ETSI GS ZSM 009-1 V1.1.1 (2021-06)

Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers

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-009-1_Cla_enab

Keywords
automation, management, network, service

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).

The present document is part 1 of a multi-part deliverable covering Closed-Loop Automation (CLA) based on the ZSM architectural framework, as identified below:

Part 1: "Enablers";

Part 2: "Solutions for automation of E2E service and network management use cases";

Part 3: "Advanced topics".

Full details of the entire series can be found in clause 4.

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 describes enablers for Closed-Loop Automation (CLA) based on the ZSM architectural framework. The present document initially specifies Closed Loop-specific requirements and introduces Closed Loops within the ZSM framework, from both functional and deployment perspectives. The specifications include enablers for Closed Loop Governance (CLG), covering the definitions of Closed Loop lifecycle management as well as Closed Loop models. Such enablers allow automatic deployment and configuration of Closed Loops involving both the end-to-end service management domain and the individual management domains. Then, the present document specifies enablers for Closed Loops coordination within the ZSM framework, including means for delegation and escalation between Closed Loops as well as other coordination means. Finally, the deliverable specifies stage-2 generic enablers for Closed-Loop Automation (CLA) and includes new management services in addition to those identified in ETSI GS ZSM 002 [2]. Closed loops running entirely within the managed entities are out-of-scope.

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 001: "Zero-touch network and Service Management (ZSM); Requirements based on documented scenarios".

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

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

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 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".

[i.2] ETSI GR ZSM 009-3: "Zero-touch network and Service Management (ZSM); Closed-loop automation; Part 3: Advanced topics".

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 [3] and the following apply:

delegation: action taken by a ZSM entity to assign responsibility for one or more goals to one or more other ZSM entities within stated limits

NOTE 1: Delegation defines the operation autonomy the receiving ZSM entity may use to achieve the assigned goal(s) and defines conditions for escalation (i.e. to address situations leading to breach in operation autonomy).

NOTE 2: Delegation is often combined with the concept of escalation in the context of operation autonomy. Operation autonomy is a central concept in Closed-Loop Automation (see clause 7 of ETSI GS ZSM 002 [2]).

NOTE 3: Delegation can define the escalation entity or entities.

NOTE 4: Delegation relies on acknowledgment by the receiving entity, and may contain a phase of negotiation between the entity that originates the delegation and the entity/entities that receive the delegation (e.g. if the receiving entity can achieve, entirely, partly, or not the delegated goal).

escalation: action taken by a ZSM entity to inform one or more ZSM entities about a breach in its operation autonomy

NOTE 1: A ZSM entity escalates an issue that prevents it from achieving its goal properly, and that is not solvable with means under its control. An escalation differs from other types of issue or problem reporting in the sense that an escalation signals a breach in operation autonomy.

NOTE 2: Escalation is often combined with the concept of delegation in the context of operation autonomy. Operation autonomy is a central concept in Closed-Loop Automation (see clause 7 of ETSI GS ZSM 002 [2]).

NOTE 3: Escalation can target explicitly an entity or list of entities. Escalation can be with or without acknowledgment by the receiving entity or entities.

NOTE 4: Escalation can contain contextual information about the problematic situation faced, attempt(s) to solve it, and other contextual information that could be useful to the recipient of the escalation, or any other information defined by the corresponding policies.

managed entity: managed resource, a managed service or a managed Closed Loop

NOTE: This term differs from the definition in ETSI GS ZSM 007 [3] because a Closed Loop may also be managed, similarly to managed resources (e.g. VNFs, PNFs) and managed services (e.g. cloud services, RFSs, CFSs).

3.2 Symbols

For the purposes of the present document, the symbols given in ETSI GS ZSM 007 [3] apply.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS ZSM 007 [3] and the following apply:

AI	Artificial Intelligence
CL	Closed Loop
CLA	Closed-Loop Automation
CLC	Closed Loop Coordination
CLG	Closed Loop Governance
E2E	End-to-End

E2ES	End-to-End Service
KPI	Key Performance Indicator
LCM	Lifecycle Management
M2O-CL	Made-to-Order Closed Loop
MD	Management Domain
ML	Machine Learning
MnF	Management Function
MnS	Management Service
RDBMS	Relational DataBase Management System
RM-CL	Ready-Made Closed Loop
ZSM	Zero-touch network and Service Management

4 Description of ZSM 009 multi-part deliverable

The present document specifies part 1 of a multi-part deliverable that focuses on Closed-Loop Automation (CLA) based on the ZSM framework architecture.

The present document specifies the enablers for Closed-Loop Automation (CLA) that can be used in different use cases. It includes aspects of Closed Loop Governance (CLG) in clause 8.1 and Closed Loop Coordination (CLC) in clause 8.2. The present document also extends the ZSM framework architecture specified in ETSI GS ZSM 002 [2] with new management services and capabilities needed for the specified Closed Loop enablers (clause 9).

ETSI GS ZSM 009-2 [i.1] specifies solutions for end-to-end service and network automation use cases, based primarily on the enablers and architectural elements specified in ETSI GS ZSM 009-1 (the present document).

ETSI GR ZSM 009-3 [i.2] investigates advanced topics related to Closed-Loop Automation (CLA) such as learning and cognitive capabilities. It documents problem statements and technical challenges, derives potential requirements and provides recommendations for the evolution of Closed-Loop Automation (CLA) standardization activities.

5 Requirements for Closed-Loop Automation

5.1 Introduction

This clause defines the requirements relevant to Closed-Loop Automation (CLA) within the ZSM framework architecture. The requirements are derived from ETSI GS ZSM 001 [1] and ETSI GS ZSM 002 [2], as well as new requirements introduced specifically for Closed-Loop Automation (CLA).

The requirements are considered for the specification of the Closed-Loop Automation (CLA) enablers in clause 8.

5.2 General requirements

[CL-general-1] Closed loops within the ZSM framework shall have access to the data necessary to allow its proper operation.

NOTE 1: Examples of data include system log data, historical data, policy data, other external data input, etc.

[CL-general-2] The ZSM framework shall allow establishing different types of relationships between Closed Loops.

NOTE 2: Examples of relationships are peer Closed Loops, hierarchical or nested Closed Loops.

[CL-general-3] Closed loops stages should be able to expose their outcomes to the authorized entities.

NOTE 3: Examples of authorized entities are other Closed Loops and ZSM framework consumers.

[CL-general-4] Authorized Closed Loops within the ZSM framework shall be able to interact with other Closed Loops within the ZSM framework in different management domains.

[CL-general-5] Closed loops within the ZSM framework shall be able to be defined across multiple management domains.

NOTE 4: The management services for different Closed Loop stages may be provided by different management domains, e.g. data can be collected from one management domain and execution of actions may be performed in another management domain, etc.

[CL-general-6] Closed loops within the ZSM framework working in a management domain shall be able to request other Closed Loops in the E2E service management domain to take necessary action(s) in different management domains.

[CL-general-7] Closed loops stages should support influences from authorized entities outside the Closed Loop, but within the ZSM framework.

[CL-general-8] The actions taken by Closed Loops shall be logged.

[CL-general-9] The period for which Closed Loops logs are retained should be configurable.

[CL-general-10] The ZSM framework should support capabilities to present external instructions to the Closed Loops in a declarative form.

[CL-general-11] Closed loops within the ZSM framework should understand external instructions received in a declarative form.

[CL-general-12] Closed loops within the ZSM framework should support the capability to provide the state of each stage for its proper operation.

[CL-general-13] Closed loops within the ZSM framework should support the capability to detect abnormal states of any stage and notify the authorized entities.

[CL-general-14] Decisions based on AI/ML made by a Closed Loop within the ZSM framework should be monitored to support administrative audit trails.

[CL-general-15] Closed loops within the ZSM framework should support data collection from multiple available and applicable data sources, including their several data schemas.

[CL-general-16] The ZSM framework should support capabilities to avoid conflicts between Closed Loops.

[CL-general-17] The ZSM framework should support capabilities to minimize the negative effects of conflicts between Closed Loops.

[CL-general-18] Closed loops within the ZSM framework should support capabilities to enable continuous data integration from several available and applicable data sources.

[CL-general-19] The ZSM framework should support capabilities to enable continuous data integration from management data sources in a Closed Loop within the ZSM framework.

[CL-general-20] The ZSM framework should support capabilities to enable real-time data integration from management data sources in a Closed Loop within the ZSM framework.

[CL-general-21] The ZSM framework shall support a capability to uniquely identify a Closed Loop instance.

[CL-general-22] The ZSM framework should support capabilities to control the execution of a Closed Loop during its run time.

NOTE 5: An example of control capabilities could be the use of pause points.

[CL-general-23] Closed loops within the ZSM framework should support capabilities to handle control requests from authorized external entities.

[CL-general-24] Closed loops within the ZSM framework should support capabilities to dynamically adapt the exposure of control capabilities to external entities.

NOTE 6: Closed loop control capabilities could be defined by the Closed Loop model and may vary depending on the phases of the defined Closed Loop life cycle.

6 Introduction to Closed-Loop Automation

A Closed Loop (CL) is a type of control mechanism that monitors and regulates a set of managed entities with the objective of achieving a specific goal. CLs can be logically decomposed into a variable number of stages, each of them responsible for performing part(s) of the functionality of the Closed Loop. Well-known Closed Loop types are OODA loop, composed of 4 stages (Observe, Orient, Decide, Act) and MAPE-K, also composed of 4 stages (Monitor, Analyse, Plan, Execute) plus Knowledge.

Closed-Loop Automation (CLA) is the combination of CL stages that create automated processes that based on feedback from monitoring data can manage the network reducing or removing human involvement from the operation of a system. CLA in management systems can be realized with the combination and chaining of management services (data, analytics, policy, orchestration, etc.), and it creates autonomous systems that are able to constantly monitor and assess the network and take corrective actions when the goals (e.g. business intents, SLSs, etc.) are not fulfilled.

Although the purpose of CLA is to reduce the direct human intervention, it is important that any autonomous system still allows interactions with human operators. Such interactions can be used for the specification and modification of the goals of the CL, as well as for monitoring the performance of the autonomous system and eventual approval/rejection of actions taken by it.

The focus of the present document is to enable the creation and execution of CLs as well as on the integration and interoperability between CLs within the scope of ZSM framework, including CLs running at different domains. The interactions between CLs and external entities such as human operators are also considered, as this is important for the gradual increase of autonomy of Closed Loops.

NOTE: There are other CLs running outside of the scope of the ZSM framework (e.g. at the network resources) that can influence the CLA within the ZSM framework.

7 Closed loops within the ZSM framework

7.1 Introduction

ETSI GS ZSM 002 [2] specifies the overall architecture of the ZSM framework. CLs may exist in any of the management domains of the ZSM architecture as shown in Figure 7.1-1.

The present document specifies the management services relating to how CLs in the (E2E) MDs can be integrated into the ZSM framework. Two main groups of management capabilities are identified: capabilities relating to:

i) Closed Loop Governance (CLG); and

ii) Closed Loop Coordination (CLC).

Management capabilities belonging to these groups are specified in clause 9. Furthermore, the information models relating to a CL are presented in clause 8.1.4.

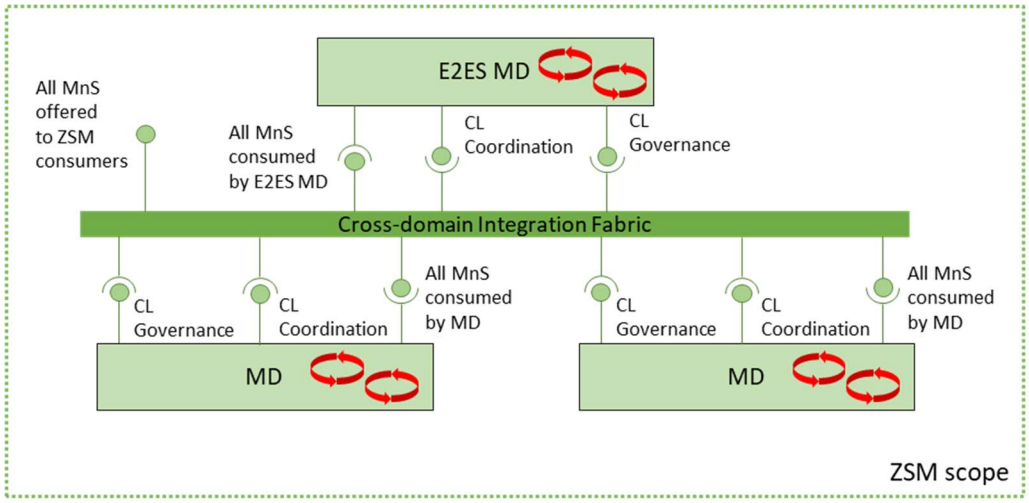

Figure 7.1-1: CL related management capabilities introduced in the present document

In the ZSM framework, CLs are realized by the interworking of management services defined in ETSI GS ZSM 002 [2] (and other future specifications in ISG ZSM) that are relevant to achieving the functionalities of the different stages of the CL. The stages when connected in a CL can be used to autonomously collect data, make decisions and execute actions upon one or more managed entities.

The set of stages that compose a CL includes at least one stage, called "Monitoring", responsible for collection of data from one or several sources (e.g. one or more managed entities, or external data sources) and one stage, called "Execution", responsible for executing actions upon one or more managed entities. The managed entities over which the execution is made are not necessarily the same as those where the data came from. Besides the two elementary stages, i.e. monitoring and execution, there may be other stages responsible for the analysis of operational and historical data and decisions based on the outcomes of the analysis. The number and functionality of intermediate stages between the "monitoring" of data and the "execution" can vary depending on the implementation and the deployment choices. This specification does not mandate a fixed number of stages that compose a CL within the ZSM framework, but it recommends at least three: one for monitoring, one for execution and one for analysis and decision making. Analysis and decision making can further be composed of several stages.

The CLs running within the ZSM framework can be represented by a functional view (clause 7.2) or by a deployment view (clause 7.3). The functional view expresses the functions that are conveyed in each stage and the flow of data and control between different stages of the Closed Loop and between the Closed Loop and external entities (clause 7.2.2). The deployment view expresses the connections between ZSM architectural components that are necessary for the realization of Closed Loops within the ZSM framework.

7.2 Functional view

7.2.1 Introduction

This specification takes as a baseline for the functional view the CL from Annex C of ETSI GS ZSM 002 [2] that is composed by 4 stages plus knowledge as its components, as shown in Figure 7.2.1-1. This collection and ordering of the components within a CL is referred to as the chain that forms the Closed Loop.

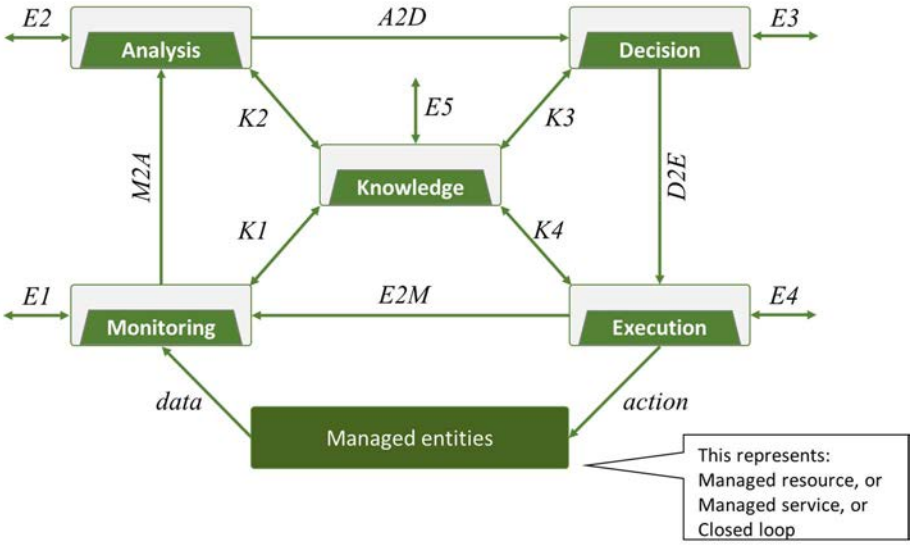

Figure 7.2.1-1: Functional view of a Closed Loop and its stages within the ZSM framework

In Figure 7.2.1-1, the "Managed entities" is not a stage in the CL, but rather the target of automation. A managed entity can be a managed service, a managed resource or another Closed Loop.

The "Monitoring" stage is responsible for collecting and pre-processing data from managed entities or from external sources.

NOTE 1: Data ingestion is a part of this stage. It is a process where data is transferred from one or more sources to a destination where it can be stored and further analysed. The data might be in different formats and come from various sources, including RDBMS, other types of databases, or from data streams. Since there is data of different origin, there is need for each source to be transformed in a way that allows it to be analysed in conjunction with data from other sources.

The "Analysis" stage is responsible for deriving insights from available data from the monitoring stage as well as historical data.

NOTE 2: An insight is produced based on data and the context of the finding. An example of insight may be the conclusion that congestion has taken place in a set of resources, and the context could be the time and date, the service affected, users involved and set of resources that make up the service. Insights can answer questions such as "What happened?" as well as "Why did it happen?". The insight derivation is a continuous process that can be enhanced by new data. Analysis should be able to continuously improve its results and, consequently, provide better decision options to the decision stage.

The "Decision" stage is responsible for deriving workflows from insights provided by the analysis stage.

NOTE 3: The decision stage governs the behaviour of the system as it decides which actions should be taken in face of issues detected in the analysis stage. The actions can be either reactive, proactive or predictive. The decision stage should decide which actions are required, but not necessarily how they should be taken in the managed entities. The translation from actions to commands is a responsibility of the execution stage.

In a CL the analysis and decision stages can optionally be combined into one single stage, kept separate as two stages, or even further split into multiple stages. The general objective of the stages that sit in between the monitoring and execution stages is to translate data into knowledge and by means of any type of intelligence mechanism (e.g. ML/AI, rules, policies) derive decisions that move the management target towards the desired state.

The "Execution" stage is responsible for executing workflows towards managed entities within the ZSM framework. Execution occurs when the decision stage determines that an action is required. Workflows may be composed of one or more operations that need to be properly orchestrated.

NOTE 4: Execution towards managed services or managed resources occurs when the CL involves one or more management domains and have direct access to steer the state of the managed services or managed resources by means of ZSM domain control services. Execution towards other CLs occurs when the goal of the CL that originates the execution is to steer the state of another CL. The latter case allows interactions between different CLs and enables the interworking of multiple distributed CLs that are required for the automation of the E2E services management.

The "Knowledge" is technically not a stage in the CL, but rather a means for storing and retrieving data that is shared between the stages within a Closed Loop, as well as between different Closed Loops. Examples of data are configuration data, operational and historical data. Knowledge can be used as a means for feedback signalling between the other stages.

Data and control flow are represented by the arrows between the stages of the Closed Loop (i.e. M2A, A2D, D2E, E2M). The types of data are information (M2A), insights (A2D), decisions (D2E) and feedback (E2M).

NOTE 5: The feedback in E2M may be used to improve responsiveness of the CL and when the CL does not use persistent data services. However, feedback is also possible to be conveyed by means of the knowledge, not only between execution and monitoring stages, but also between any stages of the CL.

The double-headed arrows between each stage and the knowledge (i.e. K1, K2, K3 and K4) are used for data-related inputs and outputs from CL stages.

The double-headed arrows pointing into each stage and the knowledge (i.e. E1, E2, E3, E4, E5) are used for data and control flows (e.g. policy/intent specification, parameter tuning, etc.) between different CLs within the ZSM framework, or between CLs and external entities. Interactions specified in clause 8.2 are conveyed through E1, E2, E3, E4 or E5.

All arrows in Figure 7.2.1-1 are realized by the endpoints exposed by the ZSM management functions that are part of the CL, as described in clause 7.3.

Clause 7.2.2 specifies examples of data and control flows that can exist between the CL stages.

7.2.2 Closed loops data and control flow

7.2.2.1 Introduction

The arrows in Figure 7.2.1-1 represent flow of data and control messages between the various components of the CL chain. There can be multiple flows running concurrently in a CL chain.

Which flows exist in a CL chain depends on the scenario and implementation choice. The typical, non-exhaustive, list of flows is detailed below.

7.2.2.2 Primary flow

Upon receiving data from the underlying managed entities and/or context sources, the primary flow can be executed. It involves all 4 Closed Loop stages and generates actions towards the managed entities. The transitions between stages in the primary flow are expressed by arrows M2A, A2D, D2E and E2M, as explained below.

M2A

- Monitoring stage provides information based on historical and/or streaming real-time data coming from various data sources. Information is a set of data processed in a meaningful way following the goals assigned to the Closed Loop. The information derived from raw data is highly depend on the context. Monitoring stage also provides capabilities for tuning the data sources and data ingestion.
    - Provide historical information
    - Provide real-time information
    - Tune data sources
    - Tune data ingestion

A2D

- Analysis stage provides insights on historical and/or streaming real-time information that is provided by the monitoring stage. Analysis also provides capabilities for tuning the analytics models and starting/terminating the analytics processes.
    - Provide historical analytics insights
    - Provide real-time analytics insights
    - Tune analytics models
    - Start/stop analytics process

D2E

- Decision stage provides action plans in form of workflows, e.g. onboarding of services and/or resources, or configuration changes.
    - Provide workflows
    - Tune decision models
    - Start/stop decision process

E2M

- Execution stage executes the workflows towards the managed entities. Optionally, it also provides information about historical and/or more recent actions that have been executed. This can be used as a feedback to the monitoring stage.
    - Provide real-time actions
    - Provide historical actions

7.2.2.3 Knowledge-enabled flow

The primary flow can preferably be augmented by data that are stored and retrieved from the knowledge. These data can be real-time (continuously generated by the operation of all CL stages) or historical (generated over time by the CL or coming from external sources).

The double-headed arrows K1, K2, K3 and K4 represent data that are stored to and retrieved from the knowledge. Any CL stage can retrieve any type of data to which it has authorized access, following the principle of the ZSM framework architecture. Each Closed Loop stage is responsible for generating and storing specific types of data, as described below:

K1

- Store historical information
- Store real-time information

K2

- Store historical analytics insights
- Store real-time analytics insights

K3

- Store historical workflows
- Store real-time workflows

K4

- Store historical actions
- Store real-time actions

7.2.2.4 Customization flow

Complementary to the primary and knowledge-enabled flows, there can be external data and control inputs and outputs that change the regular self-contained execution of the CLs. The customization flow is used for interactions between different CLs and between CLs and external entities, e.g. human operator, other management systems.

The customization flow may be used to configure the CL, monitor its continuous operations, approve/reject recommendations, explain the CL operation and decisions, as well as for interactions between multiple CLs, as specified in clauses 8.1 and 8.2.

The double-headed arrows E1, E2, E3 and E4 represent data and control inputs and outputs from/to other Closed Loops and external entities as described below:

- Start/stop process
- Change the settings of the stage
- Retrieve the current status of the stage
- Retrieve the historical data and/or real-time data of the stage
- Provide the resulting data of the stage to other Closed Loops
- Provide data to authorized entities outside of the ZSM framework

NOTE 1: "Process" in *Start/stop process* may refer to, e.g. training AI model, decision processes, etc.

NOTE 2: "Settings" in *Change settings of the stage* may refer to, e.g. attributes of CL models, configurations that define how each CL stage works, etc.

NOTE 3: "Historical and/or real-time data" in *Retrieve the historical and/or real-time data of the stage* may refer to, e.g. logs, outcomes derived in each CL stage, etc.

NOTE 4: "Authorized entities outside of the ZSM framework" in *Provide data to authorized entities outside of the ZSM framework* may refer to, e.g. external management system(s).

The double-headed arrow E5 represents data and control inputs and outputs from/to knowledge and external entities as described below:

- Retrieve the historical data and/or real-time data
- Provide data to authorized entities outside of the ZSM framework

NOTE 5: "Historical and/or real-time data" in *Retrieve the historical and/or real-time data* may refer to e.g. logs and outcomes provided by each stage and stored previously in the knowledge base.

7.3 Deployment view

Figure 7.3-1 depicts a CL instance deployed within a management domain or within the E2E service management domain. The CL instance is composed of stages, which are realized by one or more management functions, producing and/or consuming management services as defined in ETSI GS ZSM 002 [2], or in the present document.

As described in clause 6.3 of ETSI GS ZSM 002 [2], the use of the domain integration fabric for the invocation of services is recommended but not mandatory. The stages within a CL may communicate directly with each other, or indirectly via the domain integration fabric.

CL instances spanning multiple management domains may involve multiple domain integration fabrics (not shown in Figure 7.3-1) and the cross-domain integration fabric. According to ETSI GS ZSM 002 [2] specification, management services with optional or conditional external visibility may not expose their capabilities to consumers located outside the management domain where the management service is produced. Therefore, CLs that span multiple domains shall guarantee the external exposure of management services that need to be consumed across management domains. The exposure is implementation-dependent, and it is controlled by the Management capability exposure configuration.

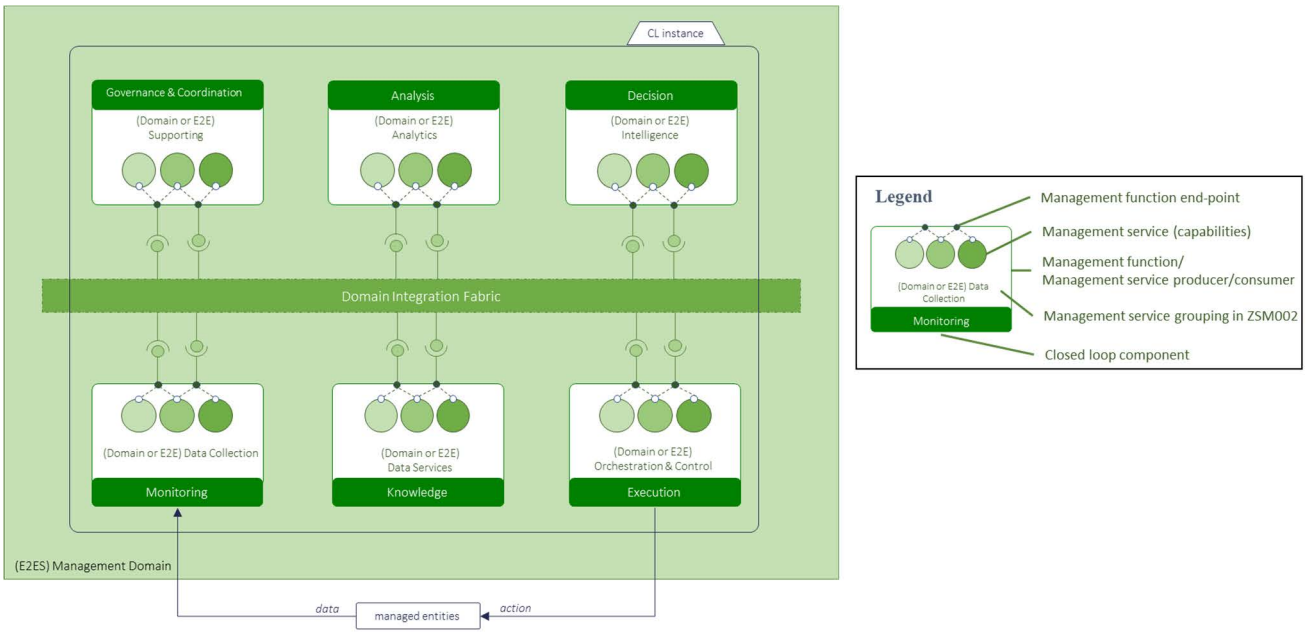

Figure 7.3-1: Deployment view of a Closed Loop instance and its stages within the ZSM framework considering a single management domain

Figure 7.3-1 depicts an example mapping of the CL stages and functionalities (Closed Loop components) to management functions based on the management services as defined the ETSI GS ZSM 002 [2].

The monitoring stage is realized, fully or in part, by the (domain or E2E) data collection management services (clauses 6.5.2 and 6.6.2 of ETSI GS ZSM 002 [2]).

The analysis stage is realized, fully or in part, by the (domain or E2E) analytics management services (clauses 6.5.3 and 6.6.3 of ETSI GS ZSM 002 [2]).

The decision stage is realized, fully or in part, by the (domain or E2E) intelligence management services (clauses 6.5.4 and 6.6.4 of ETSI GS ZSM 002 [2]).

The execution stage is realized, fully or in part, by the domain orchestration and control management services (clauses 6.5.5 and 6.5.6 of ETSI GS ZSM 002 [2]), when the CL is deployed within a management domain. When the Closed Loop is deployed within the E2E service management domain, the execution stage is realized, fully or in part, by the E2E orchestration management services only (clause 6.6.5 of ETSI GS ZSM 002 [2]).

NOTE: The E2E management domain does not interact directly with managed resources and thus does not specify Control management services.

Knowledge is realized, fully or in part, by the (domain or cross-domain) data services (clause 6.4 of ETSI GS ZSM 002 [2]). Additional management services may be used or defined to realize the functionality of the knowledge, such as management services for knowledge representation, knowledge management and knowledge reasoning. Typically, analytics management services may be used to create or derive new knowledge.

The governance and the coordination functionalities of the Closed Loop are realized, fully or in part, by the supporting management services (clauses 6.5.7 and 6.6.6 of ETSI GS ZSM 002 [2]). Additional management services may be used to realize the governance and coordination functionalities of the CL, as described in clauses 8.1 and 8.2 and clauses 9.2 and 9.3.

The communication and interoperation between the CL stages may be realized, fully or in part, by the (domain or cross-domain) integration fabric management services.

The grouping of management services in management functions that make up each stage of the CL instance is purely illustrative; other groupings are possible and may vary depending on the preferences of the ZSM framework owner and from one implementation to another.

Figure 7.3-2 depicts another CL instance deployed within a management domain or within the E2E management domain. In this deployment view, the Closed Loop can be viewed as an entity and the internal stages implementation of the CL instance is not externally visible, which means the implementation of the internal capabilities and internal management service end points between the stages of the CL do not necessarily follow the ZSM framework specifications and may be proprietary. However, one or more of the CL stages may support ZSM management service end points and may interface with the integration fabric (illustrated as dotted lines in the Figure 7.3-2).

Figure 7.3-2 shows two management service categories that are provided for Closed Loop management. Closed Loop Governance (CLG) services are the services that provide the capabilities to allow external entities to govern the Closed Loop (the details see clause 8.1). Closed Loop Coordination services (CLC) are the services that provide the capabilities to coordinate multiple Closed Loops (the details see clause 8.2).

As shown in Figure 7.3-2, the management service provided by external entities may also be consumed by the Closed Loop (CL), for example, Closed Loop instance within a management domain may consume external data collection services to collect the data outside the management domain. The external services consumed by Closed Loop are out of scope of this specification.

Similarly to the case illustrated in Figure 7.3-1, the externally visible capabilities and external management service end points for the interactions between the Closed Loop instance and other management entities in the ZSM framework are based on ZSM framework specifications and controlled by the governance management services. As examples, these external management service end points are used to provide the Closed Loop Governance (CLG) and functional inputs and outputs. Detailed specifications are provided in clauses 8.1 and 9.2.

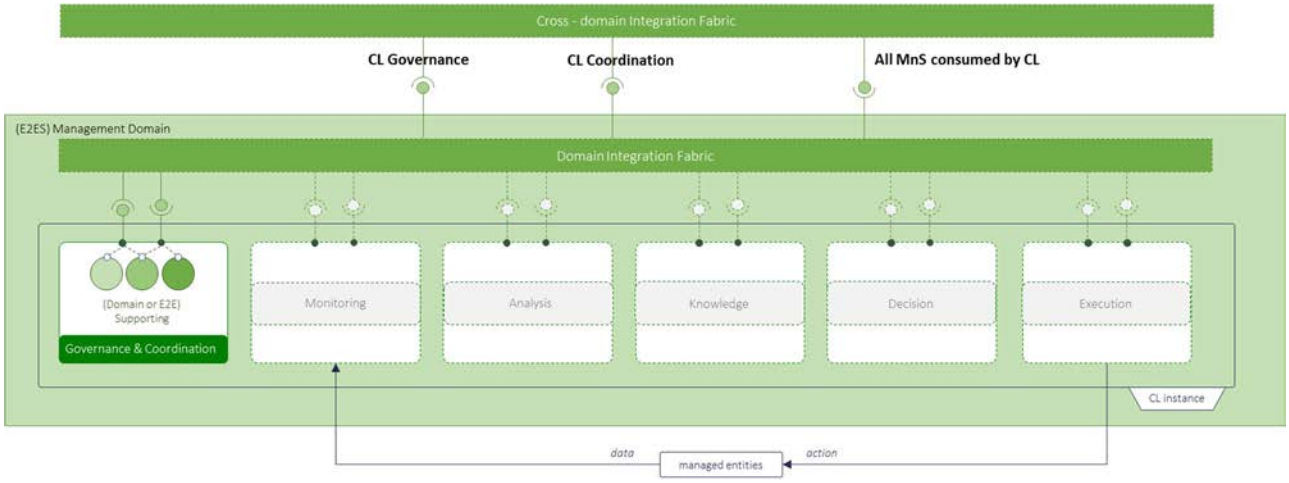

Figure 7.3-2: Deployment view of a Closed Loop instance where the internal stages implementation is not externally visible

7.4 Closed loop types

Different types of Closed Loops are supported in the ZSM framework. The different types of Closed Loops share some common characteristics and behaviours; however, they also differ on some other aspects.

This clause defines the different types of Closed Loops supported in the ZSM framework, outlines their commonalities and differences.

- Made-to-Order Closed Loops

Made-to-Order Closed Loops (M2O-CL) are assembled on demand by ZSM framework owners, or by other entities on behalf of the ZSM framework owners, using capabilities offered by the ZSM framework.

The internal and external interactions and capabilities of M2O-CLs are in-scope of this specification. As such, the M2O-CLs components, as well as the communication and interoperation between M2O-CL components, are based on ZSM-compliant building blocks, as defined in clause 6.1.2 of ETSI GS ZSM 002 [2] or in the present document.

The ZSM lifecycle scope of M2O-CLs comprises the preparation, commissioning and operation phases. More details in clause 8.1.3.

- Ready-Made Closed Loops

Ready-Made Closed Loops (RM-CL) are assembled by ZSM framework vendors prior to their use in the ZSM framework, using capabilities outside of the ZSM framework.

The RM-CLs components, as well as the communication and interoperation between the RM-CL components, are out-of-scope of the present specification. As such, only the external interactions and capabilities of the RM-CLs are in-scope of standardization.

The ZSM lifecycle scope of RM-CLs comprises the commissioning and operation phases. The preparation phase of RM-CLs is out-of-scope of the present document. More details in clause 8.1.3.

Closed Loops that have no or limited compliance with ZSM specifications and that require specific adaptation functions to operate within the ZSM framework are out of scope of the present document.

8 Closed-Loop Automation enablers

8.1 Closed Loop Governance

8.1.1 Introduction

Closed Loop Governance (CLG) is a set of capabilities that allows external entities to manage the life cycle of Closed Loops and configure their behaviour. Closed Loop Governance (CLG) may also be used to retrieve information about the Closed Loops such as their status (including health) and performance information.

The Closed Loop Governance (CLG) capabilities may include:

- Management of lifecycle of the Closed Loops, e.g. to provide the capability for a ZSM framework owner or an authorized consumer to start and stop Closed Loops. More details in clause 8.1.3.
- Management of CL models. More details in clause 8.1.4.
- Configuration of policies, rules, triggers and priorities for the Closed Loops.
- Conveying status and performance information of the CLs.

8.1.2 Options for governing a Closed Loop

CLs within management domains can be controlled by internal or external governance.

When internal governance is applied, the CLs in a management domain are governed by management services internal to the management domain and may provide status and, optionally, performance information to external authorized entities.

When external governance is applied, the CLs in a management domain are governed by authorized entities external to the management domain.

Both means of governance can also co-exist, when some aspects of the CLs in a management domain are governed by services internal to the management domain while others are governed externally.

When external governance is applied, the management service exposure configuration service from the ZSM framework integration fabric is used to determine which aspects may be governed externally.

In both ways of governance (internal or external), the capabilities specified in the Closed Loop Governance (CLG) service in clause 9.2.2 may be used.

8.1.3 Lifecycle management of Closed Loops

8.1.3.1 Closed loops lifecycle phases and activities

A CL is a managed entity within the ZSM framework and has its own lifecycle.

The lifecycle of CLs defines the mandatory and optional phases, activities and states of CLs. It provides to the ZSM framework owner, or other entities interacting with the CL, a unified view on the possible activities and states of the CL; and identifies the interacting entities and typical interactions among them (such as for the CL creation, CL deployment, CL coordination, CL modification, CL termination, etc.)

The lifecycle of a CL is composed of different phases and activities as shown in Figure 8.1.3.1-1.

The design-time of a CL comprises the preparation phase. The CL is defined during the design-time based on the CL model (see clause 8.1.4).

The run-time of a CL comprises the commissioning, operation and decommissioning phases. During the run-time the CL can get instantiated and activated and Closed Loop Governance (CLG) needs to be applied.

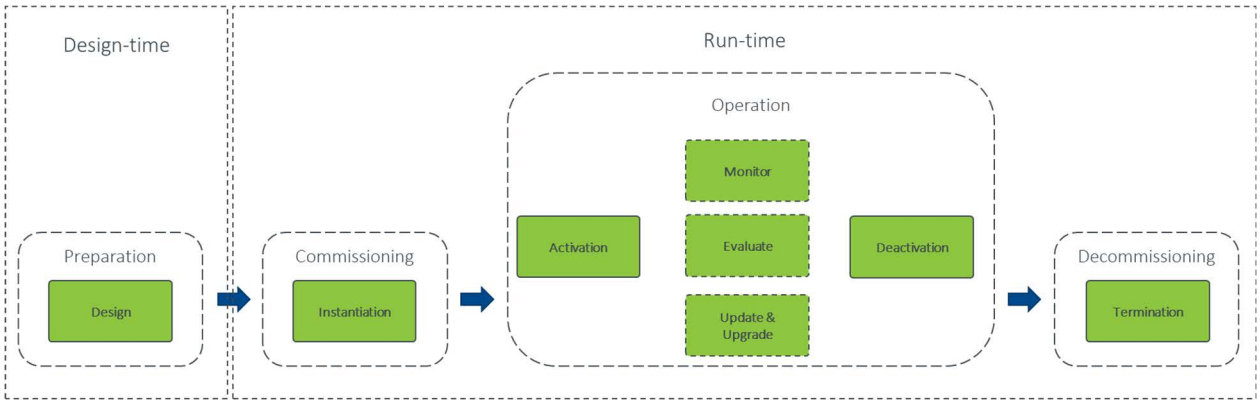

Figure 8.1.3.1-1: Closed loop lifecycle phases and activities

NOTE 1: The notion of CL phases is different from the notion of CL stages. CL stages represent the functional split internal to the CL. CL phases represent activities and states resulting from interactions with entities external to the CL.

NOTE 2: The CL lifecycle may be independent of the lifecycle of managed entities the CL is actuating upon, e.g. network slice, yet interactions between the two lifecycles may exist.

Each phase includes activities that may be triggered by events and that may result in the CL transitioning from one state to another (or falling back in the same state). The execution of the activities may be associated with conditions and constraints. It is part of the CL Lifecycle Management (LCM) to ensure the outcomes of each activity so that the CL can move to the next phase in its lifecycle.

The CL lifecycle should be model-based, which implies that not all the phases and activities (as in Figure 8.1.3.1-1) may be applicable to a CL instance. More details on the CL model to be used to specify the lifecycle phases and activities can be found in clause 8.1.4.

The CL LCM capability and the CL models management capability are defined by the Closed Loop Governance (CLG) service, as defined in clause 9.2.2.

8.1.3.2 Preparation phase

The preparation phase precedes the creation of a CL. It includes the activity of CL design whereby the CL is created based on the target use case and a specific CL goal.

The design activity is mandatory for all CLs that implement the preparation phase. The design activity results in an artefact where the CL is described, where the operator sets the boundaries and the objectives for CL. The semantics of this artifact is out of scope of this specification, but it shall be based on the CL model defined in clause 8.1.4.

8.1.3.3 Commissioning phase

In the commissioning phase a designed instance of the CL is instantiated. The instantiation activity includes the creation and the registration of the CL and its stages in the ZSM framework. This activity includes the creation of the stages and association with the CL instance. Alternatively, the association of stages may happen between already existing stages and a CL instance. This phase also includes configuration where parameter values are set.

The instantiation activity is mandatory in all CLs that implement the commissioning phase. The instantiation activity results in a CL that is ready for operation.

8.1.3.4 Operation phase

In the operation phase the CL is first activated. The activation activity includes actions that make the CL run and pursue its goal(s). It may include the subscription to the relevant communication channels provided by the ZSM integration fabric(s).

The monitor activity typically includes the real-time or periodic calculation of KPIs that are relevant to the CL and comparison with the goal(s) assigned to the given CL. This activity may result in further actions that involve the other activities in the operation phase, e.g. evaluate and update & upgrade, in order to change the CL settings and improve its performance.

The evaluate activity also includes the evaluation of results of Execution stage of CLs by e.g. investigating differences between the current traffic data and the data taken before the execution. The criteria of this evaluation can be done by specific values such as SLS on CL model defined in clause 8.1.4.

The update & upgrade activity includes actions that change the settings of the CL instance to change its behaviour and improve its performance to pursue the assigned goal(s). The update may include changes in the parameters of the management functions that constitute the CL (e.g. changing data sources, KPIs being calculated, models, policies, etc.). The upgrade may include changes in the software version of the management functions.

These activities can be executed dynamically while the CL is regularly operating and executing actions. They can also be executed upon a request received from authorized entities external to the ZSM framework. For example, an external authorized entity, who is typically a network operator, can change configuration data of a MnF consisting of a CL and/or modify parameters of CL model attributes through the Closed Loop Governance (CLG) service.

The last activity before the CL leaves the operation phase is the deactivation. The deactivation activity includes actions that make the CL stop to run, which may include deregistration from the communication channels in the ZSM integration fabric(s).

The activation and deactivation activities are mandatory in all CLs that implement the operation phase. All other activities are optional.

8.1.3.5 Decommissioning phase

In the decommissioning phase the CL is terminated. The termination activity includes the deregistration of the CL and its stages in the ZSM framework. The termination activity is mandatory in all CLs that implement the decommissioning phase.

The termination activity is irreversible. After termination the CL does not exist anymore in the ZSM framework as an entity. However, the management functions that provide the management service that were part of the CL may still exist.

8.1.4 Closed loop models

8.1.4.1 Introduction

Closed loops within the ZSM framework are represented by models. These models can be applied to different phases of the lifecycle management of Closed Loops. Closed loop models are presented below in terms of relationship diagram and inheritance diagram.

Figure 8.1.4.1-1 shows the class relationship diagram and Figure 8.1.4.1-2 shows the inheritance diagram of a model that can be applied in different phases of the Closed Loops LCM.

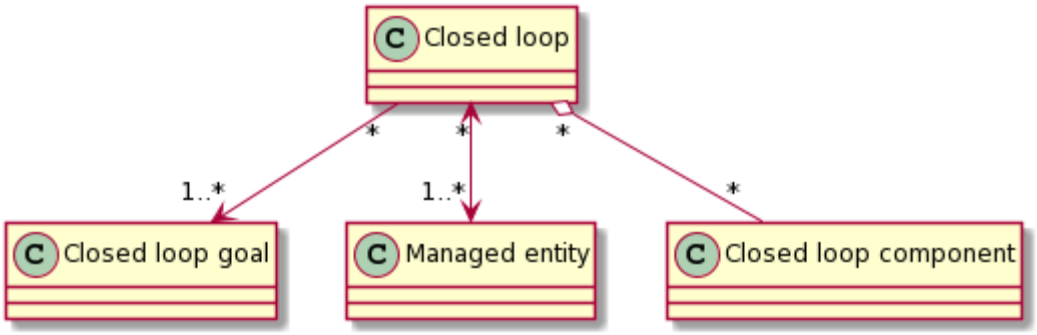

Figure 8.1.4.1-1: Closed loop class relationship diagram

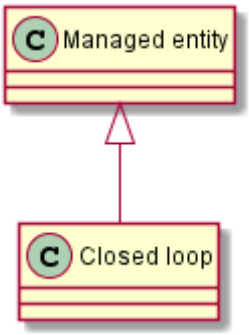

Figure 8.1.4.1-2: Closed loop class inheritance diagram

8.1.4.2 Closed loop class

The Closed Loop class represents the elements of a CL. All CL within the ZSM framework shall have at least one specified CL goal (defined in clause 8.1.4.3) and at least one target managed entity (defined in clause 8.1.4.4).

The Closed Loops shall also have one or more Closed Loop components (defined in clause 8.1.4.5).

The Closed Loops are expected to actuate upon the defined managed entities to meet the specified Closed Loop goal. Further details on the required management capabilities may be defined by the Closed Loop components.

Table 8.1.4.2-1 represents the attribute table of the Closed Loop class.

Table 8.1.4.2-1: Closed loop class

Attribute name	Description and properties
closedLoopInstanceUniqueId	It indicates the identifier of the CL instance. See notes 1, 2, 3 and 4. – **Support**: Mandatory – **Type**: String – **Multiplicity**: 1
closedLoopLifeCyclePhases	It indicates the list of supported lifecycle phases of this CL type. Allowed values in the list are Preparation, Commissioning, Operation, and Decommissioning. The lifecycle phases are specified in clause 8.1.3.1. See note 5. – **Support:** Mandatory – **Type:** Enum – **Multiplicity:** 1..4
currentClosedLoopLifeCyclePhase	It indicates which CL life cycle phase the CL is in. – **Support:** Mandatory – **Type:** Enum – **Multiplicity:** 1
closedLoopPriority	It indicates a priority of the CL. It is set to avoid conflicting actions to the same managed entity. See notes 6 and 7. – **Support:** Mandatory – **Type:** Enum – **Multiplicity:** 1
closedLoopTypeDescription	It indicates a description of the CL type. See note 8. – **Support:** Optional – **Type:** String – **Multiplicity:** 1
closedLoopGoal	It indicates goals of the CL. See clause 8.1.4.3. – **Support:** Mandatory – **Type:** closedLoopGoal – **Multiplicity:** 1..N
manageableEntityList	It indicates the types/categories of entities that can be managed by the CL. Entities are not instantiated entities, but categories/types/classes or range of products/elements. – **Support:** Mandatory – **Type:** managedEntity – **Multiplicity:** 1..N
targetEntityList	It indicates the entities that the CL instance will have to manage after being successfully deployed/instantiated. – **Support:** Mandatory – **Type:** managedEntity – **Multiplicity:** 1..N
closedLoopComponentList	It indicates the composable unit of CL, e.g. CL stages and knowledge. See clause 8.1.4.5. – **Support:** Mandatory – **Type:** closedLoopComponent – **Multiplicity:** 1..N
closedLoopPolicy	Defines policies applicable to the CL instance. See notes 9 and 10. – **Support:** Optional – **Type:** Policy – **Multiplicity:** 1..N
NOTE 1: The identifier is unique in the MD level. NOTE 2: The CL is uniquely identifiable within the ZSM framework level by combination of MD and the identifier. NOTE 3: The identifier is allocated when the CL is instantiated until the time it is terminated. NOTE 4: The identifier is used in the commissioning and operation phase. NOTE 5: The value is set in the preparation phase. NOTE 6: The priority may be used for conflict resolution. How this is used is out-of-scope of the present document. NOTE 7: The value is set in the preparation phase. NOTE 8: A description of the CL can be written in a free format. NOTE 9: The closedLoopPolicy attribute may be used to define policies of the CL instance for aspects such as autonomy, supervision, reporting, execution, coordination, usage monitoring, etc. NOTE 10: The definition of the type "Policy" is out-of-scope of the present document.	

8.1.4.3 Closed loop goal class

The Closed Loop goal determines the objective(s) a Closed Loop shall meet.

The goal specifies the wanted behaviour of the CL and the expectations that need to be met during the CL execution.

At the preparation phase, the Closed Loop goal is defined by the ZSM framework vendor, the ZSM framework owner, or by other entities on behalf of the ZSM framework owner. Multiple goals may be defined for a given Closed Loop.

At the commissioning phase, the Closed Loop goal is configured with initial, reference values by the ZSM framework owner, or by other entities on behalf of the ZSM framework owner.

At the operation phase, the values of the Closed Loop goal parameters can be changed by the ZSM framework owner, or by other entities on behalf of the ZSM framework owner, to adapt the CL behaviour. The CL goal can be managed at operation phase by using the Manage goal capability of Closed Loop Governance (CLG) service (clause 9.2.2). The changes that are allowed to be made to a CL goal are:

i) selecting a goal from the list of goals previously defined in the preparation phase; and

ii) changing the values of the parameters of the goal that is currently in use.

Table 8.1.4.3-1 represents the attribute table of the Closed Loop goal class.

Table 8.1.4.3-1: Closed loop goal class

Attribute name	Description and properties
closedLoopGoalId	It indicates the identifier of the CL instance goal. See notes 1 and 2. – **Support:** Mandatory – **Type:** String – **Multiplicity:** 1
closedLoopGoalDescription	Describes the Closed Loop goal. Description of the Closed Loop goal statement in a human-readable form. See note 3. – **Support:** Optional – **Type:** String – **Multiplicity:** 1
closedLoopGoalStatement	The Closed Loop goal statement can be a declarative or an imperative statement. The declarative statement of a CL goal is an intent that expresses the expectations to be met by the CL, including requirements and constraints. While closedLoopGoalDescription is in a human-readable form, the closedLoopGoalStatement shall be in a machine-processable form. See note 4. The imperative statement of a CL goal is a service level specification that expresses the minimum acceptable standard of service to be met. See notes 5, 6 and 7. – **Support:** Mandatory – **Type:** goalStatement – **Multiplicity:** 1
NOTE 1: The identifier is unique in the MD level. The goal is globally identifiable with MD and the closedLoopInstanceUniqueId in the Closed Loop class. NOTE 2: The identifier may be used to reference goal(s) that need to be changed, e.g. set, updated, delegated, etc. NOTE 3: The description of the Closed Loop goal is provided in the preparation phase of the Closed Loop life cycle. NOTE 4: Examples of intents are: "minimize latency", "keep throughput higher than 1 Mbps", "pro-actively mitigate faults in the transport domain", etc. NOTE 5: The statement of the Closed Loop goal is provided in the preparation phase of the Closed Loop life cycle. NOTE 6: Changing the statement of the Closed Loop in operation phase is allowed (e.g. see description above Table 8.1.4.3-1). NOTE 7: The definition of the type "goalStatement" is out-of-scope of the present document.	

8.1.4.4 Managed entity class

A managed entity is defined in clause 3.1.

Table 8.1.4.4-1 represents the attribute table of the managed entity class.

Table 8.1.4.4-1: Managed entity class

Attribute name	Description and properties
managedEntityId	It indicates the identifier of the managed entity. See notes 1 and 2. – **Support:** Mandatory – **Type:** String – **Multiplicity:** 1
managedEntityType	It indicates a type of managed entity. Allowed values are managed resource, managed service, or Closed Loop. – **Support:** Mandatory – **Type:** Enum – **Multiplicity:** 1
NOTE 1: The identifier is unique in the ZSM framework level. NOTE 2: When the managedEntityType is a Closed Loop, the managedEntityId is the closedLoopInstanceId of the Closed Loop.	

8.1.4.5 Closed loop component class

Closed loop components are management functions utilized by the Closed Loops to realize their operations (or operational flows).

Management functions for Closed Loop Governance (CLG), coordination, stages and knowledge (as described respectively in clauses 8.1, 8.2 and 7.2) are examples of Closed Loop components.

Table 8.1.4.5-1 represents the attribute table of the Closed Loop component class.

Table 8.1.4.5-1: Closed loop component class

Attribute name	Description and properties
closedLoopComponentDescription	Describes the functionality of the Closed Loop component in a human-readable form. See note 1. – **Support:** Optional – **Type:** String – **Multiplicity:** 1
inputDataList	Lists the mandatory and optional information the Closed Loop component can receive from other entities internal or external to the Closed Loop. See notes 2 and 3. – **Support:** Optional – **Type:** Enum – **Multiplicity:** 1..N
outputDataList	Lists the information the Closed Loop component can provide to other entities internal or external to the Closed Loop. See notes 4 and 5. – **Support:** Optional – **Type:** Enum – **Multiplicity:** 1..N
producedManagementCapabilitiesList	Lists the capabilities offered by the Closed Loop component for consumption by authorized entities. See note 6. – **Support:** Mandatory – **Type:** Enum – **Multiplicity:** 1..N

Attribute name	Description and properties
consumedManagementCapabiltiesList	Lists the capabilities consumed by the Closed Loop component for its functioning. See note 6. – **Support:** Mandatory – **Type:** Enum – **Multiplicity:** 1..N
NOTE 1: The description of the Closed Loop component is provided in the preparation phase of the Closed Loop life cycle. NOTE 2: The nature of the information is provided in the preparation phase of the Closed Loop life-cycle. NOTE 3: Examples of entries in the inputDataList are specific to the function or the requirements of the Closed Loop component. NOTE 4: The nature of the information is provided in the preparation phase of the Closed Loop life-cycle. NOTE 5: Examples of entries in the outputDataList are specific to the function or the requirements of the Closed Loop component. NOTE 6: The capabilities are provided in the preparation phase of the Closed Loop life cycle.	

8.1.5 Interactions between Closed Loops and external entities

8.1.5.1 Introduction

An external entity may interact directly with a CL or indirectly, via other CLs. The external entity should interact with stages of the CL. An external entity is external to the CL, but could be a part of the ZSM framework.

8.1.5.2 Interactions based on policies

A policy defines a set of actions that an external entity requires a CL to perform when a given set of conditions are met. Several policies, emanating from one or several external entities, can be performed by a CL simultaneously.

To support automation, the actions are performed in order to reach or maintain a specified behaviour in a deterministic manner. A policy may take the form of a set of rules which, when triggered by explicit or implicit events, evaluate conditions and generate appropriate actions.

Whereas traditional polices facilitate automation, zero-touch policies should extend and advance policy concepts to make them adaptive, creating an important basis for achieving autonomic behaviour.

At least two basic types of policy should be supported:

Resource policy: Related to how the managed object that a CL act on stays within its defined constraints, and how it optimizes its delivery capabilities given those constraints. Used to specify corrective behaviour when the constraints (and/or optimization criteria) cannot be met.

Service policy: Used to set characteristics and control the externally observable behaviour of services and service instances (deliveries, individual sessions), acted on by the CL. These policies typically form part of the definition of the service.

NOTE: The use of policies for zero-touch automation is a subject that needs to be further investigated and specified. It should be addressed in subsequent work items.

8.1.5.3 Interactions based on intents

An intent specifies wanted characteristics or behaviour of a managed object or of a system composed of several managed objects. The intent does not put requirements towards any specific management entity, since it does not explicitly address the management system, i.e. the system implementation managing the objects. The objective is to reach a desired result without the need for precise knowledge on how to obtain it.

Intent may be an abstracted way to specify business or operational desired state of a system, without specifying how to achieve it. Whereas intent specifies a goal for a CL to accomplish, policy specifies a behavioural pattern that a CL should follow.

An intent may be used by a stakeholder/consumer, e.g. human operator or a business support system (BSS), with limited insight in the specifics of the management domain, therefore an intent is best expressed in a form that suits the stakeholder.

An intent is declarative. It shall delegate to the management system to explore options for finding the optimal solution. An intent declares the expected results rather than prescribing a specific solution. Ideally, an intent expresses a utility level goal that describes the quantitative and qualitative properties of a satisfactory outcome, or a range for those properties, instead of requiring a specific outcome.

An intent is agnostic concerning technology and portable between different implementations. The expectations expressed by intent may originate from contracts as well as business strategies. It does not change if the underlying system is replaced or modified, unless the contract or strategy stipulates specifics about the implementation. Consequently, intent goals should not have dependency of system generations and implementations to allow for portability between them.

Consequently, an intermediate context-aware process is needed to extract information from the intent and convey the information to the appropriate CLs in a useful form. The intent intermediation may include proprietary elements.

A management system should have the ability to support multiple intents and it is expected to consider them altogether. There is a risk that different intents will specify contradicting wanted characteristics or behaviour, which will create conflicts.

Unlike traditional software systems, where requirements are analysed offline to detect and resolve conflicts prior to implementation, intents may be added during run-time. Therefore, an essential capability of a management system should be to detect and resolve conflicts between multiple intents. Information emanating from multiple, potentially conflicting, intents should either be conveyed directly as input to a CL or combined, by an entity separate from the CL, to form an unambiguous input to the CL.

NOTE 1: Artificial intelligence (AI) and machine learning (ML) are technologies that may be engaged in the interpretation. The use of AI/ML for the purpose of interpreting intents is a subject that needs to be further investigated and specified. It should be addressed in subsequent work items.

NOTE 2: The possibility of using intent to govern Closed Loops requires further specification and is not in the scope of the present document.

8.2 Closed Loop Coordination

8.2.1 Introduction

Closed Loop Coordination (CLC) is a set of capabilities that allows multiple CLs running within the ZSM framework to be coordinated, with the main objective of improving their performance and the fulfilment of their goal(s).

The Closed Loop Coordination (CLC) involves different types of interactions between multiple Closed Loops during their run-time. These interactions may happen, e.g. for delegation and escalation of goal(s) or issues, or for coordination of actions and sharing on information produced by the different CL stages.

Coordination of conflicting CLs is also part of Closed Loop Coordination (CLC) capabilities. Conflicts between CLs can negatively impact their operations. Conflicts can occur between two or more CLs, involving the same or different sets of managed entities.

NOTE: The effects of such conflicts can range from minor to severe performance degradation.

Clause 8.2.2 shows how multiple Closed Loops can be organized within the ZSM framework. Clause 8.2.3 specifies the coordination between hierarchical Closed Loops and clause 8.2.4 specifies the coordination between peer Closed Loops. The coordination mechanisms in clauses 8.2.3 and 8.2.4 are based on high-level information specified in the CL model, e.g. goals. Clause 8.2.5.4 specifies the pre-execution coordination and clause 8.2.5.5 specifies the post-execution coordination. The coordination mechanisms in clauses 8.2.5.4 and 8.2.5.5 are based on detailed information that are part of the CL flows, e.g. action plans and actions.

8.2.2 Relationships between different Closed Loops

The relationship between two CLs within the ZSM framework can be classified as hierarchical or peer CLs. Hierarchical relationship is the case when one Closed Loop is authorized to control another Closed Loop regarding a defined set of aspects. While peer relationship is the case when two Closed Loops have reasons to influence each other's behaviour, however, one Closed Loop is not responsible for the other and both exist independently.

In addition to the classification above, there may be further classification based on where each Closed Loop runs:

a) both CLs at the same management domain;

b) one CL at the E2E management domain and another at a management domain.

A convolution of the two classifications can be created resulting in Figure 8.2.2-1 and Figure 8.2.2-2.

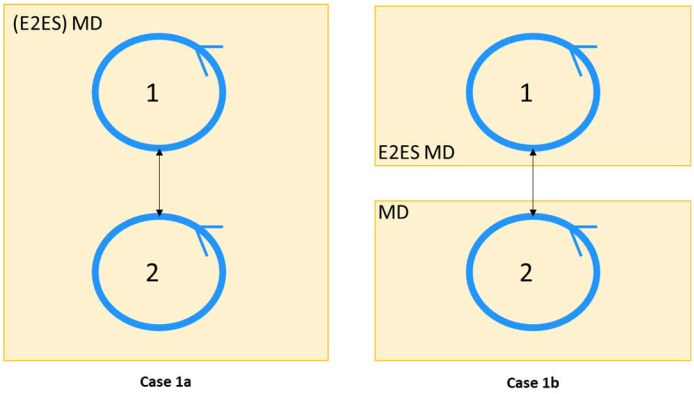

Figure 8.2.2-1: Hierarchical Closed Loops

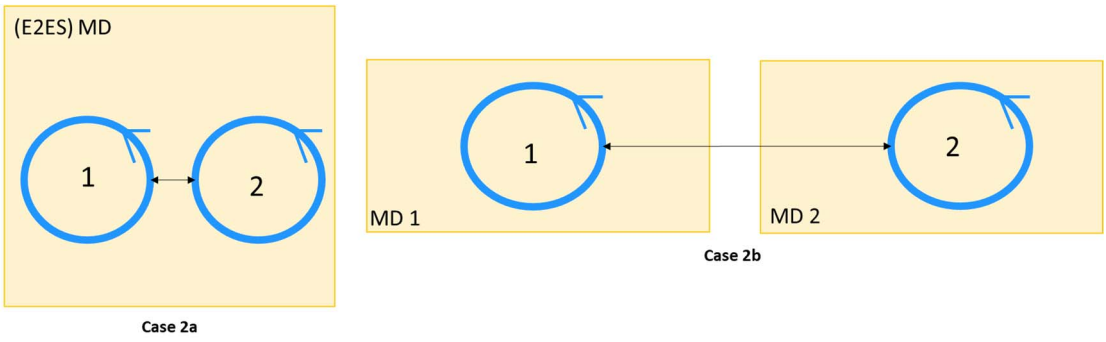

Figure 8.2.2-2: Peer Closed Loops

NOTE 1: How CLs relationships are established within the ZSM framework is out of scope of the present document and it depends on the use case and choice of operator.

NOTE 2: The relationships between CLs can form different structures, where a tree structure is one specific case.

Based on the classification and organization of CLs shown in Figure 8.2.2-1 and Figure 8.2.2-2, a non-exhaustive list of interactions is specified in the following clauses.

A CL should take into consideration accuracy and timeliness of information received from another CL. This is especially important in the hierarchical case when controlling subordinate CLs.

8.2.3 Coordination between hierarchical Closed Loops

This clause describes the coordination of interactions between hierarchical Closed Loops as illustrated in Figure 8.2.2-1, cases 1a and 1b. The end-to-end communication services are composed of multiple services managed by different management domains. There may be different CLs in the (E2ES) MD that are hierarchically organized (Figure 8.2.2-1 case 1a) or there may be CLs in the E2ES MD that have to interact with the CLs in multiple subordinate MD (Figure 8.2.2-1 case 1b). In both cases the subordinate CLs are responsible for optimization and self-healing within their scope, while the superior CLs are responsible for the coordination and optimization within their scope.

The subordinate CLs deployed in the E2ES MDs or in the MDs perform local optimizations which might not result in a global, end-to-end optimum or which might even be in conflict to each other. To this end the superior CLs shall be able to coordinate the decisions of subordinate CLs. Such coordination can happen in the following, not exclusive ways:

- Delegation - The superior CLs delegates respective goal(s) to the subordinate CLs, e.g. by setting the policies and/or the intents in a way that allow the subordinate CL to act autonomously.
- Escalation - If a subordinate CL is not able to achieve the goal(s) assigned to it, it escalates the situation to the superior CL in the E2ES MD.

Depending on the actual interaction coordination scenario, different combinations of the above approaches in a hybrid mode are possible.

Delegation and escalation coordination may be based on capabilities provided by the Closed Loop Governance (CLG) service (clause 9.2.2) and the Closed Loop information reporting service (clause 9.2.3).

8.2.4 Coordination between peer Closed Loops

This clause describes the interactions between peer Closed Loops as illustrated in Figure 8.2.2-2, cases 2a and 2b. CLs in the E2ES MD or in MDs may benefit from exchanging information to cooperate in achieving a common objective.

The peer CLs may perform local operations which might be in conflict to each other. They can also request the resolution of issues within their local scope that could be resolved by another peer CL. Such coordination can happen in the following, not exclusive way:

- Cooperation - Two or more peer CLs that are aware of each other can exchange their goal(s), their model(s) and their information (see clause 9.2.3). Based on this information, the peer CLs can adjust their own policies and/or intents, to achieve a common objective and avoid conflicts. A CL can also request a peer CL to assist in the resolution of an issue.

Cooperation may be based on capabilities provided by the Closed Loop Governance service (clause 9.2.2) and the Closed Loop information reporting service (clause 9.2.3).

8.2.5 Closed Loop Coordination services

8.2.5.1 Introduction

The coordination of Closed Loops should happen preferably in an automated way to maximize the value of Closed Loop operation by providing continuity of automation, and coping with dynamicity and complexity.

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.).
- Capability to evaluate the impact and effectiveness of Closed Loops actions after their execution (e.g. impact assessment).

Such capabilities can be mutualized as common Closed Loop Coordination services, as exemplified by Figure 8.2.5.1-1.

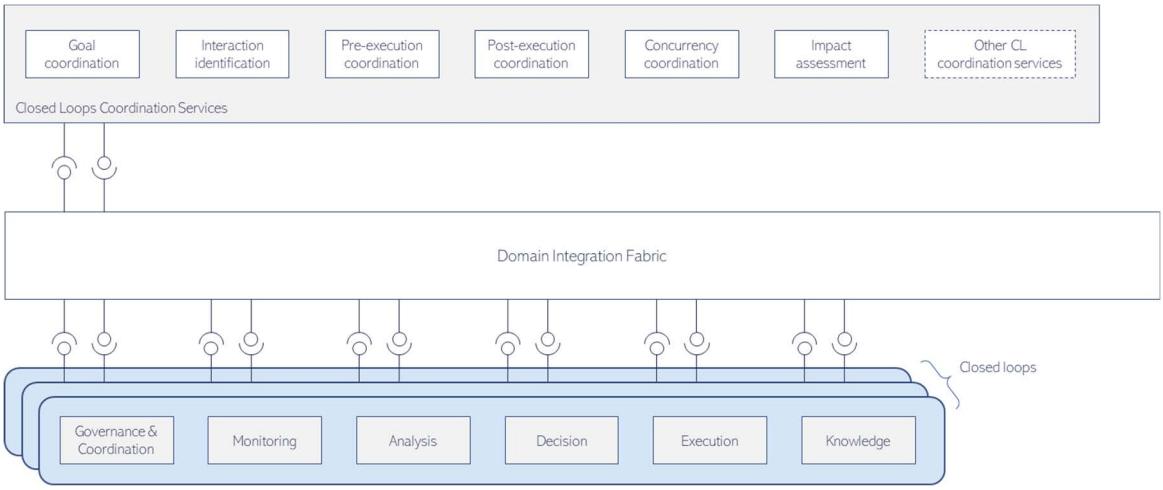

Figure 8.2.5.1-1: Example of Closed Loop Coordination services

The Closed Loop Coordination services can interact with each other in different ways and at different times. An exemplary timeline describing the typical occurrence time of these services is depicted in Figure 8.2.5.1-2. For readability, the set of possible interactions flows between the different Closed Loop Coordination services is not depicted.

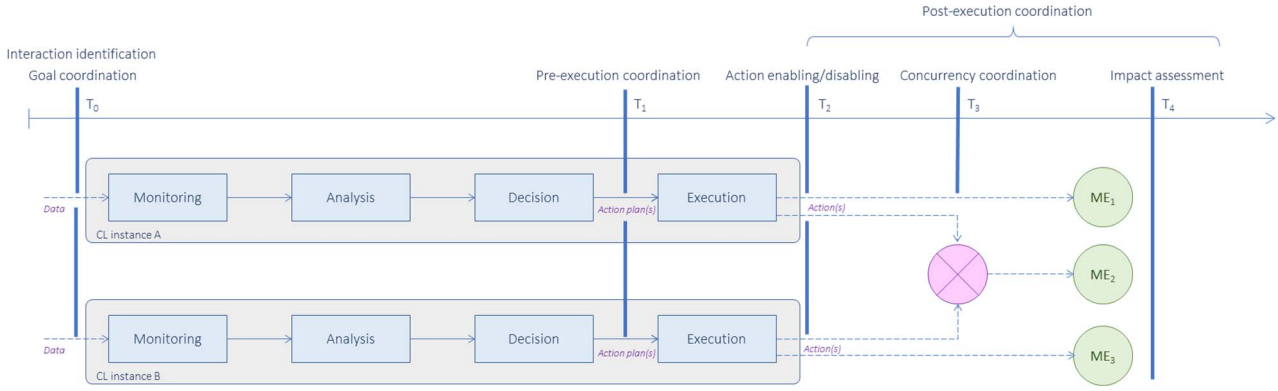

Figure 8.2.5.1-2: Exemplary Closed Loop Coordination timeline

8.2.5.2 Interactions identification

Closed loops interactions identification determines and characterizes the interactions that may exist between two or more CL instances. The interactions identification service may be used by other Closed Loop Coordination (CLC) services designed to manage or arbitrate the coordination between CL instances and that need to know beforehand if interactions exist and, optionally, other information about the interaction(s), and details on CL instance attributes involved in the interactions, etc.

Interactions identification typically occurs when new Closed Loops are instantiated, or when other coordination services require information about interactions between given Closed Loops. Those occurrence times are represented as time T0 in Figure 8.2.5.1-2, although a single time positioned at the start of the Closed Loop flow does not reflect accurately the time occurrences that interactions identification may have.

8.2.5.3 Goal coordination

Goal coordination is responsible for aligning goals (and goal parameters) that should be achieved by the various deployed Closed Loops. Typically, this operation is done by a human operator. Given the potentially high number and diversity of Closed Loops, the operator may want to automate this process of setting and coordination goals for the Closed Loops. This process includes the negotiation phase among the Closed Loops in the cases where one Closed Loop actions may hinder another Closed Loop from reaching its targets. Such interaction can relate, for instance, to the coordination between peer Closed Loops, as described in clause 8.2.4.

Each Closed Loop deployed in the network is responsible for specific goal(s), e.g. optimization of handover performance or minimization of energy consumption. For each Closed Loop, the exact target to achieve for the respective goal needs to be stated, specifically, in way that balances that goal with other related CL goals.

Goal coordination may also use information from other services such as the impact assessment service (see clause 8.2.5.5.4) to gain additional knowledge and a broader understanding on how goal alignment between Closed Loops could be achieved.

Goal coordination typically occurs when new Closed Loops are instantiated or when other coordination mechanisms cannot achieve to address long-term dependency between Closed Loops interactions, thus requiring (re-)alignment of their goals. Those occurrence times are represented as time T0 in Figure 8.2.5.1-2, although a single time positioned at the start of the Closed Loop flow does not reflect accurately the time occurrences that goal coordination may have.

8.2.5.4 Pre-execution coordination

8.2.5.4.1 Introduction

Pre-execution coordination refers to the management of interactions between Closed Loops before the triggering of the Execution stage, typically occurring at time T1 as depicted n Figure 8.2.5.1-2. Pre-execution coordination is responsible for optimizing the effects of actions taken by interacting Closed Loops.

Interacting Closed Loops and/or the Closed Loops coordination functionality receive one or more action plans. The action plans are provided prior to their execution by the interacting Closed Loops. Pre-execution coordination relies on capabilities for identifying conflicts and for determining which combination of action plans contributes best to the coordination goal. These capabilities are provided by the Pre-execution coordination service, as specified in clause 9.3.2.

8.2.5.4.2 Action plans conflict detection

If executed without coordination, actions taken by interacting Closed Loops may cause undesirable effects on the managed entities. To avoid such detrimental situations, the Pre-execution coordination service is used to detect conflict before the interacting Closed Loops execute their actions. The conflict detection works as follows:

1) Retrieve the action plans which contain the information of target resources and scheduled time for execution.

2) Check if there are any conflicting actions based on the provided information.

3) Notify the detected conflict(s) to the related Closed Loops and/or the Closed Loops coordination functionality.

8.2.5.4.3 Action plans selection

The Pre-execution coordination service is used to select the most appropriate combination of action plans to be executed.

The most appropriate combination of action plan(s) can be evaluated by multiple means and by using, for instance, historical data and/or operational data. This service can be used to address the detected conflicts identified as well as the non-conflicting action plans provided by the interacting Closed Loops. The action plans selection works as follows:

1) Retrieve the action plans which contain the information of target resources, scheduled time for execution, and other additional information such as historical results of the proposed actions.

2) Assess each plan and choose the most appropriate combination of action plan(s) based on the selection policy.

3) Notify the selected action plan(s) to the related Closed Loops and/or the Closed Loops coordination functionality.

8.2.5.5 Post-execution coordination

8.2.5.5.1 Introduction

Post-execution coordination refers to the management of interactions between Closed Loops after the triggering of the Execution stage, and typically spans between times T2 and T4 as depicted in Figure 8.2.5.1-2. Post-execution coordination is responsible for ensuring that all actions that are executed result in positive outcomes and any actions that do not are identified and flagged as such.

8.2.5.5.2 Action enabling and disabling

Coordination amongst Closed Loops may require disabling actions (actions are changes that a Closed Loop can perform over a managed entity such as configuring an attribute) of a Closed Loop. Action enabling or disabling typically occurs at time T2 as depicted in Figure 8.2.5.1-2.

8.2.5.5.3 Concurrency coordination

Concurrency coordination ensures that the actions of CL instances that have managed entities in common are applied consistently and in accordance with the operational policies, rules, or decision criteria. A typical example is to compare the value assigned to the closedLoopPriority attribute of the CL instances under coordination to decide in which order the CL instances action(s) should be executed on the shared managed entity.

Concurrency coordination orchestrates access control to managed entities, and avoids race conditions. For example, if two or more CL instances decide on actions resulting in different changes to the same managed entity(ies) at the same time, the concurrency coordination can identify the issue and decide which of the CL instances can proceed with the execution of its action. Concurrency coordination typically occurs at time T3 as depicted in Figure 8.2.5.1-2.

8.2.5.5.4 Impact assessment

Impact assessment allows evaluating the direct and indirect effects of CL actions, and determining remediation measures to cover the following cases:

- For some Closed Loops, the scope of the action (be it in time, space, or network function) may not be known a priori, either by the Closed Loop itself or the Closed Loops coordination functionality. Correspondingly, any negative effects cannot be easily anticipated and most importantly, they may not be easily resolvable by simple if-then-else rules. However, post-execution coordination shall still be able to identify actions that lead to negative outcomes and flag them accordingly.

- For some Closed Loops, the expected, bounded scope of the action may be known either to the Closed Loop itself or to the Closed Loops coordination functionality. In some cases, even if not specified such scope may be easily derived from the description of the command(s) that are executed in the action.

In the above situations, the post-execution coordination should evaluate a wider scope and rely on the additional information (e.g. knowledge gained from other Closed Loops) to:

1) determine if there are unwanted outcomes;

2) diagnose if the executed action(s) is/are responsible for those outcomes, especially for the case where multiple Closed Loops have concurrently taken actions, and

3) determine what needs to be done to undo the degradation and to avoid it in future.

Impact assessment typically occurs at time T4 as depicted in Figure 8.2.5.1-2.

NOTE: Most of capabilities necessary for post-execution coordination are left for future stages of the specification. Some capabilities specified in clause 9.3.3 can be used for this purpose.

9 Specification of management services relevant to Closed Loops

9.1 New management services

In this clause new management services are introduced to the ZSM framework. They follow the conventions for management services specifications as in ETSI GS ZSM 002 [2] clause 3.4.

9.2 Management services for Closed Loop Governance

9.2.1 Introduction

This clause specifies ZSM management services related to Closed Loop Governance (see clause 8.1). The following sub-clauses contain the descriptions and the service definition tables of management services which are classified as Closed Loop Governance (CLG) related services.

9.2.2 Closed Loop Governance service

Each management domain may provide a Closed Loop Governance (CLG) service that is used for the governance of Closed Loops. The Closed Loop Governance (CLG) service is described in Table 9.2.2-1.

Table 9.2.2-1: Service definition

Service name	Closed Loop Governance (CLG) service
External visibility	OPTIONAL (see note 2).
Service capabilities	
Manage Closed Loops lifecycle (M)	Manage the Closed Loop lifecycle according to the Closed Loop model. This could include phases such as preparation, commissioning, operation and decommissioning, as specified in clause 8.1.3. The Closed Loop model(s) are specified in clause 8.1.4 and their management is done by the manage Closed Loop models capability of this management service.
Manage Closed Loop models (M)	Manage (create, read, update, delete, list) Closed Loops models, as specified in clause 8.1.4. Certain phase transitions (in Closed Loop life cycle) may be included in the Closed Loop model. Read and list operations are mandatory while the rest are optional.
Request issue resolution (O)	Request other entity(ies), e.g. peer Closed Loop(s) (see clause 8.2.2), to solve an issue which a Closed Loop is not able to solve.
Escalate issue (O)	Escalate an issue to be further solved by another entity when a Closed Loop in the subordinate management domain is not able to achieve the goal(s) assigned to it (as specified in clause 8.2.3). The issue can be related to previously delegated Closed Loop goal(s).
Manage Closed Loop policy (O)	Manage (create, read, update, delete, list) policies of a Closed Loop instance. See note 1.
Manage Closed Loop goal (M)	Manage (create, read, update, delete, list) goal(s) of a Closed Loop in the respective management domain(s) to allow the domain to act autonomously. This capability can be used for delegation of goal(s) (as specified in clause 8.2.3) between different Closed Loops, or for configuration of goal by the ZSM framework consumers.
NOTE 1: The policy of a Closed Loop instance refers to the closedLoopPolicy attribute of the Closed Loop model, as defined in clause 8.1.4.1. NOTE 2: This service is mandatory for internal use of management domains that implement Closed Loops, but its external visibility is optional.	

9.2.3 Closed loop information reporting service

The CL information reporting service allows providing current or past information of one or more Closed Loops or Closed Loop instances. The CL information may include the health status of the CL, the values of CL attribute(s) as defined in the CL model (defined in clause 8.1.4), the status of goal fulfilment related to managed (created, updated) CL goals in the past, etc. The Closed Loop information reporting service is described in Table 9.2.3-1.

Table 9.2.3-1: Service definition

Service name	Closed loop information reporting service
External visibility	OPTIONAL
Service capabilities	
Provide notifications about Closed Loop information (M)	Provide notifications about Closed Loop information.
Query Closed Loop information (M)	Query Closed Loop information.
Provide Closed Loop performance information (O)	Provide performance information on Closed Loops. Performance information could include for example the Closed Loop's expected impact time.
Configure service (O)	Configure what Closed Loop information are provided by this service.

9.2.4 Closed loop execution management service

The Closed Loops execution management service provides the ability to pause the execution of a Closed Loop at a "pause point". Pause point within a Closed Loop is a place, typically defined during the design phase, where the execution of the Closed Loop can be temporarily halted. When a pause point is enabled and reached during the execution of the Closed Loop, the execution of that particular flow in the Closed Loop chain is temporarily halted. The authorized consumer that enabled the pause point is notified. For example, when a pause point after the analytics stage is enabled and reached: the analytics insight(s) generated in that stage are not forwarded to the next stage of that specific Closed Loop. Instead a notification to the authorized management service consumer that enabled the pause point is provided together with the analytics insight(s). The authorized management service consumer may then choose to resume the execution or completely stop further execution of the Closed Loop. In addition, an additional optional capability of the authorized managed service consumer to request pausing the Closed Loop as soon as possible (the next reachable pause point) is provided. The management service is described in Table 9.2.4-1.

Table 9.2.4-1: Service definition

Service name	Closed loop execution management service
External Visibility	OPTIONAL
Service capabilities	
Provide Closed Loop pause point information (O)	Provides the supported pause points of a given a Closed Loop and the related information. Related information could for example describe the pause point and the notification received when the pause point is reached.
Enable/Disable pause point(s) (M)	Provides an ability for the authorized management service consumer to enable or disable pause point(s) for a Closed Loop. A time limit may be configured for enabled pause points. The authorized consumers may configure whether the Closed Loop automatically continues further execution or stops after the time limit is reached.
Provide notification for a pause point reached (M)	Provides a notification to the subscribers once a pause point is hit. For enabled pause points the notification is mandatorily transmitted to the authorized service consumer that enabled it. For disabled pause points the notification is optional and may just be used for logging purposes. A time limit until when the closed operation can be resumed may be provided.
Manage Closed Loop action plans (M)	Allow authorized entities to manage (R, U) Closed Loop action plans.
Continue Closed Loop execution (M)	The authorized consumer asks for continuing a paused Closed Loop execution.
Pause a Closed Loop (O)	Pause a Closed Loop at the next reachable pause point.

9.2.5 Closed loop usage statistics management service

The CL usage statistics management service allows configuring and controlling the CL usage statistics collection and allows sharing usage statistics with consumers of the service. Usage statistics consist in metrics and information about the CL instances such as frequency and duration of utilization, entities requesting use of the CL instances, invocations of CL instances capabilities, etc.

The CL usage statistics management service is described in Table 9.2.5-1.

Table 9.2.5-1: Service definition

Service name	Closed loop usage statistics service
External visibility	OPTIONAL
Service capabilities	
Provide available usage statistics (O)	Provide a list of the usage statistics available at one or more Closed Loop instances.
Control usage statistics collection (M)	Enable or disable the usage statistics collection of one or more Closed Loop instances. Filters may be specified.
Provide CL usage statistics (O)	Provide notifications about usage statistics of one or more Closed Loop instances. See note 3.
Query Closed Loop usage statistics (O)	Query usage statistics of one or more Closed Loop instances. See note 3.
Manage subscriptions (O)	Manage (create, read, update, delete, list) subscriptions to one or more Closed Loop instances usage statistics. See note 1.
Configure usage statistics collection (O)	Configure the Closed Loop usage statistics management service (e.g. how usage statistics should be collected and made available to consumers). See note 2.
NOTE 1: Management communication service provided in the integration fabric, as defined in ETSI GS ZSM 002 [2], could also provide this capability. NOTE 2: This capability is used to configure the closedLoopPolicy attribute of the Closed Loop model for aspects related to usage, as defined in clause 8.1.4.1. NOTE 3: At least one of these capabilities shall be supported.	

9.3 Management services for Closed Loop Coordination

9.3.1 Introduction

This clause specifies ZSM management services related to Closed Loop Coordination (see clause 8.2). The following sub-clauses contain the descriptions and the service definition tables of management services which are classified as Closed Loop Coordination (CLC) related services.

9.3.2 Pre-execution coordination service

The pre-execution coordination service is used to detect conflicting action plans provided by multiple Closed Loops in the Decision stage and select the most appropriate combination of action plans according to the evaluation. The conflict resolution may be provided by other services. This service also allows to configure control information (e.g. parameters for conflict detection, subscriptions to notifications). The pre-execution coordination service is described in Table 9.3.2-1.

Table 9.3.2-1: Service definition

Service name	Pre-execution coordination service
External visibility	OPTIONAL
Service capabilities	
Provide notifications of conflicting action plans (M)	Provide notifications of conflicting action plans obtained from different Closed Loops when a conflict has been detected in the action plans.
Get recommended action plans (O)	Obtain the combination of action plans which is expected to contribute best according to the evaluation.
Get the results of pre-action evaluation (O)	Obtain the evaluation results of comparing different action plans.
Manage subscriptions (O)	Manage (create, read, update, delete, list) subscriptions to notifications about pre-action conflicts. See note.
Configure conflict detection (O)	Configure the parameters for conflict detection (e.g. Closed Loops to be observed).
NOTE: Management communication service provided in the integration fabric, as defined in ETSI GS ZSM 002 [2], could also provide this capability.	

9.3.3 Post-execution coordination service

Coordination amongst Closed Loops may require disabling actions (actions are changes that a Closed Loop can perform over a managed entity such as configuring an attribute) of a Closed Loop. For example, such actions could be part of operational policies. For example, when two or more CLs are acting on the same managed entity it could be a good design principle to avoid both Closed Loops from executing the same actions on a managed entity in parallel. In such a case, an authorized entity could disable one Closed Loop from taking such actions using the capabilities proposed in Table 9.3.3-1.

Table 9.3.3-1: Service definition

Service name	Post-execution coordination service
External visibility	OPTIONAL
Service capabilities	
Provide list of managed entities and respective attributes modifiable by a Closed Loop (O)	Provide the attributes associated with a managed entity and the actions in relation to those attribute values that a given Closed Loop is authorized to do. This shall include actions that have been disabled.
Enable/Disable actions (M)	Provides an ability for the management service consumer to enable or disable an action of a Closed Loop. The Closed Loop is then unable to execute disabled actions in execution stage. Optionally, conditions under which the action is executed may be specified. Example of condition: network usage greater than 80 %.

Annex A (informative): Possible implementation of pause points

Figure A-1 shows the difference in interaction between stages X and stages X+1 of that belong to a CL in case of a disabled versus an enabled pause point. When the pause is disabled (steps L1 and L2) the integration fabric (IF) passes the messages as and when they are published by stage X to the stage X+1 that has subscribed to them as part of the said CL.

NOTE: This is one way of implementing pause points and is shown as an example only and assumes certain properties on the integration fabric and the relevant stages in the CL.

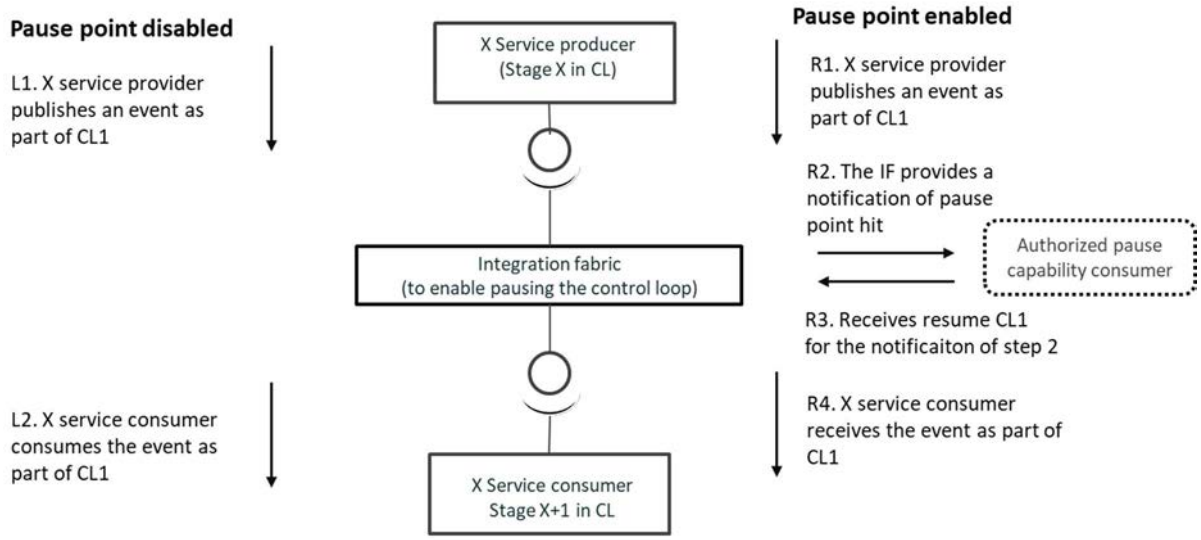

Figure A-1: Changes in interaction between stages with pause enabled

In contrast, when the pause point between two stages X and X+1 is enabled the integration fabric intercepts the messages published by stage X of the CL (step R1). Instead of passing on the message to stage X+1, the IF publishes a notification to the authorized entity that had previously enabled the pause point (step R2). When the IF receives a response to resume execution for this flow in the CL (in step R3), the IF forwards the message published by stage X in step R1 to Stage X+1 (step R4). Note that steps L1 and R1 or L2 and R4 are the same.

Annex B (informative): Change History

Date	Version	Information about changes
August 2019	0.0.1	Skeleton approved during ZSM-7d tech call
27 September 2019	0.1.1	Incorporated contributions: - ZSM(19)000461_ZSM009-1_Clause_4_Multi-part_work_item - ZSM(19)000463_ZSM009-1__Abbreviations - ZSM(19)000512r3_ZSM0091_Types_of_automation_loops__interaction - ZSM(19)000531r2_ZSM009-1__5_x_CL_Requirements_1
1 October 2019	0.1.2	Incorporated contributions: - ZSM(19)000541r3_ZSM009-1_Clauses_7_and_7_1
14 November 2019	0.2.1	Incorporated contributions: - ZSM(19)000579r2_Input_to_Clause_7_3 - ZSM(19)000464r3_ZSM009-1__Requirements_General - ZSM(19)000571r1_ZSM009-1_Requirements_on_MDs_and_CLs - ZSM(19)000591r1_ZSM009-1_Requirement_for_policy_driven_closed_loop - ZSM(19)000592r1_ZSM009-1_Requirement_for_logging_the_actions_taken_by_a_clos
18 November 2019	0.2.2	Incorporated contributions: - ZSM(19)000601r2_ZSM009-1_Clause_7_1_Adding_description - ZSM(19)000602r2_ZSM009-1_Changes_to_the_diagram
26 November 2019	0.2.3	Incorporated contributions: - ZSM(19)000504r4_ZSM009-1__5_1_General_requirements - ZSM(19)000508r2_ZSM009-1__5_1_General_requirements__additional_requirements - ZSM(19)000510r2_ZSM009-1__5_1_General_requirements__additional_requirements - ZSM(19)000597r4_ZSM009-1_Clause_6_1 - ZSM(19)000598r2_ZSM009-1_Clause_6 - ZSM(19)000600r4_ZSM009-1_Clause_7_1_1_Abstracted_view_of_CL - ZSM(19)000624r2_ZSM009-1__5_1_General_requirements
17 December 2019	0.3.1	Incorporated contributions: - ZSM(19)000490r3_ZSM009-1_General_requirements_for_closed-loop_automation - ZSM(19)000528r3_ZSM009-1__5_1_General_requirements__additional_requirement - ZSM(19)000576r3_ZSM009-1__5_1_General_requirements__additional_requirements - ZSM(19)000653r1_ZSM009-1_Aligning_requirements - ZSM(19)000654r1_ZSM009-1_Illustrating_closed_versus_open_loop (1) - ZSM(19)000655r2_ZSM009-1_Improving_Clause_6 - ZSM(19)000666_ZSM009-1_Proposed_restructuring_of_section_7
10 February 2020	0.4.1	Incorporated contributions: - ZSM(20)000033r2_Update_Clause_7_1_Closed_loop_as_an_entity - ZSM(20)000037r2_ZSM0091_-_Proposal_for_CLMnS - ZSM(20)000040r1_ZSM009-1_External_and_internal_control_of_closed_loops - ZSM(20)000046r1_ZSM009-1__3_1_Managed_entity_definition - ZSM(20)000047r3_ZSM009-1__Closed-loop_automation_enablers_-_functional_view - ZSM(20)000056r2_ZSM009-1__Closed_loop_automation_enablers_-_deployment_view
06 March 2020	0.4.2	Incorporated contributions: - ZSM(20)000082r2_Input_to_paragraph_7_2_in_ZSM009-1
06 March 2020	0.4.3	Incorporated contributions: - ZSM(20)000038r1_ZSM009-1__Interactions_based_on_intents
20 March 2020	0.5.1	Incorporated contributions: - ZSM(20)000095r1_ZM009-1_Fixes_to_Clause_7_1 - ZSM(20)000112_ZSM009-1_Remove_annexes - ZSM(20)000114_ZSM009-1_Adding_Normative_references
09 April 2020	0.5.2	Incorporated contributions: - ZSM(20)000089r3_ZSM009-1_7_3_1_update_to_define_priority_levels_of_the_CLs - ZSM(20)000097r4_ZSM009-1_Closed_loop_governance_service - ZSM(20)000121_ZSM009-1_CL_interaction_management_requirement

Date	Version	Information about changes
27 April 2020	0.5.3	Incorporated contributions: – ZSM(20)000104r3_ZSM009-1_Simplifying_clause_7_2_2
04 May 2020	0.5.4	Incorporated contributions: – ZSM(20)000096r3_ZSM009-1_Adding_back_CL_inputs
21 May 2020	0.6.1	Incorporated contributions: – ZSM(20)000156r1_ZSM009-1_Editor_s_notes_resolution_in_clause_5
27 May 2020	0.6.2	Draft 0.6.1 had a wrong table of content. This version fixes the issue.
16 June 2020	0.7.1	Incorporated contributions: – ZSM(20)000140r1_ZSM009-1_addition_to_clause_7_1_2_Deployment_View – ZSM(20)000157r2_ZSM009-1_Fixes_to_clause_7_1_1 – ZSM(20)000172r1_ZSM009-1_Resolving_editor_note_clause_7_2_1 – ZSM(20)000174_ZSM009-1_Removing_interactions_between_CLs_at_different_oper – ZSM(20)000176r1_ZSM009-1_Resolving_editor_notes_clause_8_1_1 – ZSM(20)000183r1_ZSM009-1_Closed_loop_data_and_control_flows
25 June 2020	0.7.2	Incorporated contributions: – ZSM(20)000171r5_ZSM009-1_Clause_7_2_5_LCM_of_closed_loops – ZSM(20)000199_ZSM009-1_Closed_loops_data_and_control_flows
09 July 2020	0.8.1	Incorporated contributions: – ZSM(20)000224r1_clause_7_2_CL_governance_intro – ZSM(20)000230r3_ZSM009-1_7_1_2_Update_to_define_customization_flow – ZSM(20)000246_ZSM009-1_Solving_editor_s_notes_clause_8_1_1
14 July 2020	0.8.2	Incorporated contributions: – ZSM(20)000126r2_ZSM009-1_E2E_CL_Coordination – ZSM(20)000147r1_ZSM009-1_CL_Coordination_-_Post-Action_Coordination_of_CL_Cr – ZSM(20)000148r1_ZSM009-1_CL_Coordination_-_Post-Action_Verification – ZSM(20)000264r2_ZSM009-1_Update_to_Clause_7_2_Governance_Introduction
28 July 2020	0.8.3	Incorporated contributions: – ZSM(20)000237r2_ZSM009-1__Add_MnS_for_enabling_CL_pause_point – ZSM(20)000251r3_ZSM009-1_Clause_7_2_6_Closed_loop_model – ZSM(20)000275r3_ZSM009-1_CL_model_attributes – ZSM(20)000279r1_ZSM009-1_Clause_5_1_General_requirement_for_CL_Identifier
17 August 2020	0.8.4	Incorporated contributions: – ZSM(20)000247r2_ZSM009-1_Update_of_scope – ZSM(20)000289_ZSM009-1_Resolving_ENs_Clause_4_and_5 – ZSM(20)000291r2_ZSM009-1_-_Resolving_ENs_Clause_7_2_5_1 – ZSM(20)000293_ZSM009-1_-_Resolving_ENs_Clause_8_1_1 – ZSM(20)000302_ZSM009-1_Proposal_to_restructure_the_clauses
24 August 2020	0.8.5	Incorporated contributions: – ZSM(20)000299r1_ZSM009-1_7_1_2_Update_to_define_customization_flow – ZSM(20)000313_ZSM009-1_Moving_8_2_3_to_Closed_Loop_Governance – ZSM(20)000314r2_ZSM009-1_Updating_functional_view – ZSM(20)000315_ZSM009-1_Updating_CL_Data_and_Control_Flows – ZSM(20)000316_ZSM009-1_Updating_Deployment_View – ZSM(20)000317r1_ZSM009-1_Updating_Introduction_to_CL_Automation – ZSM(20)000318_ZSM009-1_Updating_description_of_ZSM009 – ZSM(20)000319_ZSM009-1_Updating_requirements – ZSM(20)000323_ZSM009-1_Updating_clauses_on_Policy_and_Intent
10 September 2020	0.8.6	Incorporated contributions: – ZSM(20)000312_ZSM009-1_Delete_Clause_8_1_4 – ZSM(20)000331r1_ZSM009-1_Resolving_editors_note_8_1_1_and_8_1_2 – ZSM(20)000332r2_ZSM009-1_Resolving_EN_clause_8_1_3_1_3 – ZSM(20)000333r3_ZSM009-1_Closed_loop_coordination_clause
06 October 2020	0.9.2	Incorporated contributions: – ZSM(20)000363r1_ZSM009-1_Merge_Clause_8_1_3_to_Clause_7_3 – ZSM(20)000366r2_ZSM009-1_closedLoopStageClass – ZSM(20)000367r2_ZSM009-1_knowledgeClass – ZSM(20)000368r1_ZSM009-1_managedEntityClass
14 October 2020	0.10.1	Incorporated contributions: – ZSM(20)000340r3_ZSM009-1_Add_Pre-action_coordination – ZSM(20)000357r3_ZSM009-1__additions_to_clause_on_Interactions_based_on_inten – ZSM(20)000377r2_ZSM009-1_Additions_to_closed_loop_coordination_ (1)

Date	Version	Information about changes
21 October 2020	0.10.2	Incorporated contributions: – ZSM(20)000399r1_ZSM009-1_modify_CL_class
18 November 2020	0.10.3	Incorporated contributions: – ZSM(20)000342r6_ZSM009-1_Resolving_ENs_in_Closed_Loop_Models – ZSM(20)000388r3_ZSM009-1_closed_loop_stage_class_properties – ZSM(20)000389r1_ZSM009-1_knowledge_class_properties – ZSM(20)000390r1_ZSM009-1_managed_entity_class_properties – ZSM(20)000418r1_ZSM009-1_Updates_to_CL_Model – ZSM(20)000443r1_ZSM009-1_Correct_inconsistencies_in_the_Control_Flow_Descrip – ZSM(20)000453r1_ZSM009-1_Remove_EN_regarding_Time_limit – ZSM(20)000458r1_ZSM009-1_Add_example_of_pause_point_implementation
14 December 2020	0.10.4	Incorporated contributions: – ZSM(20)000413r2_ZSM009-1_CL_Types_Definition – ZSM(20)000451r2_ZSM009-1_Adding_overview_on_Architecture – ZSM(20)000455r2_ZSM009-1_Review – ZSM(20)000510r1_ZSM009-1_Update_Clause_9_to_align_with__419 – ZSM(20)000512_ZSM009-1_Legacy_CLs
27 January 2021	0.10.5	Incorporated contributions: – ZSM(20)000328r5_ZSM009-1_Add_disabling_capabaility – ZSM(20)000440r5_ZSM009-1_Add_Pre-action_coordination_MnS – ZSM(21)000020r3_ZSM009-1_Additional_capabilities_of_CL_governance_service – ZSM(21)000025r2_ZSM009-1_Add_concept_of_flows_and_chains – ZSM(21)000026r3_ZSM009-1_Add_management_of_action_plans_to_execution_service – ZSM(21)000028r1_ZSM009-1_Changes_in_clause_8_2_-_Closed_loop_coordination – ZSM(20)000446r2_ZSM009-1__5_1_General_requirements__additional_requirement
25 February 2021	0.10.6	Incorporated contributions: – ZSM(20)000444r1_ZSM009-1__Concerning_7_2_Modification_proposal_regarding_the – ZSM(20)000447r3_ZSM009-1__5_1_General_requirements__additional_requirement – ZSM(21)000044r5_ZSM009-1_CL_reporting_service
29 March 2021	0.11.1	Incorporated contributions: – ZSM(21)000016r9_ZSM009-1_review_for_final_draft – ZSM(21)000108r1_ZSM009-1_Renaming_Collection_stage_to_Monitoring – ZSM(21)000119r1_ZSM009-1_Remove_word_subordinate_and_add_goal_translation
03 May 2021	0.11.2	Incorporated contributions: – ZSM(20)000402r1_ZSM009-1_Review_Comments_for_Clause_3 – ZSM(20)000423r3_ZSM009-1_CL_Coordination_MnS_-_Interactions_identification – ZSM(20)000429r5_ZSM009-1_CL_Coordination_MnS_-_Access_and_concurrency_control – ZSM(20)000432r4_ZSM009-1_CL_Governance_MnS_-_CL_usage_management – ZSM(20)000433r4_ZSM009-1_CL_Governance_MnS_-_CL_autonomy-supervision_managem – ZSM(20)000434r2_ZSM009-1_Additions_to_CL_Model – ZSM(20)000435r5_ZSM009-1_Definitions__escalation__delegation__coordination – ZSM(21)000045r1_ZSM009-1__5_1_General_requirements__additional_requirement – ZSM(21)000165_ZSM009-1_resolve_editorial_problems – ZSM(21)000166_ZSM009-1_Editorial_clean-up_from_EditHelp
10 May 2021	0.11.3	Incorporated contributions: – ZSM(21)000173r1_ZSM009-1_Editorial_fixes_for_final_draft

History

Document history		
V1.1.1	June 2021	Publication