.
- Supported intent roles.
- Supported intent optional operations (e.g. Judge).
- 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 (as defined in ETSI GS ZSM 002 [1]).

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.

The registration capability of the IME sends the capability profiles to the IME registry, publishing the existence of an IME along with its scope of responsibility and capabilities. The discovery capability of the IME permits searches in the publicly available capability profile of IMEs. An IME discovery is an essential step for intent owners deciding if there are intent handlers with the right scope of responsibilities capable of handle the owner's needs, and what an intent owner can express with this intent.

7 Additional Services and Capabilities

7.1 Management Services for Intents

7.1.1 Introduction

The present clause specifies ZSM management services related to intent management. The present clause contains the descriptions and the service definition tables of management services (Table 7.1.1-1) which are classified as intent management related services.

Table 7.1.1-1: Intent management service

Service name	Intent management service.
Service capabilities	
Manage intents (M)	Manage (create, read, update, delete) intent objects (as specified in clause 6.4) in the respective management domain(s).
Judge (O)	Request the intent owner to judge and rank (according to the intent owner preference) the solutions provided by an intent handler, as specified in clause 6.4.2.2.
Best (O)	Request the best results for a given intent expectation, as specified in clause 6.4.2.3.
Intent feasibility check (O)	Explore what intent expectations are feasible by the intent handler, as specified in clause 6.4.2.4.
Register IME profile (O)	This capability is used by the IME to register its profile, as specified in clause 6.5.
Discover IME profile (O)	This capability is used by the IME to request IME profiles, as specified in clause 6.5.

7.2 Intent-driven Closed Loop Governance Service

The present clause specifies additions to ZSM management service related to Closed Loop Governance (the target service is defined in ETSI GS ZSM 009-1 [2], clause 9.2.2) to support intent-driven operations. The extension of the CLG to support intent-based information manipulation service is described in Table 7.2-1.

Table 7.2-1: Service definition extension

Service name	Intent-driven Closed Loop Governance (IdCLG) service.
External visibility	OPTIONAL.
Service capabilities	
Manipulate intent information (M)	Convert the intent-related information to CL LCM operations.

7.3 Intent-driven Closed Loop Information Reporting Service

The IdCL information reporting service allows providing current or past information of one or more Intent-driven Closed Loops or Intent-driven Closed Loop instances. The extension of the Closed Loop information reporting service to support intent report format is described in Table 7.3-1.

Table 7.3-1: Service definition extension

Service name	Intent-driven Closed loop information reporting service.
External visibility	OPTIONAL.
Service capabilities	
Convert to intent report format (M)	Convert the CL status information to intent report format.

Annex A (informative): Intent LCM Aspects

A.1 Introduction

Intents are used for interactions between two distinct intent management entities. The lifecycle management of a given intent is performed by the intent owner. As described in clause 4.1.2 an IME in a specific MD may assume two distinct roles:

1) **intent handler**; and

2) **intent owner**.

The intent owner is the only entity allowed to create, modify or remove the intent.

The intent handler plays an active role in the lifecycle of intents mainly during the negotiation and formal definition of the intent after an intent is defined and created (intent fulfilment stage). An IME becomes the intent handler of an intent when it receives that particular intent from an intent owner. An intent handler has to consider the requirements, goals and constraints specified in the intent when operating the domain and infrastructure. It is responsibility of the intent handler to keep the intent 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 it for the best option.

An IME may assume one of the two roles (handler or owner) for each intent and the role will determine the responsibilities and tasks in the lifecycle management of the intents. The relationships between an intent owner and an intent handler is one-to-one, as shown in Figure A.1-1. There are never multiple intent owners for a given intent created at the intent handler, and an intent is never created at multiple intent handlers. Instead, an intent owner may generate multiple (and different) intents to address the requirements for each affected domain and send these to different intent handlers.

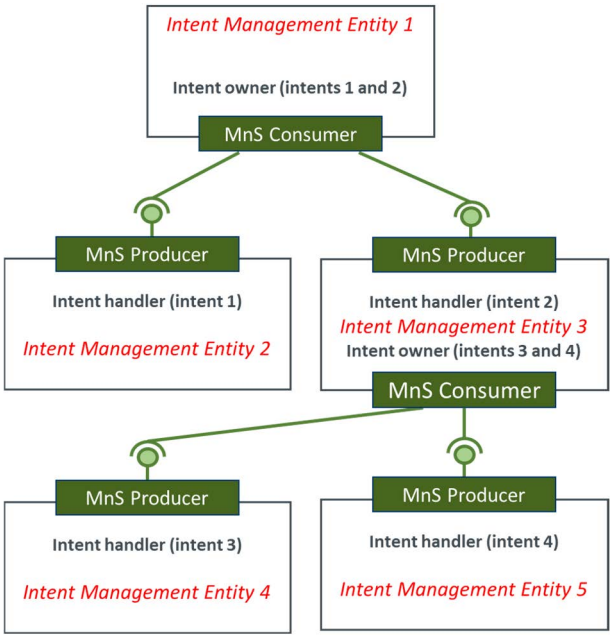

Figure A.1-1: Relationship between intent owner and intent handler

A.2 Phases of the intent lifecycle

A.2.1 Background

The intent LCM consists of different phases as shown in Figure A.2.1-1.

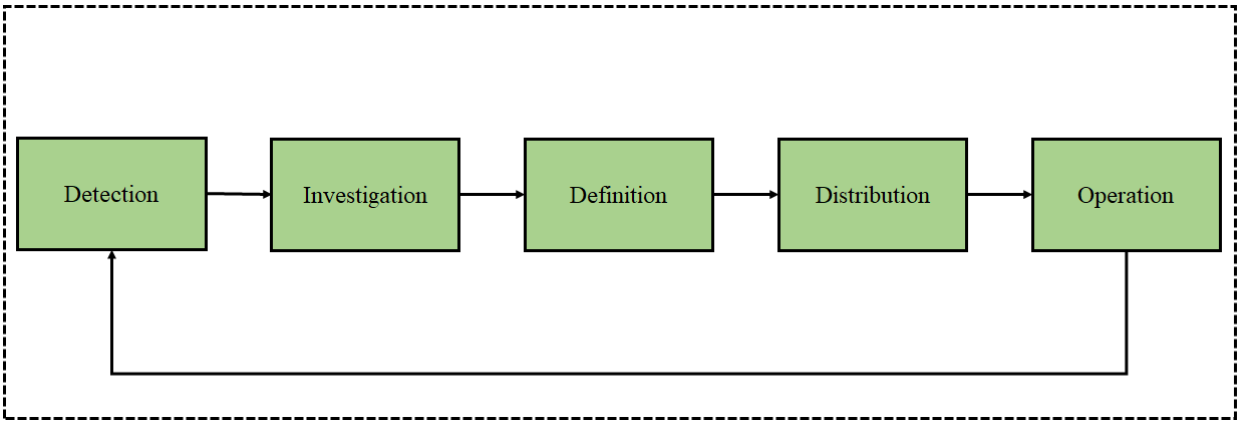

Figure A.2.1-1: Intent LCM phases

The lifecycle of the intent begins before the intents is accepted at the intent handler, given that the intent is originated at the intent owner and there are steps that need to be performed before creating the intent at the intent handler. The intent handler/consumer 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.

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 an intent handler. The intent reports coming from the intent handler may serve as information sources for the detection phase.

Investigation:

In the investigation phase an intent owner, in collaboration with the intent handler, determines if the intent is feasible.

This has two aspects: first, the intent owner needs to find the intent handler that have the right domain responsibilities and support the intent information 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 6.4.2). The intent owner can explore what the intent handler result of a wanted intent would be, what would be the best result the intent 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 intents are feasible and which intents 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 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 6.5.

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.

A.2.2 Intent LCM examples

A.2.1.1 Introduction

The following examples explore process flows covering some of the intent interface operations (mandatory and optional) through different intent LCM phases. The steps performed in these examples also correlate with the expected intent owner/handler behaviours defined in clause 5.2.2. Scenarios improving or mixing the examples below (e.g. scenarios where more than one intent handler is involved) are also possible. But, for simplicity, they are not considered. All following examples are only related to one intent handler being involved.

A.2.1.2 Intent LCM: Basic example 1

In this example, an intent owner is requesting a management service using an intent, which is to be fulfilled by the intent handler. In this scenario, the service is delivered without issues:

1) **Intent handler:** Shares its capabilities, allowing it to be discovered by any intent owner.
2) **Intent owner:** Identifies the need for an intent. [Detection phase].
3) **Intent owner:** Starts assessing available intent handlers and their capabilities. Then, select the handler for fulfilling the specific intent. [Investigation phase].
4) **Intent owner:** Formulates the intent. [Definition phase].
5) **Intent owner:** Sends the intent to the intent handler chosen using the CREATE operation. [Distribution phase].
6) **Intent handler:** Sends the first intent report accepting the intent to the intent owner. [Operation phase].
7) **Intent handler:** Starts the operations for fulfilment of the intent. [Operation phase].

8) **Intent handler:** Sets up the service that fulfils the intent. [Operation phase].

9) **Intent handler:** Sends a report on fulfilment status to the intent owner reporting it is being successfully fulfilled. [Operation phase].

10) **Intent owner:** Evaluates the intent handler report and check if there is the need to change or update the intent. [Detection phase]:

    a) At this point the intent owner could use the UPDATE operation to update the intent if needed.

11) **Intent owner:** Uses the DELETE operation since the intent is not needed anymore. [Distribution phase].

12) **Intent handler:** Sends the final fulfilment status report to the intent owner and confirms the intent's deletion. [Operation phase].

A.2.1.3 Intent LCM: Basic example 2

In this example, an intent owner is requesting a management service using an intent, which is to be fulfilled by the intent handler. In this scenario, the intent needs to be negotiated since there is something wrong with the intent (e.g. conflicting requirements):

1) **Intent handler:** Shares its capabilities, allowing it to be discovered by any intent owner.

2) **Intent owner:** Identifies the need for an intent. [Detection phase].

3) **Intent owner:** Starts assessing available intent handlers and their capabilities. Then, select the handler for fulfilling the specific intent. [Investigation phase].

4) **Intent owner:** Formulates the intent. [Definition phase].

5) **Intent owner:** Sends the intent to the intent handler chosen using the CREATE operation. [Distribution phase].

6) **Intent handler:** Sends the first intent report rejecting the intent and providing the rejection reason to the intent owner. [Operation phase].

7) **Intent owner:** Considers the reason for rejection and decides to change. [Detection, Investigation and Definition phase].

8) **Intent owner:** Sends the new (reformulated) intent to the intent handler using the CREATE operation. [Distribution phase].

9) **Intent handler:** Sends an intent report accepting the new intent to the intent owner. [Operation phase].

10) **Intent handler:** Starts the operations for fulfilment of the intent. [Operation phase].

11) **Intent handler:** Sets up the service that fulfils the intent. [Operation phase].

12) **Intent handler:** Sends a report on fulfilment status to the intent owner reporting it is being successfully fulfilled. [Operation phase].

13) **Intent owner:** Evaluates the intent handler report and check if there is the need to change or update the intent. [Detection phase]:

    a) At this point the intent owner could use the UPDATE operation to update the intent if needed.

14) **Intent owner:** Uses the DELETE operation since the intent is not needed anymore. [Distribution phase].

15) **Intent handler:** Sends the final fulfilment status report to the intent owner and confirms the intent's deletion. [Operation phase].

A.2.1.4 Intent LCM: Intent handler not being able to fulfil an intent with strict requirements

In this example, an intent owner is requesting a management service using an intent with strict requirements, which is to be fulfilled by the intent handler. In this scenario, a change in the network occurs affecting the service and the intent handler is not able to fulfil the intent anymore, then the fulfilment process finishes. The meaning of an intent with strict requirements is that all requirements has to be fulfilled. This way, they are not able to be negotiated. If the intent handler is not able to fulfil any of the given requirements, the intent is deleted:

1) **Intent handler:** Shares its capabilities, allowing it to be discovered by any intent owner.
2) **Intent owner:** Identifies the need for an intent. [Detection phase].
3) **Intent owner:** Starts assessing available intent handlers and their capabilities. Then, select the handler for fulfilling the specific intent. [Investigation phase].
4) **Intent owner:** Formulates the intent with strict requirements. [Definition phase].
5) **Intent owner:** Sends the intent to the intent handler chosen using the CREATE operation. [Distribution phase].
6) **Intent handler:** Sends the first intent report accepting the intent to the intent owner. [Operation phase].
7) **Intent handler:** Starts the operations for fulfilment of the intent. [Operation phase].
8) **Intent handler:** Sets up the service that fulfils the intent. [Operation phase].
9) **Intent handler:** Sends a report on fulfilment status to the intent owner reporting it is being successfully fulfilled. [Operation phase].
10) **Intent owner:** Evaluates the intent handler report. [Detection phase].
11) **Network:** Changes happen, causing intent degradation.
12) **Intent handler:** Sends a report on fulfilment status informing the intent degradation to the intent owner. [Operation phase].
13) **Intent owner:** Uses the DELETE operation since the intent since it is not meeting the intent owner expectation anymore. [Distribution phase].
14) **Intent handler:** Sends the final fulfilment status report to the intent owner and confirms the intent's deletion. [Operation phase].

A.2.1.5 Intent LCM: Intent handler using JUDGE operation

In this example, an intent owner is requesting a management service using an intent, which is to be fulfilled by the intent handler. In this scenario, the intent handler found multiple solutions for fulfilling the intent. The intent handler cannot further decide what is the best solution, then ask the intent owner for guidance:

1) **Intent handler:** Shares its capabilities, allowing it to be discovered by any intent owner.
2) **Intent owner:** Identifies the need for an intent. [Detection phase].
3) **Intent owner:** Starts assessing available intent handlers and their capabilities. Then, select the handler for fulfilling the specific intent. [Investigation phase].
4) **Intent owner:** Formulates the intent. [Definition phase].
5) **Intent owner:** Sends the intent to the intent handler chosen using the CREATE operation. [Distribution phase].
6) **Intent handler:** Sends the first intent report accepting the intent to the intent owner. [Operation phase].
7) **Intent handler:** Sets up a closed loop based on the intent requirements to fulfil the intent . [Operation phase].

8) **Intent handler:** Uses JUDGE operation since multiple solutions were found. Then, send these alternatives (only the action's result, not the action's details) to the intent owner for consideration. [Operation phase].

9) **Intent owner:** Evaluates, decides, and sends the preferred alternatives to the intent handler. [Detection phase].

10) **Intent handler:** Sets up the service that fulfils the intent based on the intent owner preference. [Operation phase].

11) **Intent handler:** Sends a report on fulfilment status to the intent owner reporting it is being successfully fulfilled. [Operation phase].

12) **Intent owner:** Evaluates the intent handler report and check if there is the need to change or update the intent. [Detection phase]:

    a) At this point the intent owner could use the UPDATE operation to update the intent if needed.

13) **Intent owner:** Uses the DELETE operation since the intent is not needed anymore. [Distribution phase].

14) **Intent handler:** Sends the final fulfilment status report to the intent owner and confirms the intent's deletion. [Operation phase].

A.2.1.6 Intent LCM: Intent owner using Best Intent operation

In this example, an intent owner is requesting a management service using an intent, which is to be fulfilled by the intent handler. In this scenario, the intent owner wants to know what requirements it can be used in an intent and still have an intent handler fulfilling it:

1) **Intent handler:** Shares its capabilities, allowing it to be discovered by any intent owner.

2) **Intent owner:** Identifies the need for an intent. [Detection phase].

3) **Intent owner:** Starts assessing available intent handlers and their capabilities. Then, select the handler for fulfilling the specific intent. [Investigation phase].

4) **Intent owner:** Formulates an intent highlighting the requirements it would like to know the best value the intent handler can deliver. [Investigation phase].

5) **Intent owner:** Sends the intent to the intent handler chosen using the BEST Intent operation. [Investigation phase].

6) **Intent handler:** Sends the first intent report accepting the BEST intent to the intent owner. [Investigation phase].

7) **Intent handler:** Evaluate different options to achieve the best values based on the intent requirements. [Investigation phase].

8) **Intent handler:** Sends a report containing the best values for the requirements it can achieve to the intent owner. [Investigation phase].

9) **Intent owner:** Evaluates the intent handler report . [Investigation phase].

10) **Intent owner:** Uses the DELETE operation to stop the investigation. [Investigation phase].

11) **Intent handler:** Sends the final report to the intent owner to confirms the investigation is finished. [Operation phase].

12) **Intent owner:** Formulates the intent based on the investigation performed. [Definition phase].

13) **Intent owner:** Sends the intent to the intent handler chosen using the CREATE operation. [Distribution phase].

14) **Intent handler:** Sends the first intent report accepting the intent to the intent owner. [Operation phase].

15) **Intent handler:** Starts the operations for fulfilment of the intent. [Operation phase].

16) **Intent handler:** Sets up the service that fulfils the intent. [Operation phase].

17) **Intent handler:** Sends a report on fulfilment status to the intent owner reporting it is being successfully fulfilled. [Operation phase].

18) **Intent owner:** Evaluates the intent handler report and check if there is the need to change or update the intent. [Detection phase]:

    a) At this point the intent owner could use the UPDATE operation to update the intent if needed.

19) **Intent owner:** Uses the DELETE operation since the intent is not needed anymore. [Distribution phase].

20) **Intent handler:** Sends the final fulfilment status report to the intent owner and confirms the intent's deletion. [Operation phase].

A.2.1.7 Intent LCM: Intent owner using Intent Feasibility Check operation

In this example, an intent owner is requesting a management service using an intent, which is to be fulfilled by the intent handler. In this scenario, the intent owner wants to explore what intent requirements and constraints are possible to be fulfilled by the intent handlers:

1) **Intent handler:** Shares its capabilities, allowing it to be discovered by any intent owner.

2) **Intent owner:** Identifies the need for an intent. [Detection phase].

3) **Intent owner:** Starts assessing available intent handlers and their capabilities. Then, select the handler for fulfilling the specific intent. [Investigation phase].

4) **Intent owner:** Formulates an intent to check its expected result if it would be considered by an intent handler. [Investigation phase].

5) **Intent owner:** Sends the feasibility check intent to the intent handler chosen using the Intent Feasibility Check operation. [Investigation phase].

6) **Intent handler:** Sends the first intent report accepting the feasibility check intent to the intent owner. [Investigation phase].

7) **Intent handler:** Evaluate the feasibility check intent. [Investigation phase].

8) **Intent handler:** Sends a report containing the expected results if the intent would be set for fulfilment to the intent owner. [Investigation phase].

9) **Intent owner:** Evaluates the intent handler report . [Investigation phase].

10) **Intent owner:** Uses the DELETE operation to stop the investigation. [Investigation phase].

11) **Intent handler:** Sends the final report to the intent owner to confirms the investigation is finished. [Operation phase].

12) **Intent owner:** Formulates the intent based on the investigation performed. [Definition phase].

13) **Intent owner:** Sends the intent to the intent handler chosen using the CREATE operation. [Distribution phase].

14) **Intent handler:** Sends the first intent report accepting the intent to the intent owner. [Operation phase].

15) **Intent handler:** Starts the operations for fulfilment of the intent. [Operation phase].

16) **Intent handler:** Sets up the service that fulfils the intent. [Operation phase].

17) **Intent handler:** Sends a report on fulfilment status to the intent owner reporting it is being successfully fulfilled. [Operation phase].

18) **Intent owner:** Evaluates the intent handler report and check if there is the need to change or update the intent. [Detection phase]:

    a) At this point the intent owner could use the UPDATE operation to update the intent if needed.

19) **Intent owner:** Uses the DELETE operation since the intent is not needed anymore. [Distribution phase].

20) **Intent handler:** Sends the final fulfilment status report to the intent owner and confirms the intent's deletion. [Operation phase].

History

Document history		
V1.1.1	October 2024	Publication