# ETSI TS 128 567 V19.0.0 (2025-10)

TECHNICAL SPECIFICATION

**5G;**
**Management and orchestration;**
**Management Aspects of Closed Control Loops**
**(3GPP TS 28.567 version 19.0.0 Release 19)**

Reference

DTS/TSGS-0528567vj00

Keywords

5G

# 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.

# Contents

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

| | |
|---|---|
| **shall** | indicates a mandatory requirement to do something |
| **shall not** | indicates an interdiction (prohibition) to do something |

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

| | |
|---|---|
| **should** | indicates a recommendation to do something |
| **should not** | indicates a recommendation not to do something |
| **may** | indicates permission to do something |
| **need not** | indicates permission not to do something |

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

| | |
|---|---|
| **can** | indicates that something is possible |
| **cannot** | indicates that something is impossible |

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

| | |
|---|---|
| **will** | indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document |
| **will not** | indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document |
| **might** | indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document |

| | |
|---|---|
| **might not** | indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document |

In addition:

| | |
|---|---|
| **is** | (or any other verb in the indicative mood) indicates a statement of fact |
| **is not** | (or any other negative verb in the indicative mood) indicates a statement of fact |

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document describes, concepts and background, and specifies use cases and requirements, information models and procedures for use, control, conflict management and coordination of Closed control loops in network management.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 28.535: "Management and orchestration; Management services for communication service assurance; Requirements".

[3] 3GPP TS 28.536: "Management and orchestration; Management services for communication service assurance; Stage 2 and stage 3".

[4] 3GPP TS 28.111: "Technical Specification Group Services and System Aspects; Management and orchestration; Fault Management (FM)".

[5] 3GPP TS 28.622: "Technical Specification Group Services and System Aspects; Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)".

[6] 3GPP TS 28.572: "Management and orchestration; Management of planned configurations".

[7] 3GPP TS 28.104: "Management and orchestration; Management Data Analytics (MDA)"

[8] 3GPP TS 28.625: "Telecommunication management; State management data definition Integration Reference Point (IRP); Information Service (IS)"

[9] ITU-T X.731: "Information technology - Open Systems Interconnection - Systems management: State management function "

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

**Closed Control Loop**: A management function that monitors and controls a set of managed entities, and operates without any intervention from a human operator or any other management entity other than possibly the initial configuration.

## 3.2 Symbols

Void.

## 3.3 Abbreviations

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

| | |
|---|---|
| CCL | Closed Control Loop |

# 4 Concepts and overview

## 4.1 Closed Control Loops

A Closed Control Loop (CCL) is a type of control mechanism that monitors and regulates a set of managed entities with the objective of achieving specific requirements. A CCL can be logically decomposed into several stages or steps, each providing a specific functionality and where the steps work together to achieve the stated requirement. Any two CCLs may have the same functionality but supported in different count of steps, i.e. the steps implement differing functional capabilities although together, each set achieves the same functionality. Similarly, any two CCLs with the same functionality and same count of steps, the respective steps may not have the same functionality.

A control loop is a building block for management of networks and services. The basic principle of any control loop is to adjust the value of an observed variable (expressed as for example an attribute) to control/influence the value of another attribute for a controlled entity, such as a managed entity or managed function. The producers of the measurements or observations services, analysis services and control service, are all required to fully realize and use a control loop.

A Closed Control Loop (CCL) is a control loop which operates without any intervention from a human operator or any other management entity other than possibly the initial configuration of the measurement producer and configuration of the control loop. In a closed control loop the input to the control loop provided by human operator or other management entity may include the requirementsor policies. Besides the provisioning needed to realize the requirement, the output of the closed control loop may also include closed control loop status to a human operator or other management entity.

Examples of well-known Closed Loop types are OODA loop, composed of 4 stages/steps (Observe, Orient, Decide, Act) and MAPE, also composed of 4 stages/steps (Monitor, Analyse, Plan, Execute).

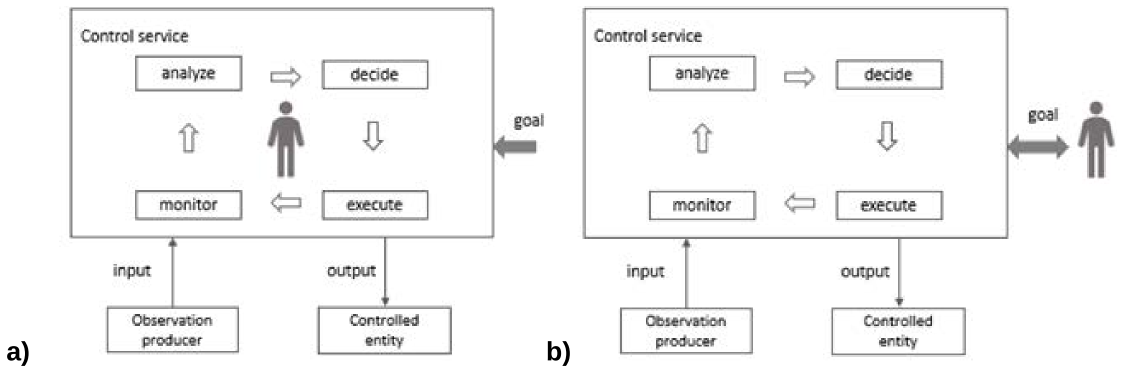


**Figure 4.1-1: Open control loop entities versus Closed control loop entities (see TS28.535[2])**

## 4.2 Functional steps of a closed control loop

A closed control loop may manage any managed entity, e.g. a network resource or a communication service as described in TS 28.535 [2] and TS 28.536 [3]. Generally, the control loop consists of the steps Monitoring/data collection, Analysis, Decision and Execution. The adjustment of the resources of the managed entity used is completed by the continuous iteration of the steps in a management control loop.

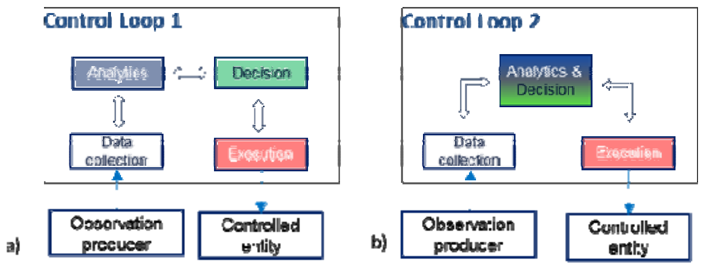


**Figure 4.2-1: Steps of a Control Loop**
**a) the four functional steps; and**
**b) 2 steps combined into a single management function**

- The "Monitoring/data collection" step is responsible for collecting and pre-processing data from managed entities or from external sources.
- The "Analysis" step derives insights from the available data obtained in the monitoring/data collection step. The insights provide answers to the questions, for example What is likely to happen, what has happened and why.
- The "Decision" step is responsible for deriving workflows from insights provided by the analysis step. It decides which reactive, proactive or predictive actions should be taken in consideration of insights obtained in the analysis step.
- The execution step manages the activation of commands on the controlled resources or entities. The decision step should decide which actions are required, but not necessarily how they should be taken in the managed entities. So, the translation from actions to commands is a responsibility of the execution step.

## 4.3 Characteristic information of a CCL

### 4.3.1 Overview

A CCL is associated with a set of Characteristic information that describes its properties, behaviours and impact. This information includes CCL Goals, CCL Triggers, CCL actions and action plans as well as CCL scopes.

### 4.3.2 CCL requirements

The CCL is responsible for a set of outcomes that need to be achieved or realized by the CCL. Each expected outcome is called a CCL requirement. A CCL may have one more CCL goals each containing a set of CCL targets requirement contains for example a metric.

### 4.3.3 CCL actions and actions plans

A CCL action is a configuration change that a CCL can perform over a managed entity such as configuring an attribute of managed object. A CCL may decide to perform several actions which can be combined in a single CCL action plan.

### 4.3.4 CCL scopes

The scope is the set of managed objects, their properties and network outcomes that are associated with the CCL for measurement, configuration and impact. The scopes for the different CCLs can be managed by the MnS consumer, i.e. they can be defined on to the CCL or revised by the MnS consumer. A CCL may have four scopes: the measurement scope, target (impact) scope, control scope and impact scope, defined as follows:

- measurement scope: the measurement scope is where related measurements are collected- control scope: control scope is the scope to which the CCL's actions are desired to be applied, e.g., the set of network functions and attributes that are the planned candidates to be modified by the CCL. The control scope is also called the action-space as it describes the set of candidate actions that the CCL can (is configured to be able to) execute.

- targeted scope: which relates to purpose of the CCL
- desired impact scope: the scope to which the CCL's actions are desired to have influence, e.g., it is both the network functions and attributes as well network outcomes like coverage areas that are planned to be influenced by the configuration’s actions of the CCL.
- control scope: control scope is the scope on which the CCL executes actions, e.g., the set of managed objects which the CCL configures
- Monitored scope: Monitored scope is the scope which a CCL monitors to see if there are conflicts.
- impact scope: impact scope is the scope to which the CCL's actions have influence, e.g., it is both the network functions and attributes as well network outcomes like coverage areas that are influenced by the configuration actions of the CCL. This is different from the measurement scope, i.e. the scope where the CCLs measure and control scope, i.e. the scope where they act.

  The impact scope may be known and bounded or unbounded and thus unknown - see figure 4.3.4-1. The bounded scope indicates that the area known by the CCL is the scope where its actions will impact. The unbounded impact-scope is the full network scope where the CCL’s action will have impact, but the CCL does have information that its action will have that impact to that scope.

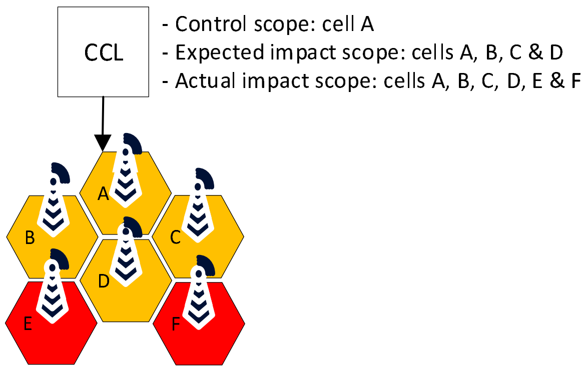

CCL
- Control scope: cell A
- Expected impact scope: cells A, B, C & D
- Actual impact scope: cells A, B, C, D, E & F
A
B
C
D
E
F

**Figure 4.3.4-1: Exemplification of known/bounded vs. unknown/unbounded impact scope: CCL A takes action in cell A expecting impact in cells A, B, C and D. if the impact is strictly in cells A, B, C and D, then the impact scope is known and bounded. However, if the impact scope includes cells E and F, then for the CCL, the true impact scope is unknown and thus unbounded.**

## 4.4 Closed control loops purposes and uses

CCLs automate the management of network resources thereby taking control away from operators. TS 28.535 [2], TS 28.536 [3] describe the use of CCLs for service assurance. Additionally, CCLs may be used for several other use cases including among others use of CCLs for problem analysis and for fault management.

In each case, the CCL can be viewed as an entity to be managed, management capabilities related to the CCL will be exposed by the MnS producer that is associated with the CCL to enable the MnS consumer to manage the CCL. E.g., the characteristics and behaviours of the CCLs need to be directed by operators as consumers of CCL-related management services. Moreover, the MnS also exposes capabilities for coordinating the CCL's activities. Example capabilities include capability for providing information on conflict resolution and feedback on monitoring the impact of the CCL's actions on other closed control loops or management functions.

The 3GPP management system should provide capabilities that enable a consumer to:

- Manage the execution of CCLs. E.g. to request for and be notified about the instantiation of CCLs. For instance, if the consumer wants to request for instantiation of an Energy saving CCL for 10,000 cells.

- Compose or request for and be notified about the composition of a CCL from a set of specific components (such as analytics services or SON functions).
- Manage a closed loop composed from multiple components.

## 4.5 Realisation of closed control loops

The CCL may be composed on demand into what is called an open-box CCL from discrete entities in the Management system that accomplish the functional steps, e.g., using capabilities offered by the Management system. The CCLs steps (or the components accomplishing them) as well as the interactions among them, are based on 3GPP management services. E.g., the CCL may apply one or more PM jobs for the Monitoring/data collection step and apply an MDA capability for an analysis stage.

However, the CCL (e.g., in any form in Figure 4.2-1) can be pre-integrated into what are called Closed-Box CCLs. where, the CCLs components and their interactions are assembled prior to instantiating the CCL in the Management system. They are not accessible to the MnS consumers, only the CCL's external interactions and capabilities, e.g., the control interface to define the CCL scopes is accessible to the MnS consumers.

## 4.6 Closed Control Loops conflicts

Multiple CCLs could co-exist and concurrently act within the same environment. The CCLs can affect one another, in the worst cases leading to conflicts. The conflicts may occur among desired outcomes, scopes or actions of the CCLs. The possible conflict scenarios include:

- CCL outcomes conflicts
- CCL Scope conflicts
- CCL-Trigger-time:
- CCL actions conflicts, with 2 subtypes – concurrent and non-concurrent actions conflicts.
- CCL metric-value conflicts: with 2 subtypes – concurrent and non-concurrent metric-value conflicts.

# 5 Management capabilities

## 5.1 Dynamic control and composition of CCLs - DynCCL

### 5.1.1 Description

CCLs may be dynamically realized. There are two aspects to dynamically realization of CCLs - dynamic instantiation of a CCL from an existing template and dynamically composing the CCL from discrete components based e.g. on the provided requirements.

The trigger functionality describes the need of CCL instaitation and execution of CCL actions based on specific triggers.

The information about the CCL that existed in the past need to be maintained as historical CCL information. The historical CCL information may be utilized in the creation of new CCL.

### 5.1.2 Use cases

#### 5.1.2.1 General CCL Control – DynCCL_01

A CCL contains a set of logic functionalities or steps, each providing a specific functionality and where the steps work together to achieve the stated desired outcomes over a given network scope. The MnS consumer should be able to configure and receive  information about the desired outcomes of the CCL.

Generally, the four CCL steps of Monitoring, Analysis, Decision and Execution are expected with the expectation that each step is accomplished by a single management function or service. However, one management function or service may also accomplish the functionality of more than 1 step. The MnS consumer should be able to receive information about the management functions or services that form the CCL.

A CCL may have four scopes including a desired outcomes scope, measurement scope, a control scope and an impact scope. The scopes for the different CCLs can be managed by the MnS consumer. The MnS consumer should also be able to receive reports about these different aspects of the CCL, e.g., about the status of the CCLs execution as well as to configure the reporting.

#### 5.1.2.2 Composing a CCL from discrete components – DynCCL_02

A CCL may be composed from steps provided by different management functions or management services. i.e. the CCLs is assembled on demand by MnS consumers, using capabilities offered by the Management system, e.g. from independent management functions. The CCLs components, as well as the communication and interoperation between components, are based the different 3GPP management services. Accordingly, the MnS consumer should be able to identify and indicate the MnFs or MnS producers that should be used to compose a CCL. Moreover, the MnS consumer may indicate towards the MnS producer the request to compose the CL of a particular type (e.g. for optimizing energy efficiency) without requiring to state the specific components that should be used.

Two approaches are possible:

- Composition from management Functions: Different management functions may be used to realize the different steps of a closed loop, for example, an MDA function may realize the analytics step of the CCL while another management function may realize the decision step of the CCL.

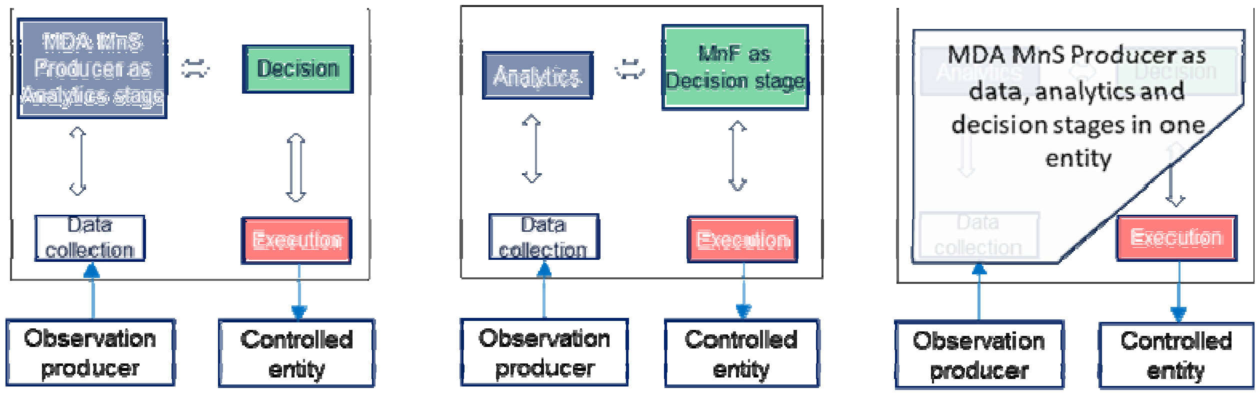


**Figure 5.1.2.2-1: Management functions as steps of a closed control loop**

- Composition from management services: Different management services may be used to realize the different steps of a closed loop, i.e. the management service provides the output expected from a specific step.

EXAMPLE: A capability of the MDA MnS realizes an analytics step of the CCL while another capability may realize a specific data collection step of the CCL.

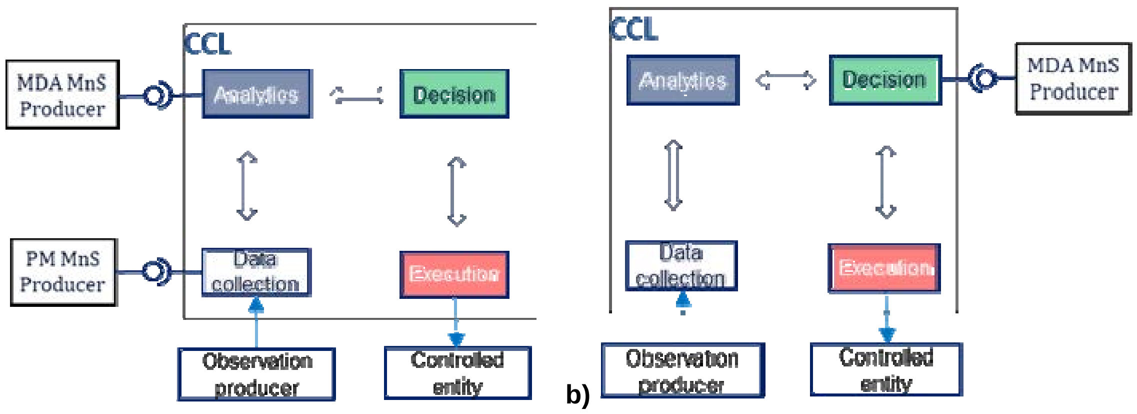


**Figure 5.1.2.2-2: Management services used as implementations of CCL steps:**
**a) MDA MnS and PM job the respective implementations of the analysis and data collection steps;**
**and b) MDA MnS as the implementation of the decision step**

The MnS consumer should be enabled to control the composition of such a CCL. The MnS consumer could request for and be notified about the composition of a CCL from a set of specific components (i.e. specific management functions or management services). The request could indicate components with specific given capabilities (such as analytics services with specific analytics types) which should be combined to achieve the closed loop. Moreover, the request could be for composition of a CCL required to achieve a specific set of desired outcomes or requirements.

### 5.1.2.3 Triggered CCL instantiation based on conditions – DynCCL _03

The MnS consumer may want to request for a CCL to be instantiated not immediately but when certain conditions are met. For example, the MnS consumer may want that for a CCL of a stated type or that matches a set of stated characteristics (e.g. goal) to be instantiated under certain conditions and another with variations in goals to be instantiated under other conditions. The MnS consumer should be enabled to define those conditions so that the CCL is instantiated when the stated conditions are met. The MnS Producer monitors the conditions to check if they are met.

The conditions can be related to events based on management data (e.g., performance, fault, configuration).

Performance events are defined related with performance measurements and KPIs that need to be monitored by the producer to see if an CCL is to be initiated. For example, if the value of a particular performance measurement goes beyond a particular value, a CCL should be instantiated to keep the value of the same performance measurement below a defined value.

Fault events are defined by related information (e.g alarm type, alarm severity) that need to be monitored by the producer to see if a CCL is to be initiated. For example, if the total number of alarm with type `QUALITY_OF_SERVICE_ALARM` and `perceivedSeverity` `MAJOR` goes beyond a particular value, a CCL should be instantiated.

Provisioning events (e.g `CreateMOI`) are defined that need to be subscribed by the producer to see if a CCL is to be instantiated. For example, the creation on an Intent MOI can be a trigger to instantiate a CCL. For another example, when a pre-defined system time specified by operators can be a trigger to instantiated a CCL.

NOTE: The use case requires to set the conditions and then the conditions need to be continuously monitored and tracked.

### 5.1.2.4 CCL creation based on Historical CCL data capability – DynCCL -04

This use case describes the need of maintaining information about the CCLs that existed in the past. Those CCLs are called Historical CCLs.

In an automation environment, before a consumer request to create a CCL it would like to know the data related with Historical CCLs that were available with the producer. This information will enable consumer to request for an optimal CCL. The information about historical CCL may include, scope of the CCL, configured goals/targets, controlled entity, etc.

Further, Historical CCL information serves as a valuable data source for predictive analytics within the CCL system executed as Analytics step. It enables the system to move from a reactive mode, where it responds to current issues, to a proactive mode, where it anticipates and prevents problems based on historical trends and patterns. This proactive approach enhances network reliability, minimizes downtime, and improves the overall efficiency of network operations.

The Historical CCL information may be used by the management system to setup or initialize a CCL. The Historical CCL information provides the profiles of a CCL for CCL at different hierarchies. For example, CCLs that do not do coordination which are at a lower hierarchy L and CCLs responsible for coordination (as coordination entities) which are at a higher hierarchy H. For a new CCL at a lower hierarchy, the management system obtains the profiles of the several CCLs at different hierarchies and correlates the information of the new CCL (e.g. its goal information) against the profiles of the CCLs at the different hierarchies. Based on this, the management system computes the complete profile of the new CCL (including e.g. its measurement and control scope) which is then configured onto the new CCL.

### 5.1.2.5 Triggered CCL action execution based on conditions – DynCCL _05

For the CCLs that have been instantiated, the MnS consumer may want to define conditions under which a CCL may execute actions on the network, e.g. when the performance on a certain threshold is crossed, or when the confidence on the decision is above a stated threshold. The consumer does not need to be aware of all decisions, but by providing conditions under which decisions may be activated or not, it is able to have supervision over the CCL without having to continuously track the decisions. The MnS consumer should be enabled to define those conditions for executing the CCL actions. Otherwise, the consumer should be enabled to define alternative actions, e.g. to notify the consumer of the decision that is not executed.

By supporting this, the execution can be affected by producer based on consumer's conditions or requirements. To ensure oversight or accountability by the MnS consumer, the MnS consumer may be notified by the CCL about the executed action for any conditionally execution. The MnS consumer can intervene as needed.

NOTE: The use case requires to set the conditions and then the conditions need to be continuously monitored and tracked.

### 5.1.3 Requirements

**Table 5.1.3-1**

| Requirement label | Description | Related use case(s)/ Motivation |
|---|---|---|
| REQ-DynCCL_01-01 | The CCL MnS Producer should support a capability to provide information to the MnS consumer about the management functions and services that make up the CCL and where applicable the functionality accomplished by the components. | UC-DynCCL_01 Clause 5.1.2.1 |
| REQ-DynCCL_01-02 | The CCL MnS Producer should support a capability enabling the MnS consumer to configure and receive information on status of execution of the CCL. | UC-DynCCL_01 Clause 5.1.2.1 |
| REQ-DynCCL_01-03 | The CCL MnS Producer should support a capability enabling the MnS consumer to configure and receive information on the scopes of the CCL | UC-DynCCL_01 Clause 5.1.2.1 |
| REQ-DynCCL_01-04 | The CCL MnS Producer should support a capability to report to the MnS consumer about the CCL | UC-DynCCL_01 Clause 5.1.2.1 |
| REQ-DynCCL_02-01 | The CCL MnS Producer should support a capability enabling the MnS consumer to request for a CCL (instance) to be composed from a set of management function types or instances or management services | UC-DynCCL_02 Clause 5.1.2.2 |
| REQ-DynCCL_02-01 | The MnS producer for CCL management should support a capability enabling the MnS consumer to request that a CCL of a specific type or fulfilling a stated requirementsshould be composed from a set of management function types or instances or services | UC-DynCCL_02 Clause 5.1.2.2 |
| REQ-DynCCL_02-01 | The MnS producer for CCL management should support a capability enabling the MnS consumer to provide conditions under which a CCL can be dynamically composed or instantiated triggered to execute | UC-DynCCL_02 Clause 5.1.2.2 |
| REQ-DynCCL_02-01 | The MnS producer for CCL management should support a capability enabling the MnS consumer to be notified when a CCL is dynamically composed or instantiated or triggered to execute | UC-DynCCL_02 Clause 5.1.2.2 |
| REQ-DynCCL_03-01 | The 3GPP management system should enable authorized MnS consumer to request for information (e.g. CCL identification, configured goals/targets and the related status, scope of the CCL, conflict information) related with Historical CCL. | UC-DynCCL_03 Clause 5.1.2.3 |
| REQ-DynCCL_03-02 | The 3GPP management system shall have the capability to configure the profile of a CCL based on the historical CCL information that describes the profile of other CCLs at different hierarchies. | UC-DynCCL_03 Clause 5.1.2.3 |
| REQ-DynCCL_04-01 | The 3GPP management system should enable authorized consumers to define conditions related to performance, fault and configuration data that can be monitored and used to trigger CCL instantiation. | UC-DynCCL_04 Clause 5.1.2.4 |
| REQ-DynCCL_04-02 | The 3GPP management system should enable authorized consumers to define conditions related to performance, fault and configuration data that can be monitored and used to trigger CCL update. | UC-DynCCL_04 Clause 5.1.2.4 |
| REQ-DynCCL_05-03 | The 3GPP management system should enable authorized consumers to define conditions related to performance, fault and configuration data that can be monitored and used to trigger CCL deletion. | UC-DynCCL_05 Clause 5.1.2.5 |
| REQ-DynCCL_05-01 | The 3GPP management system should enable authorized consumers to define conditions related to performance, fault and configuration data under which a CCL may execute its actions. | UC-DynCCL_05 Clause 5.1.2.5 |

## 5.2 CCL Performance Monitoring - CCLPERF

### 5.2.1 Description

CCL performance need to be assured in order to ensure efficient CCL based network management.

### 5.2.2 Use Cases

#### 5.2.2.1 Performance Evaluation of a Closed Control Loop – CCLPERF_01

The advanced monitoring functionalities of a CCL can provide real-time insights into the performance and outcomes of a CCL. The monitoring activity for a Closed Control Loop may result in further actions that happen in the operation phase, e.g., evaluate and update, in order to change the closed control loop settings and improve its performance. So, there is a need to evaluate the performance of a Closed Control Loop itself. Such metrics are important to understand and change a CCL's behaviour and to improve its performance to pursue the assigned requirement(s).

For example, certain performance aspects of a CCL can be very crucial to know in order to evaluate and decide upon a CCL's performance, such as the number of breached requirements, time taken to meet a breached requirement, number of conflicts occurred by a CCL etc. With the knowledge of such performance aspects of an existing CCL, a MnS consumer can more effectively update or create a new CCL.

An operator can also compare different CCLs based on these performances and choose the best one for its network deployment.

#### 5.2.2.2 MnS Consumer's feedback on CCL actions – CCLPERF_02

A CCL should derive its actions without the involvement of any other entity (such as the managed network object) but the actions can have different levels of satisfaction for the different MnS consumers. The MnS consumers should be able to provide feedback to the CCL indicating how satisfied the MnS consumer is with the quality of the CCL actions, which should enable the CCL to fine-tune and optimize its decisions.

EXAMPLE: The MnS consumer feedback may grade the usefulness of the executed action on a fixed scale say from 0 (indicating a terrible and never to be re-used action) to 10 (indicating a very good action for the interests of the MnS consumer). Other criteria may be added, e.g., to address the case that two consumer experience the same outcomes but may have different grade for feedback,

To be able to gauge the satisfaction, the MnS consumer should be able to receive information about the provisioning operations executed by the CCL. This information includes operation performed, MOIs updated, etc. The CCL does not break its execution when it provides information to the MnS consumer or to wait for feedback from the MnS consumer. The feedback from an MnS consumer does not break the loop.

It may be needed to determine what impact the CCLs' action(s) had on a given scope that is the responsibility of other CCLs. Based on the CCL actions and the resulting impact on PMs, it may be determined that new actions are needed to undo the degradation and to avoid it in future.

Based on some local policies or due to degradations observed, the consumer may prefer that a particular NF is not updated as part of the Execution step of CCL. The consumer should be enabled to request the CCL to revoke the changes made to a NF. Consumer may also update the CCL to ensure that a particular NF is never updated in future.

#### 5.2.2.3 Assessment and resolution of CCL Impact on unknown impact-scope - CCLPERF_03

For some CCLs, the impact-scope affected by the actions of a CCL A may not be known a priori. For example, when a CCL A adjusts transmit power (e.g. to minimize interference), the neighbour cells and related CCLs acting on those cells that would be affected by any transmit power decrease or increase cannot be explicitly enumerated. Any negative effects cannot be easily anticipated, and most may not be easy to resolve by if-then-else rules. Instead, the affected CCLs should report their observed negative or positive impacts to CCL A to determine how to resolve the impact or avoid them in future.

'Related CCLs need to be notified that CCL A has executed an action and the impact-time of the action, i.e. the maximum time within which the action is expected to have impact and at which an observed impacts should be reported. For example, the impact of load balancing is visible in a few seconds while the impact of a handover decision can take several minutes. After the notified impact time, the impacted CCLs need to report the impact that CCL A had to their performance metrics . The impact may be reported an index say in the range [0,10] where 0 implies an unacceptable action and 10 implies a good action. CCL A can then derive an appropriate remediation, e.g. by reconfiguring the candidate actions of the acting CCL (i.e. CCL A) or by undoing the action.

### 5.2.3 Requirements

**Table 5.3.3-1**

| Requirement label | Description | Related use case(s)/ Motivation |
|---|---|---|
| REQ-CCLPERF_01 -01 | The 3GPP management system should be able to obtain a CCL's performance with respect to the total number of occurrences of a requirement breach. | UC-CCLPERF_01 Clause 5.3.2.1 |
| REQ-CCLPERF_01-02 | The 3GPP management system should be able to obtain a CCL's performance with respect to the time taken by CCL to meet a breached requirement. | UC-CCLPERF_01 Clause 5.3.2.1 |
| REQ-CCLPERF_01-03 | The 3GPP management system should be able to obtain a CCL's performance with respect to the total number of conflicts occurred by a CCL | UC-CCLPERF_01 Clause 5.3.2.1 |
| REQ-CCLPERF_02-01 | The 3GPP management system should enable MnS consumer to provide its feedback on the action(s) taken by CCL. | UC-CCLPERF_01 Clause 5.2.2.2 |
| REQ- CCLPERF_02-02 | The 3GPP management system should enable MnS consumer to request for revocation of the action(s) taken by the CCL. | UC-CCLPERF_01 Clause 5.2.2.2 |
| REQ-CCLPERF_02-03 | The 3GPP management system should have a capability enabling the MnS consumer to receive information (e.g. operation performed, MOIs updated) about the action(s) taken by a CCL A. | UC-CCLPERF_01 Clause 5.2.2.1 UC-CCLPERF_02 Clause 5.2.2.2 |
| REQ-CCLPERF_03-01 | The 3GPP management system should support a capability enabling an MnS consumer to receive a report containing an executed action and the impact that the action had to a particular impact-scope. | UC-CCLPERF_02 Clause 5.2.2.2 UC-CCLPERF_03 Clause 5.2.2.3 |
| REQ-CCLPERF_03-02 | The 3GPP management system should support a capability enabling an MnS consumer to propose to a CCL a remediation against the noted impact of a CCLs' actions, e.g. the reconfiguration of the candidate actions of the CCL. | UC-CCLPERF_03 Clause 5.2.2.3 |

## 5.3 Closed Control Loops usage scenarios - CCLUSE

### 5.3.1 Description

Closed control loops can be used for different purposes or scenarios Two example scenarios are fault management and network performance problem recovery.

### 5.3.2 Use Cases

#### 5.3.2.1 Closed Control Loops for fault management – CCLUSE_01

This use case describes a scenario in which an MnS consumer may request a CCL for fault management. The consumer may request to identify the root cause of the fault and take actions to mitigate and/or resolve the root cause for a given list of alarms. Furthermore, the request may include fault management policies and actions specified by an MnS consumer in order to mitigate and/or resolve the root causes for the given alarms.

Based on the request, a CCL may take action to further enhance the correlation of alarms, for example, correlation of alarms with change in PM/KPIs and/or fault supervision events to find solutions to mitigate and/or resolve the identified root causes. In addition, a fault management CCL may clear the alarms that otherwise have to be manually cleared by the MnS consumer, which are defined as ADMC Alarms in TS 28.111.

The MnS producer reports the result of fault management. The report may include information on the status of each alarm, such as any identified root cause and correlation information as well as indication of the successful mitigation and/or resolution of root causes for any given alarm.

#### 5.3.2.2 Closed Control Loops for network performance problem recovery CCLUSE_02

Based on the concept in 3GPP TS 28.104 [7], MDA reports may contain root cause analysis of ongoing issues, predictions of potential issues and corresponding relevant causes and recommended actions for preventions, and/or prediction of network and/or service demands. For example:

- MDA for Coverage problem analysis can provide the following information in the MDA report:
    - coverageProblemId;
    - coverageProblemType;
    - coverageProblemAreas; and
    - recommendedActions.

The MnS consumer may decide to use CCLs to resolve the observed performance problems based on the analytics reports (e.g. provided by MDA) and other management data (e.g. historical decisions made previously) if necessary. It can be possible that one MnF is responsible for network performance problem observation and recovery, while another MnF is responsible for the decision on whether the network performance problem needs to be resolved. In this scenario, The MnF for decision can decide whether it needs the other MnF to recover from the observed network performance problems (e.g. coverage problem) based on MDA report (e.g. root cause information, recommended solutions) and other information (e.g. user experience information, information from other domains). If it decides to recover the observed network performance problems, The MnF for decision needs to request another MnF to recover the specified network performance problems observed from the MDA report. MnS consumer may specifies the network scope and time window for network performance problem recovery, which means the MnS producer needs to recover the problem at the specified time window for the network scope. During problem recovery phase, as process for network performance problem recovery is complex and time-consuming, the MnS consumer needs to obtain the progress of the recovery process. When the last step of the network performance problem process is completed, MnS producer needs to send the result of this network performance problem recovery process to the MnS consumers.

If a closed control loop instance can be used to resolve network performance problem, the MnS consumer may need to know the result of resolving the network performance problem by the closed control loop instance, including the network performance problems which are resolved by the closed control loop as well as network performance problem resolution statistics (e.g. the number of network problem resolved by the closed control loop in the specified period).

### 5.3.3 Requirements

**Table 5.4.3-1**

| Requirement label | Description | Related use case(s)/ Motivation |
|---|---|---|
| REQ-CCLUSE_01-01 | The 3GPP management system shall have the capability to allow MnS consumer to request a closed control loop for fault management | UC-CCLUSE_01 |
| REQ-CCLUSE_01-02 | The 3GPP management system shall have the capability to allow MnS consumer to get a report from the closed control loop regarding the status of fault management | UC-CCLUSE_01 |
| REQ-CCLUSE_02-01 | The 3GPP management system should have the capability to allow the MnS consumer to request a CCL for resolving the network performance problems. | UC-CCLUSE_02 Closed Control Loops for network performance problem recovery |
| REQ-CCLUSE_02-02 | The 3GPP management system should have the capability to allow the MnS consumer to obtain the result of network performance problem resolved by the closed control loop. | UC-CCLUSE_02 Closed Control Loops for network performance problem recovery |
| REQ-CCLUSE_02-03 | The 3GPP management system should have the capability to allow the MnS consumer to obtain the progress information of network performance problem recovery. | UC-CCLUSE_02 Closed Control Loops for network performance problem recovery |
| REQ-CCLUSE_02-04 | The 3GPP management system should have the capability to allow the MnS consumer to configure a CCL with the network scope and time window to be monitored for resolving the network performance problems. | UC-CCLUSE_02 Closed Control Loops for network performance problem recovery |

## 5.4 CCL Conflict management and coordination Capability - CONF

### 5.4.1 Description

#### 5.4.1.1 Overview

A CCL may experience direct conflicts on its requirements, targets, scopes, trigger time and execution time. The management system needs to support capabilities to avoid, detect and resolve the conflicts.

The possible conflict scenarios are defined as follows:

- **CCL Scope conflicts:** These are conflicts among the scopes of the CCLs, specifically the scenarios where a given scope is considered differently by distinct CCL instances. An example is where the measurement scope of one CCL is the control scope of another CCL. Where applicable, it is desirable that the scopes are allocated such that that one CCL instance does not read a scope that is concurrently being controlled or adjusted by another CCL. These also include conflict among the desired outcomes of the individual CCLs sharing a given scope.

- **CCL-Trigger-time**: These are conflicts for the times at which the CCLs are triggered to derive and activate action plans. , For example, Energy saving decisions may impact handovers so the triggers should be set such that Energy saving is not triggered after handover optimization, e.g., to ensure that handover optimization does not read a measurement scope that changes after reading it.

- **CCL actions conflicts:** These are conflicts among the actions of the CCLs, specifically the scenarios where two CCL instances attempt to differently control the same parameters of the same managed objects. Where applicable, it is desirable that the actions are decided and allowed such that that two CCL instances will not control or adjust the same set of parameters on the same set of managed objects.

There are 2 subtypes of CCL actions conflicts – concurrent and non-concurrent actions conflicts.

- **CCL concurrent actions conflicts:** These are conflicts where the actions are executed within a time period less than the impact time of the action, i.e., the action of the second CCL instance is executed before the impact of the first CCL instance is registered. In the simplest scenario, the two CCL instances try to execute the contradictory actions at exactly the same time. Concurrent actions conflicts are also called "action-execution-time conflicts"

- **CCL non-concurrent actions conflicts:** These are conflicts where the actions are executed within a time period longer than the impact time of the action, i.e., the action of the second CCL instance is executed after the impact of the first CCL instance is registered. The second CCL instance in effect tries to undo the impact of the CCL instance.

- **CCL metric-value conflicts:** These are conflicts for the desired value of one or more performance metrics by two CCL instances that do not have conflicts for desired outcomes on stated scopes or actions. The two CCL instances which have different desired outcome and two distinct control and measurement scopes but the actions of one CCL instance have impact on the measurement scope of the other CCL instance, i.e. one CCL's actions will indirectly affect the network performance metrics that the other CCL is responsible for. For example, a conflict could occur among the metrics if a CCL that optimizes energy consumption affects handover performance metrics which are supposed to be optimized by another CCL.

There are 2 subtypes of CCL metric-value conflicts – concurrent and non-concurrent metric-value conflicts.

- **CCL concurrent metric-value conflicts**: These are metric-values conflicts between CCLs with close trigger times, i.e., where the CCL instances are triggered to act concurrently or to execute actions within the same time.

- **CCL non-concurrent metric-value conflicts**: These are conflicts where the CCL instances are triggered to act in different time periods, e.g. where one CCL instance is active while the other is only monitoring its measurement scope.

### 5.4.1.2 Example conflicts

Examples characterizing the differences among the conflicts are summarized by Table 5.4.1-1.

**Table 5.4.1-1: Types of potential conflicts among CCL instances for desired outcome g1, g2 and g3**

| Conflict Type | Description | CCL-A | CCL-B | Comments |
|---|---|---|---|---|
| Scope conflict | For CCLs CCL-A and CCL-B, CCL-A and CCL-B have different desired outcomes and actions but their scopes are overlapping - e.g. CCL-A's control scope (i.e. the controlled entities in the network) is part of CCL-B's measurement scope (i.e. the measured entities in the network). | Measurement scope:<br>- cells g1<br>Control Scope:<br>- g1<br>Desired outcome:<br>- EC/bit is < 1WA<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: switch off g1 | Measurement scope:<br>cells g1, g2, g3, g4<br>Control Scope:<br>- g2<br>Desired outcomes:<br>- Load < 80 %<br><br>Actions:<br>- Entity: gNB-g2<br>- Change: change CIO | By switching off g2, CCL-A affects the scope which CCL-B reads for its load distribution measurements |
| Trigger-time | For CCLs CCL-A and CCL-B, CCL-A and CCL-B have different related desired outcomes, actions or scopes - e.g. CCL-A's impact scope is part of CCL-B's measurement scope, so their triggers can cause clashes. | Measurement scope:<br>- cells g1<br>Control Scope:<br>- g1<br>Desired outcome: optimize Energy consumption<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: switch off g1 | Measurement scope:<br>cells g2, g3,<br>Control Scope:<br>- g2<br>Desired outcomes:<br>optimize handovers<br><br>Actions:<br>- Entity: gNB-g2<br>- Change: change CIO | By switching off g1, CCL-A affects handover measurements in g2 measured and controlled by CCL B |

| Conflict Type | Description | CCL-A | CCL-B | Comments |
|---|---|---|---|---|
| Action Conflict | **Concurrent direct actions conflicts:**<br>For CCLs CCL-A and CCL-B, when both CCL-A and CCL-B are trying to configure the same characteristics of same entity (gNB-g1) in contradiction, the actions executed within a short time period e.g. less than the impact period of their actions | expected outcomes:<br>- Throughput > 10 Gbps<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: scale-out<br>- Time: 04:00 | expected outcomes:<br>- EC is < 10KVA<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: scale-in<br>- Time: 04:00 | Conflict due to the time of executing the configuration actions on the same scope at the execution step |
| | **Non-concurrent direct actions conflicts:**<br>For CCLs CCL-A and CCL-B, when both CCL-A and CCL-B is trying to configure the same characteristics of same entity (gNB-g1) in contradiction, the actions far apart from each other; e.g. in a time period longer than the impact period of their actions | **Example 1** | | Conflict due to configuration actions at execution step because both CCL want contradicting values for a particular characteristic of gNB-g1.<br><br>Effect: the value may ping-pong continuously. |
| | | expected outcomes:<br>- Throughput > 10 Gbps<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: scale-out virtual resource | expected outcomes:<br>- EC is < 10 KVA<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: scale-in virtual resource | |
| | | **Example 2** | | |
| | | expected outcome:<br>- HO failure is < 2 %<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: set CIO to a small **positive** value{to guarantee HOs with low chances of HO failure} | expected outcome:<br>- Load < 80 %<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: set CIO to a small negative value [to advance HOs and move load to other cells] | |
| Metric-value conflict | **CCL concurrent metric-value conflicts:** For CCLs CCL-A and CCL-B, when CCL-A [optimize handover] and CCL-B [minimize interference] have different desired outcomes but are executed within a short time intervals between each other and the actions of CCL-A affect the desired outcomes of CCL-B. | expected outcome:<br>- HO failure is < 2 %<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: reduce CIO {to reduce chances of HO failure} | expected outcome:<br>- SINR > 10 dB<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: lower antenna tilt | By reducing antenna tilt to minimize interference CCL-B affect the HO desired outcomes that are being optimized by CCL-A |
| | **CCL non-concurrent metric-value conflicts:**<br>For CCLs CCL-A and CCL-B, when CCL-A [optimize handover] and CCL-B [minimize interference] have different desired outcomes but are executed far apart from each other but the actions of CCL-A affect the desired outcomes of CCL-B. | expected outcome:<br>- HO failure is < 2 %<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: reduce CIO {to reduce chances of HO failure} | expected outcome:<br>- SINR > 10 dB<br><br>Actions:<br>- Entity: gNB-g1<br>- Change: lower antenna tilt | By reducing antenna tilt to minimize interference CCL-B affect the HO outcomes that are assumed optimal and stable by CCL-A |

The CCL may detect or observe events that identify the possibility of any one of the above conflicts. The conflict can be avoided using information or the policies (e.g. priority) provided by the consumer. The respective information is described in the use cases below. If the conflict actually occurs, the CCL MnS producer should support services to inform MnS consumers the confirmed detected conflicts. This may also include informing MnS consumer about the potential conflict.

### 5.4.1.3 Alternative CCL coordination Approaches for conflicts handling

To address the conflicts, coordination interactions are required either among CCLs or between the CCLs and one or more higher hierarchy coordination functions to avoid or detect and resolve the conflicts. This is required when CCL are actuating in the same set of resources.

The coordination of CCLs could be accomplished via one of three approaches illustrated by Figure 5.4.1.3-1:

- Distributed coordination with distributed execution (Figure 5.4.1.3-1 a), where the CCLs directly coordinate with one another, and each manages execution of its decisions. The CCL exchange information with each other avoid, detect or resolve conflicts. The information may for example include notifications of executed actions or observed impacts.

- Hierarchical coordination with distributed execution (Figure 5.4.1.3-1 b), where the CCLs coordinate through a separate coordination layer, say via a CCL coordination entity, but each manages execution of its coordinated decisions. The CCL exchange information with the CCL coordination entity to avoid, detect or resolve conflicts. A CCL may send notifications of its executed actions or observed impacts which the CCL coordination entity may relay to other CCLs. The CCL coordination entity may configure the CCLs but each CCL executes its action based the CCL coordination entity’s configuration.

- Hierarchical coordination and execution (Figure 5.4.1.3-1 c), where the CCLs coordinate through a separate coordination layer, say via a coordination entity that besides coordination also manages execution of the coordinated decisions. The CCL exchange information with the CCL coordination entity to avoid, detect or resolve conflicts including notifications of their executed actions or observed impacts which the CCL coordination entity may relay to other CCLs. The CCL coordination entity may configure the CCLs and the CCL execute their actions through the CCL coordination entity.

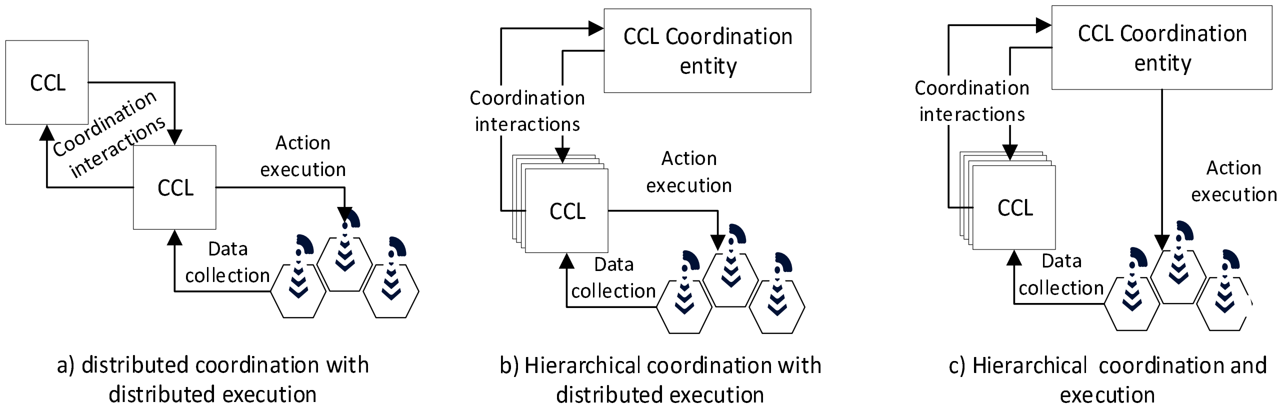

CCL
Coordination interactions
CCL
Action execution
Data collection
a) distributed coordination with distributed execution
CCL Coordination entity
Coordination interactions
CCL
Action execution
Data collection
b) Hierarchical coordination with distributed execution
CCL Coordination entity
Coordination interactions
CCL
Action execution
Data collection
c) Hierarchical coordination and execution

**Figure 5.4.1.3-1: Closed Control Loop Coordination approaches**

Distributed coordination can lead to too many exchanges between the CCLs which may unnecessarily clog the system. On the other hand, "Hierarchical coordination and execution" implies that too much responsibility is concentrated in a single CCL. A desired behavior is that the individual CCLs are responsible for their own decision execution, so it is recommended that to follow the "hierarchical coordination with distributed execution" approach. In this approach, the CCLs are responsible for making their decisions and executing actions, but they coordinate with the CCL coordinator before, during or after execution.

### 5.4.1.4 Hierarchical CCL-coordination-interactions for conflicts handling

To address the conflicts, coordination interactions are required between the CCLs and one or more higher hierarchy coordination functions to avoid or detect and resolve the conflicts among requirements, control scopes or actions of the CCLs. The 3GPP management system includes at least one entity called the Coordination entity that undertakes the role of CCL coordination. The Coordination entity can be implemented as a CCL, an AIML inference engine or any other functionality that is found appropriate. The coordination entity may support coordination for conflict management for different conflicts described in clause 5.4.2; scope conflicts, CCL-Trigger-time and CCL-action-execution-time conflicts, Direct actions conflicts as well as metric-value conflicts.

The coordination of CCLs could be required at different execution points of the CCL translating into different CCL coordination use cases with corresponding CCL coordination services required at those points as illustrated by example Figure 5.4.1.4-1. The coordination of CCLs could be achieved via direct interaction among the CCLs or via a third-party entity, say called the CCLs coordination Function (or simply CCL Coordinator).

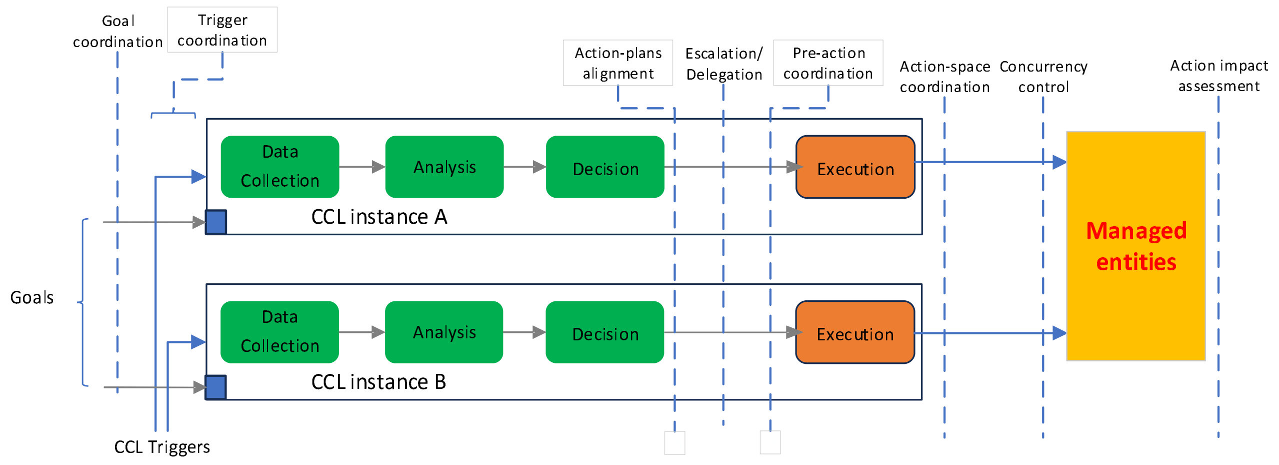


**Figure 5.4.1.4-1: Exemplary Closed Control Loop Coordination interaction points**

NOTE: The terms at the top indicate general naming of the groupings of coordination interactions at the different execution points during the execution of the CCL. Action-space coordination implies coordinating the sets of actions that the different CCL can apply. Concurrency control implies coordinating the times at which different CCLs can execute actions. Action-impact assessment indicates interactions and processes on the evaluation of the impacts of the different CCLs.

The coordination purpose attributes contain the information and data needed or used by the coordination entity for interacting with the CCL when handling conflicts.

## 5.4.2 Use Cases

### 5.4.2.1 CCL scope conflicts handling – CONF_01

Each CCL should have specific scopes for which it is responsible. The network may be assumed to be a muti--dimensional space, with say n dimensions, i.e., the network has full scope S of n dimensions including, e.g., time, geography, etc. A CCL is assigned a sub scope D that is only a portion of the network's scope (illustrated by Table 5.4.2.1-1). Scope assignment is the mapping of CCLs to sub scopes S that are part of the network's full scope. A scope conflict occurs if the scope assigned to a CCL overlaps in an undesirable way with another scope assigned to another CCL. The 3GPP management system should support the capability to coordinate the scope assignment to enable detection and avoidance of potential scope conflicts. The 3GPP management system should also support the capability to coordinate the outcomes desirable for the different scopes to enable detection, avoidance and resolution of conflicts on the CCL's outcomes for those scopes. It may be desirable to define the full scope space S and a set of scope rules to be used to derive the best scope to be assigned to each CCL. An example rule may be that the defined CCL scope should not overlap. The rules may for example be defined by an operator or can be implementation specific depending on the types of CCLs that are to be configured.

**Table 5.4.2.1-1: Example of a network scope-space from which the scope of CCL may be derived**

| Scope dimension | Granularity | Example values to be assigned |
|---|---|---|
| Time | Seconds, minutes, days | Every hour,<br>Every Saturday at 2:00 hours |
| Network domains | | Radio<br>Core |
| Geography | Region/City | City x<br>Street y in City x |
| Network Elements | gNB | gNB X |
| | Cells | Cell A on gNB X |
| | Terminals, e.g. types of users | users |
| Resources | Slices | |
| | Network Function | Virtual Network Function A<br>Physical Network Function B |
| | Transport containers (links, flows, etc.) | an identifiable link,<br>a specific flow |
| Purpose | The purpose of the CCL | Coverage, Performance, Energy Efficiency, Fault Management, UE specific mobility |

NOTE: Table 5.4.2.1-1 is not complete and can be improved and/or extended as needed. Scope conflicts are only considered actual if the application of the defined scopes results in negative outcomes. The management system should support the capability to coordinate the scope assignment to detect and resolve actual scope conflicts. The CCLs monitor changes in their scope. If the scope is changed, it is desirable for the CCLs to notify the scope assignment MnS consumer of the changes or differences between what was configured and the actual scopes. The scope assignment MnS consumer may then trigger scope conflict evaluation based on the actual scope.

#### 5.4.2.2 CCL trigger conflicts handling - CONF_02

Typically, a CCL whose start is triggered based on conditions, needs to be triggered to run at a specific time and terminate when certain conditions are met, to run when a certain performance threshold is crossed. If triggered independently, there may be conflicts among the CCLs. The triggers for different CCLs to be executed need to be coordinated to avoid conflicts among the CCLs. The triggers for execution of different CCLs need to be coordinated to avoid conflicts among the CCLs.

In some instances, the conditions in the network may be such that it is not clear which CCL should be triggered, requiring to trigger multiple CCL in sequence. The triggering may be done by a coordination function that consumes the CCL-related information with which to evaluate the conditions and determines which CCL to be triggered. The CCL coordination entity evaluates network data and analytics to identify the nature of the problem and best CCLs to be scheduled at specific times to address the problem but without their execution conflicting with one another.

It may be the case that CCLs need to operate in a hierarchy with each CCL having an operational profile indicating the specific level of hierarchy. The operational profile describes characteristic sunder which the CCL operates, e.g. when or after which other CCLs, this CCL should be executed. For example, to ensure that handovers are always optimal, a CCL on handover optimization may need to be triggered every after a CCL on Energy saving has been executed to be sure that there are appropriate handover relations even when some cells may have been disabled. The CCL may be involved in more than 1 hierarchies or in a single hierarchy, the CCL may relate to multiple other CCLs in one or more domains. This requires the hierarchies to be coordinated. The CoordinationEntity obtains the operational profiles of the CCLs, evaluates the correlation among them to set the appropriate hierarchies for triggering the CCLs. The MnS consumer that coordinates the execution times of the CCLs needs to configure the appropriate hierarchy for the CCLs.

#### 5.4.2.3 CCL Concurrent actions conflicts handling - CONF_03

Several CCLs may want to execute actions onto the network. It may not be desirable that their actions are executed within the same time frame. For example, if executed so close to one another, their effects will be super-imposed and neither CCL can identify the effect of its actions on the network.

The management system should support the capability for detection of potential concurrent actions conflicts. A coordination entity  acting as a supervisory action-critic oversees the actions of the different CCLs may need to receive information enabling the detection of such conflicts. The action-critic functionality takes the responsibility for the end-to-end performance across several CCLs enabling evaluation of cases when the actions of multiple CCLs collide.

For a given CCL, the MnS consumer may need to receive the recommended changes from the CCLs, to evaluate them and see if they overlap with other proposed changes from other CCLs. Where there are likely conflicts and expected undesired impacts, the MnS consumer may propose to the CCLs, the changes that should be undertaken to minimize concurrent changes on the same network resources. The MnS consumer may need to provide feedback to the CCL instance (s) regarding their recommended actions.

In some instances, the conditions in the network may be such that it is not clear which CCL should be triggered, requiring to trigger multiple CCL in sequence. The CCLs may operate in a hierarchy with each CCL having an operational profile indicating the specific level of hierarchy. The MnS consumer that coordinates the execution times of the CCLs needs to configure the appropriate hierarchy for the CCLs. The triggering by a coordination capability based on information from the CCL allows resolution of CCL Concurrent actions conflicts.

## 5.4.2.4 CCL non-concurrent actions conflicts handling –CONF_04

When two (or more) CCLs attempt to adjust the same network parameter but with different and contradicting values, the desired actions of the 2 CCL will be in conflict. For example, a CCL assuring throughput of a slice may be scaling-out the virtual resources of the slice. Whereas a CCL minimizing the energy consumption may be scaling-in the virtual resource of the same slice. It can be when the CCLs execute actions at the same time. However, it also happens when the CCLs execute at different times, and the scenario for actions to be separated in time is the more likely than actions occurring simultaneously. casein these conflict scenarios, the network parameter continuously ping-pongs between the two values. Such a conflict may be called an action conflict.

NOTE: A potential conflict can for example be detected if a CCL observed that PMs on a certain object keep flipping between two values. The constant flipping can be an indication that 2 CCL instances are attempting to change the same scope.

The CCL may detect or observe events that identify the conflicts. The conflict can be avoided using some information or the policies (e.g. priority) provided by the consumer. If the conflict actually occurs, the CCL MnS producer should support services to inform MnS consumers the confirmed detected conflicts. It is needed to maximize the avoidance of conflict, including “requesting” information from MnS consumer and to inform MnS consumer about the potential conflict. CCL MnS Producer may also provide recommendations, for updating/deleting the conflicting CCLs, that would result in the resolution of detected conflict. The recommendation for update may include suggestions for modified requirements.

Note: The exact information that can be exchanged is not specified in this document

## 5.4.2.5 CCL concurrent metric-value conflicts handling – CONF_05

Two (or more) CCLs configuring different control parameter may all influence the same metric. In other cases, the two CCLs influence two metrics Y1 and Y2 that are coupled, i.e., which have a logical relationship between them. E.g. handover (HO) failure and SINR are coupled since a bad SINR can lead to more HO failures. If the two CLs desire different values for the metric, or different values for two metrics Y1 and Y2 but the requirements are coupled, the CCLs are in conflict for the metric resulting into a metric-value conflict. The concurrent metric-value conflict is observed from oscillations in the metrics.

Two metrics Y1 and Y2 may be coupled such that actions to optimize any of them lead to correlated oscillations/degradations in Y1 or Y2, e.g. Y1 ensuring "HO failure is < 2 %" and Y2 wanting "SINR > 10dB". The correlated oscillations indicate a potential conflict, but the CCLs may not see the oscillations in the metric that is not of their interest. The management system should support the capability for detecting potential metric-value conflicts. An MnS consumer may analyse the correlations to detect the potential conflict between CCL1 and CCL2. The MnS consumer should be able to inform CCL1 and CCL2 about the detected potential conflict represented by the correlated oscillations.

This severity of degradation in the performance metrics of the related CCLs could be the confirmation that a detected potential conflict is an actual harmful conflict. The management system should support the capability for detecting or confirming actual metric-value conflicts. The threshold to determine the severity may be defined by the MnS consumer (e.g. the operator) so that if the degree of degradation is higher than the threshold then it is a confirmed conflict that requires resolution.

The management system should support the capability for avoiding potential concurrent metric-value conflicts. CCLs need to avoid large and frequent changes to network parameters which may affect network stability since they increase the probability of occurrence of conflicts. CCLs should take small smooth changes in the cases where the impact is not so clear and only make the large changes when the CCL is sure that the impact is positive. It is desirable for the CCL to notify to the MnS consumer the planned change, its claimed/predicted performance improvement and reliability/confidence in that action/decision. The MnS consumer may evaluate the claimed performance improvement and reliability/confidence to determine if the action should be allowed or not. The MnS consumer should be enabled notify the decision and possibly the failed criteria to the CCL - to either be executed or to be used to compute better decisions. Based on the inputs, the CCL may update its decision-making and repeat the decision evaluation process. If the CCL has consistently made good large-action-decisions, the MnS consumer should be enabled to inform the CCL that the CCL has consistently made good decisions and achieved its ultimate trust and that no more coordination of its decisions is needed.

The management system should support the capability for resolving detected metric-value conflict. The MnS consumer should be enabled to trigger one or more CCLs to respond to the detected potential conflict. And if the triggered CCLs is unable to resolve that conflict, the CCL should inform the MnS consumer about the failure to resolve the problem. The MnS consumer can set the thresholds for performance degradation that triggers conflict detection and resolution.

Note : The criteria for accurately setting the thresholds for performance degradation is not specified in this document.

#### 5.4.2.6 CCL non-concurrent metric-value conflicts handling - CONF_06

Two (or more) CCLs configuring different control parameter may all influence the same metric. If the two CL desire different values for the metric, the CCLs are in conflict for the metric resulting into a metric-value conflict. In effect the actions of the two CCLs are in conflict but indirectly since they are conflicting for the same control parameter but their impacts are conflict on the desired value of the metric or target. Such conflicts are metric-value conflicts and if their actions are far enough apart that their effects cannot be related to one another, they are non-concurrent metric-value conflicts.

The management system should support the capability for avoidance of concurrent metric-value conflicts. Since each CCL focuses on a smaller scope of the network problem space, several CCLs may need to be executed. For actions in a given network scope, the CCLs can be explicitly scheduled by the management system. Where the scopes overlap, the CCLs need to align the action plans, for example, which action plan to execute and when. There is a need to assess each plan and choose the most appropriate combination of action plan(s) based on the selection policy and then notify the selected action plan(s) to the related CCLs. The MnS consumer may also be notified when it is safe to ignore the conflict. The MnS consumer may configure the criteria for evaluating the severity of conflicts.

For a detected metric-values conflict, the coordinator CCL can trigger one or more CCLs to respond to the detected potential conflict. If the CCLs that has been requested to resolve potential conflict is unable to resolve that conflict, the CCL should inform the CCL coordination MnS producer about the failure to resolve the problem.

#### 5.4.2.7 Coordinating CCLs with other management functions – CONF_07

A CCL can make and execute decisions in different network contexts and for different network functions and parameters. Yet within the network, there may be other management functions or features including MDA functions, SON functions, and AIML Functions, which also make decisions that affect the same network functions and parameters as the CCL. The operation of CCLs needs to be coordinated with the other management functions.

NOTE 1: This use-case only focuses on coordinating CCLs with other management functions for executing decisions.

For a given context, the CCL should indicate the set of network functions and corresponding parameters which it is interested in changing. Accordingly, the MnS consumer, say responsible for coordinating the CCLs with management functions may subscribe to be notified of changes on network functions and parameters. The MnS consumer should be able to inform the CCL of the latest changes to a network function or its parameter and a management entity/function (e.g. CCL, MDA, SON, AI/ML inference Function) responsible for the change to the parameter.

The CCL may want to obtain the history of previous values of the parameter. The history includes, for each previous value, the identifier of a respective management entity/function responsible for that change to the parameter. The CCL may define a favourable range of values of the parameter based on the received information on the latest change and the history of previous changes to the parameter. The CCL can calculate a new value of the parameter considering the favourable range as a constraint for the new value. The CCL needs then to update the value of the parameter of the network function to the new value.

NOTE 2: The MnS consumer may for example be the functionality that is responsible for coordinating CCL and other management functions.

## 5.4.3 Requirements

**Table 5.4.3-1**

| Requirement label | Description | Related use case(s) |
|---|---|---|
| **REQ-CONF_01-01** | The 3GPP Management System should support a capability to detect and inform an authorized MnS consumer about a potential or actual CCL scope conflicts. | **CONF_01** |
| **REQ-CONF_01-02** | The 3GPP Management System should support a capability to confirm a potential CCL scope conflict as an actual CCL scope conflict and inform an authorized MnS consumer about a confirmed actual CCL scope conflict. | **CONF_01** |
| **REQ-CONF_01-03** | The 3GPP Management System should support a capability to avoid or resolve a CCL scope conflict that has been detected | **CONF_01** |
| **REQ-CONF_01-04** | The 3GPP Management System should support a capability to coordinate the resolution of CCL scope conflicts among multiple CCLs | **CONF_01** |
| **REQ-CONF_02-01** | The 3GPP Management System should support a capability to trigger execution of CCLs according to defined hierarchies | **CONF_02** |
| **REQ-CONF_02-02** | The 3GPP Management System should support a capability to detect and inform an authorized MnS consumer about a potential or actual CCL trigger-time conflicts. | **CONF_02** |
| **REQ-CONF_02-03** | The 3GPP Management System should support a capability to confirm and inform an authorized MnS consumer about a detected CCL trigger-time conflict after it is confirmed. | **CONF_02** |
| **REQ-CONF_02-04** | The 3GPP Management System should enable authorized MnS Consumer to provide information that can be used to support a capability to avoid or resolve a CCL trigger-time conflict. | **CONF_02** |
| **REQ- CONF_03-01** | The 3GPP Management System should support a capability to detect and inform an authorized MnS consumer about a potential CCL concurrent actions conflict. | **CONF_03** |
| **REQ-CONF_03-02** | The 3GPP Management System should support a capability to confirm a potential CCL concurrent actions conflict as an actual conflict and inform an authorized MnS consumer about the confirmed actual CCL concurrent actions . | **CONF_03** |
| **REQ-CONF_03-03** | The 3GPP Management System should support a capability to avoid or resolve a CCL concurrent actions conflict that has been detected | **CONF_03** |
| **REQ-CONF_03-04** | The 3GPP Management System should support a capability enabling the MnS consumer to configure a hierarchy of a CCL | **CONF_04** |

| Requirement label | Description | Related use case(s) |
|---|---|---|
| **REQ-CONF_04-01** | The 3GPP Management System should support a capability to detect and inform an authorized MnS consumer about a potential action conflict. | **CONF_04** |
| **REQ-CONF_04-02** | The 3GPP Management System should support a capability to confirm and inform an authorized MnS consumer about an actual action conflict. | **CONF_04** |
| **REQ-CONF_04-03** | The 3GPP Management System should enable authorized MnS consumers to provide information that can be used to resolve a CCL action conflict. | **CONF_04** |
| **REQ-CONF_04-04** | The 3GPP Management System should enable authorized MnS consumers to provide information that can be used to avoid the action conflict. | **CONF_04** |
| **REQ-CONF_05-05** | The 3GPP Management System should support a capability to coordinate the resolution of CCL action conflicts among multiple CCLs | **CONF_05** |
| **REQ-CONF_06-01** | The 3GPP Management System should support a capability to detect and inform an authorized MnS consumer about a potential or actual CCL Metric-value conflicts. | **CONF_06** |
| **REQ-CONF_06-02** | The 3GPP Management System should support a capability to confirm and inform an authorized MnS consumer about a detected CCL Metric-value conflict after it is confirmed. | **CONF_06** |
| **REQ-CONF_06-03** | The 3GPP Management System should support a capability to avoid or resolve a CCL Metric-value conflict that has been detected | **CONF_06** |
| **REQ-CONF_07-01** | The CCL MnS producer should have a capability to indicate to an MnS consumer the set of network functions including their parameters which it is interested in changing | **CONF_07** |
| **REQ-CONF_07-02** | The management system should have a capability enabling an authorized CCL instance acting as MnS consumer to receive information on the latest changes to a network function parameter and an identifier of a management entity/function including MDA Function, a SON Function or an AI/ML inference Function that responsible for the change to the parameter. | **CONF_07** |
| **REQ-CONF_07-03** | The management system should have a capability enabling an authorized MnS consumer to receive the history of previous values of the parameter, including, for each previous value, the identifier of a respective management entity/function responsible for that change to the parameter. | **CONF_07** |

## 5.5 CCL decision escalation – ESC

### 5.5.1 Description

This use case related to the capability to escalate decision making to another entity e.g. another CCL.

### 5.5.2 Use Cases

#### 5.5.2.1 Triggering CCL decision escalation – ESC_01

Not all decisions made by CCLs in different network contexts (states, status, conditions, etc.) are equally effective. The CCL may need to inform another entity about its lack of confidence in its decision with a request to escalate its decision making to that entity. For example, a CCL for optimizing energy saving may fail to decide the sequence in which cells may be deactivated when there is a failure for some cells. The CCL may escalate the scenario to a CCL on problem recovery.

The MnS consumer should be able to configure MnS producer regarding the escalation recipient to which the decision is escalated. The degree to which the CCL can independently execute decisions or escalates them, should be configurable by the MnS consumer through a confidence threshold. The confidence threshold is an index on a fixed scale say from 0 (indicating lowest confidence) to 10 (indicating highest confidence). It could be configured based on the sensitivity of the operations under the CCLs' control, the trust level in the decisions of the CCL and the necessity to consider a bigger picture at times. Then, based on how much confidence the CCL has in its decisions, the CCL can escalate a decision or situation to an escalation recipient (e.g. another CCL or a CCL coordination entity) which has this bigger picture (say has wider scope), can execute a different(larger) set of actions or has better capabilities, e.g. a larger and more capable ML model.

NOTE: The computation of confidence within the CCL is up to implementation as it depends on the CCL's purpose and the scenario that the CCL is addressing. The escalation recipient CCL should enable the escalator CCL to request for escalation for a given network context or state with e.g. information about the escalator CCL preferences and observed constraints when driving decisions. Based on its evaluations, the escalation recipient CCL should provide to the escalator CCL a report that holds the outcomes for a given escalation request.

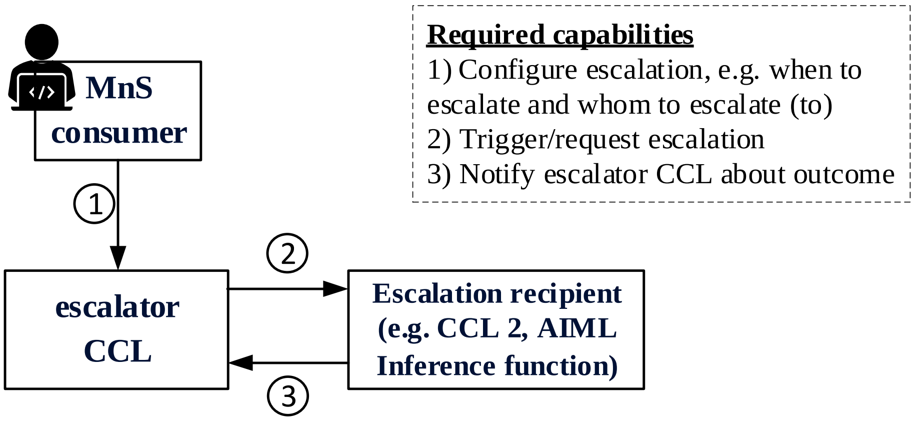

Required capabilities
1) Configure escalation, e.g. when to escalate and whom to escalate (to)
2) Trigger/request escalation
3) Notify escalator CCL about outcome
MnS consumer
1
2
3
escalator CCL
Escalation recipient (e.g. CCL 2, AIML Inference function)

**Figure 5.5.2.1-1: required interactions for CCL decision escalation**

### 5.5.3 Requirements

**Table 5.5.3-1**

| Requirement label | Description | Related use case(s) |
|---|---|---|
| **REQ-ESC_01-01** | The 3GPP management system should have a capability to enabling an authorized MnS consumer to configure a CCL with the degree of autonomy of to define when the CCL can escalate and the entity to which to escalate decision making. | **UC-ESC_01 Clause 5.5.2.1** |
| **REQ-ESC_01-02** | The 3GPP management system should have a capability to enabling an authorized MnS consumer (e.g. an escalator CCL) to request to escalate decision-making for a network context or state to an escalation recipient. | **UC-ESC_01 Clause 5.5.2.1** |
| **REQ-ESC_01-03** | The 3GPP management system should have a capability enabling an escalation recipient CCL to report to an authorized MnS consumer (e.g. an escalator CCL) the outcomes for a given escalation request | **UC-ESC_01 Clause 5.5.2.1** |

# 6 Model

## 6.1 Imported information entities and local labels

| | |
|---|---|
| 3GPP TS 28.622 [6], `DataType, TimeWindow` | `TimeWindow` |
| 3GPP TS 28.622 [6], `DataType, DateTime` | `DateTime` |

## 6.2 Class diagram

### 6.2.1 Relationships

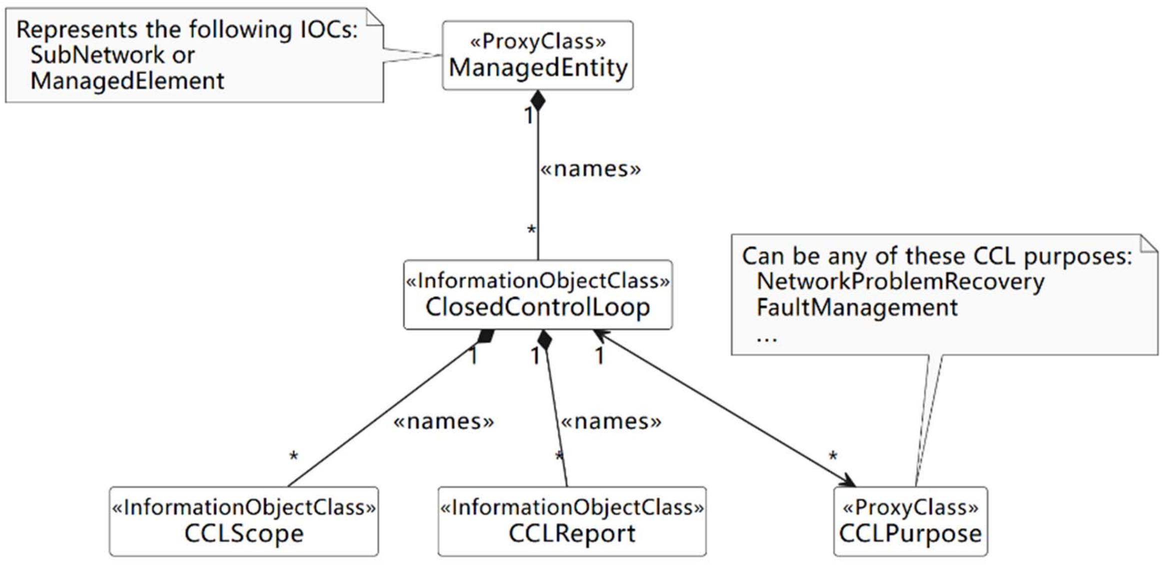


Figure 6.2.1-1: Relations for common information models for CCL management

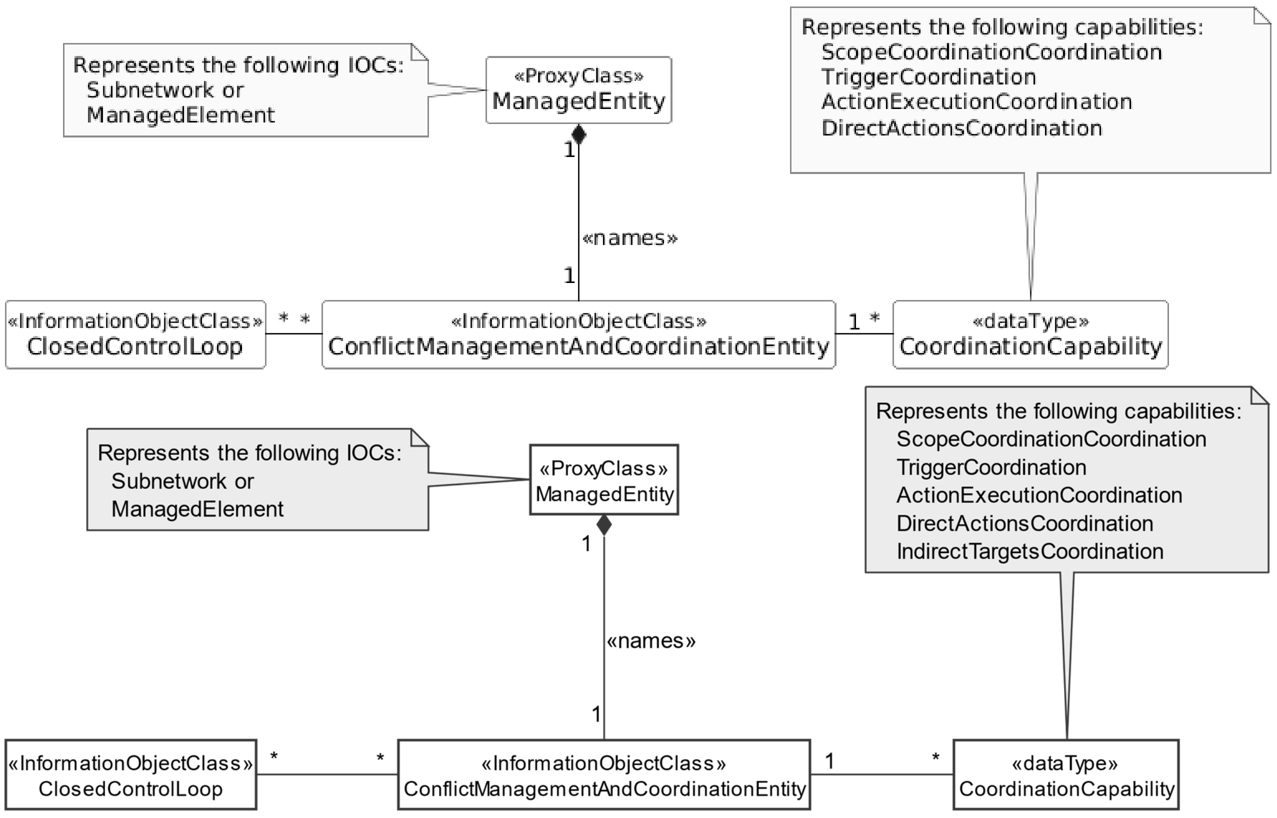

Represents the following IOCs:
Subnetwork or
ManagedElement
«ProxyClass»
ManagedEntity
Represents the following capabilities:
ScopeCoordinationCoordination
TriggerCoordination
ActionExecutionCoordination
DirectActionsCoordination
1
«names»
1
«InformationObjectClass»
ClosedControlLoop
* *
«InformationObjectClass»
ConflictManagementAndCoordinationEntity
1 *
«dataType»
CoordinationCapability
Represents the following IOCs:
Subnetwork or
ManagedElement
«ProxyClass»
ManagedEntity
Represents the following capabilities:
ScopeCoordinationCoordination
TriggerCoordination
ActionExecutionCoordination
DirectActionsCoordination
IndirectTargetsCoordination
1
«names»
1
«InformationObjectClass»
ClosedControlLoop
*
*
«InformationObjectClass»
ConflictManagementAndCoordinationEntity
1
*
«dataType»
CoordinationCapability

Figure 6.2.1-2: NRM fragment for conflict management and Coordination entity

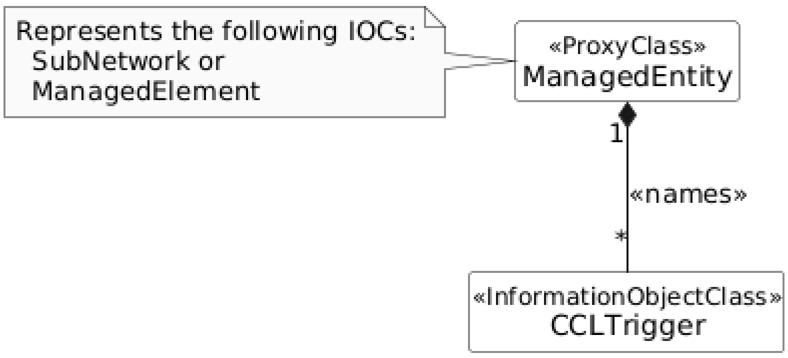


Figure 6.2.1-3: NRM fragment for CCLTrigger

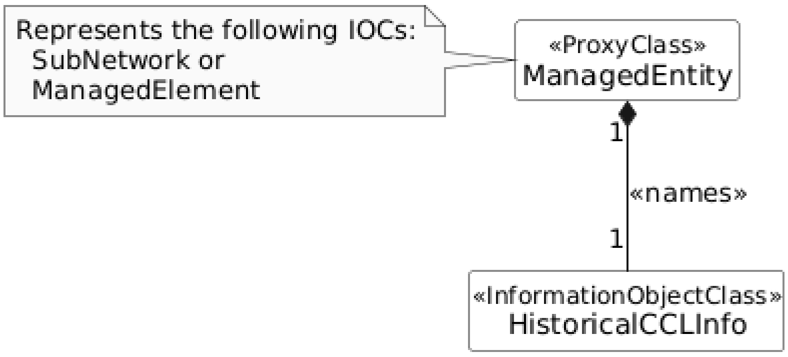
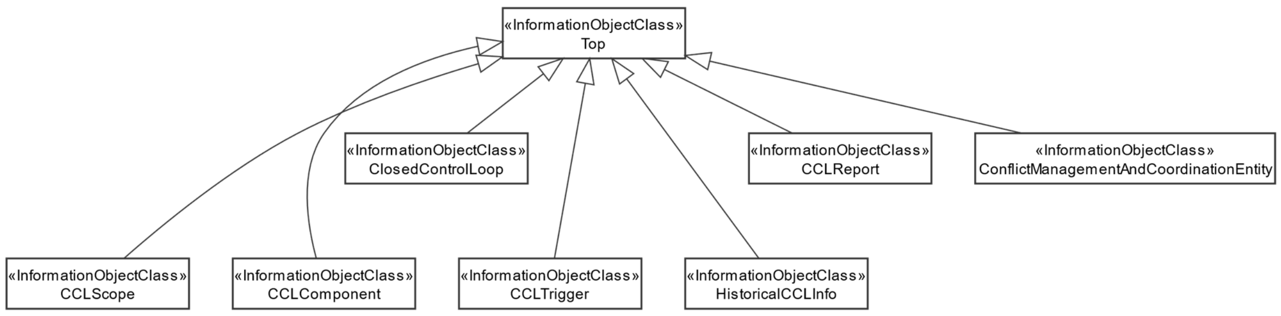


Figure 6.2.1-4x: NRM fragment for Historical CCL

## 6.2.2 Inheritance



FFigure 6.2.2-1: Inheritance Hierarchy for Closed Control Loops and for conflict management and Coordination entity

## 6.3 Class definitions

### 6.3.1 ClosedControlLoop

#### 6.3.1.1 Definition

This IOC represents the closed control loop. It represents the information for controlling and monitoring a CCL associated with a stated scope.

The `ClosedControlLoop` is name-contained by `SubNetwork` or `ManagedElement` and is associated with a CCLreport that contains reported information about the CCL. Accordingly, the report about a CCL can exist even when the CCL is deleted.

The capabilities of the CCL are contained in one or more `CCLPurposes` that describe what the CCL is capable of doing or can be configured to do - including information about the network resources for which the CCL can execute decisions and actions. So, the `ClosedControlLoop` is associated with one or more `CCLPurpose`(s) that indicate(s) a list of characteristics that describe what a CCL can/is expected to be able to do. The purpose describes the type of functionality that can be executed including problem recovery and fault management .

The operational information about the CCL is contained in the `CCLScope`(s), so the `ClosedControlLoop` is associated with one or more `CCLScope`(s). The `CCLScope` defines what the CCL has been configured to read, evaluate, control, etc.

A CCL can be created from several components that are dynamically composed from a set of management services, each representing one component of the CCL. The attribute `cCLComponentList` indicates the list of components which are combined to create a CCL.

The attribute cCLType identifies the type of CCL that needs to be composed. The specific details of the purpose that is fulfilled by the CCL are then written into the CCL purpose.

### 6.3.1.2 Attributes

The `ClosedControlLoop` IOC includes attributes inherited from `Top` IOC (defined in TS 28.622[5]) and the following attributes:

**Table 6.3.1.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| cCLComponentsInfo | O | T | T | F | T |
| operationalState | M | T | F | F | T |
| administrativeState | M | T | T | F | T |
| cCLPriority | M | T | T | F | T |
| cCLComponentList | O | T | T | T | T |
| cCLType | O | T | T | T | T |
| cCLActionTrigger | M | T | T | F | T |
| desiredBehavior | O | T | T | F | T |
| precedentEntities | O | T | T | F | T |
| desiredMetrics | M | T | T | F | T |
| **Attribute related to role** | | | | | |
| cCLPurposeRefList | M | T | T | T | T |

### 6.3.1.3 Attribute constraints

None

### 6.3.1.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

## 6.3.2 CCLScope

### 6.3.2.1 Definition

It indicates a scope of a CCL. It may be the measurement scope, control scope or impact scope.

The `CCLScope` includes the attribute `scopeType` that indicates the type of scope that represented by the particular scope instance.

The ScopeDescription attribute describes the scope that is instantiated or being informed about. The objectParameters lists the parameters on the objects in the ScopeDescription which are part of the scope.

The scopeOutcomes attribute indicates the set of outcomes desired for a given scope.

#### 6.3.2.2 Attributes

The CCLScope IOC includes attributes inherited from Top IOC (defined TS 28.622[5]) and the following attributes:

**Table 6.3.2.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| scopeType | M | T | F | F | T |
| ScopeDescription | M | T | F | F | T |
| objectParameters | M | T | F | F | T |
| scopeOutcomes | M | T | T | F | T |

#### 6.3.2.3 Attribute constraints

None.

#### 6.3.2.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

### 6.3.3 CCLReport

#### 6.3.3.1 Definition

This class represents the reported outcomes on a CCL instance, e.g., the information about the outcomes on one or the executing of the CCL. An CCLReport is contained by the entity containing the CCL, since the CCLReport can exist beyond the life of the CCL on which it is reporting.

There is one CCLReport per CCL for an observation time. The content of the CCLReport may be different for different observation time.

#### 6.3.3.2 Attributes

The CCLReport IOC includes attributes inherited from Top IOC (defined TS 28.622[5]) and the following attributes:

**Table 6.3.3.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| faultManagementCCLReport | CM | T | F | F | T |
| **Attributes related to role** | | | | | |
| | | | | | |

#### 6.3.3.3 Attribute constraints

**Table 6.3.3.3-1**

| Name | Definition |
|---|---|
| FaultManagementCCLReport | Condition: fault management is supported by CCL |

#### 6.3.3.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

## 6.3.4 CCLTrigger

### 6.3.4.1 Definition

This defines the criteria for CCL instantiation, composition and action execution.

### 6.3.4.2 Attributes

The `CCLTrigger` IOC includes attributes inherited from `Top` IOC (defined TS 28.622[5]) and the following attributes:

**Table 6.3.4.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLInstantiationTrigger` | O | T | F | F | T |
| `cCLCompositionTrigger` | O | T | F | F | T |
| **Attribute related to role** | | | | | |
| `closedControlLoopRef` | CM | T | F | F | T |

### 6.3.4.3 Attribute constraints

**Table 6.3.4.3-1**

| Name | Definition |
|---|---|
| `closedControlLoopRef` | Condition: `cCLInstantiationTrigger or cCLCompositionTrigger are` defined |

### 6.3.4.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

## 6.3.5 HistoricalCCLInfo

### 6.3.5.1 Definition

This IOC defines the historical information specific for a particular CCL. This IOC is instantiated by the producre as appropriate.

### 6.3.5.2 Attributes

The `HistoricalCCLInfo` IOC includes attributes inherited from `Top` IOC (defined TS 28.622[5]) and the following attributes:

**Table 6.3.5.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLObjectClass` | M | T | F | F | T |
| `cCLInstanceIdentifier` | M | T | F | F | T |
| `satisfactionScore` | M | T | F | F | T |
| `metricBreachInformation` | M | T | F | F | T |

### 6.3.5.3 Attribute constraints

None

#### 6.3.5.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

### 6.3.6 ConflictManagementAndCoordinationEntity

#### 6.3.6.1 Definition

This defines the conflict management functionality.

The IOC represents the `ConflictManagementAndCoordinationEntity` that is responsible for coordinating closed control loops to avoid, detect or resolve CCL conflicts.

The `ConflictManagementAndCoordinationEntity` is name-contained by `SubNetwork` or `ManagedElement` and is associated with one or more CCLs which the `ConflictManagementAndCoordinationEntity` shall be responsible for coordinating.

#### 6.3.6.2 Attributes

**Table 6.3.6.2-1**

| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLScopecoordinationCapability` | M | T | T | F | T |
| `cCLTriggerCoordinationCapability` | O | T | T | F | T |
| `cCLActionCoordinationCapability` | M | T | T | F | T |
| `cCLMetricValueCoordinationCapability` | M | T | T | F | T |
| `coordinatedCCLsScopes` | M | T | T | F | T |
| `cCLActionConflictsHandling` | M | T | T | F | T |
| `cCLhierarchyList` | O | T | T | F | T |
| `desiredCCLActions` | M | T | T | F | T |
| **Attribute related to role** | | | | | |
| | | | | | |

#### 6.3.6.3 Attribute constraints

None

#### 6.3.6.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

### 6.3.7 FaultManagement

#### 6.3.7.1 Definition

This IOC represents the Fault Management CCL purpose, which a list of attributes that describe the capabilities of the Fault Management CCL.

#### 6.3.7.2 Attributes

**Table 6.3.7.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `faultManagementAlarmIdList` | M | T | T | F | F |
| `faultManagementTimeWindow` | M | T | T | F | F |
| `faultManagementBackUpObjectRequirement` | O | T | T | F | F |
| `faultManagementIsolateObjectRequirement` | O | T | T | F | F |
| `clearUserId` | CM | T | T | F | F |

#### 6.3.7.3 Attribute constraints

**Table 6.3.7.3-1**

| Name | Definition |
|---|---|
| clearUserId | These attributes shall be supported for Fault Management CCL that clears ADMC alarms, as specified in TS 28.111 [4]. |

#### 6.3.7.4 Notifications

The common notifications defined in clauses 6.5 are valid for this IOC, without exceptions.

### 6.3.8 CCLComponentInfo <<dataType>>

#### 6.3.8.1 Definition

This data type represents a single purpose that describes what a CCL can do. The purpose is a list of characteristics that describe the capabilities of the CCL.

#### 6.3.8.2 Attributes

**Table 6.3.8.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLComponentId` | M | T | F | F | T |
| `cCLSteps` | M | T | F | F | T |

#### 6.3.8.3 Attribute constraints

None.

#### 6.3.8.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.9 CCLComponent <<dataType>>

#### 6.3.9.1 Definition

This dataType defines a CCL component that can be used or has been used to dynamically compose a closed control loop by the MnS consumer.

#### 6.3.9.2 Attributes

The `CCLComponent` IOC includes attributes inherited from Top IOC (defined in TS 28.622[5]) and the following attributes:

**Table 6.3.9.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLComponentRole` | M | T | T | T | T |
| `cCLComponentIdentification` | M | T | T | F | T |

#### 6.3.9.3 Attribute constraints

None

#### 6.3.9.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.10 FaultManagementCCLReport <<dataType>>

#### 6.3.10.1 Definition

This data type represents the Fault Management CCL report, which is a list of attributes that describe the result of the Fault Management.

#### 6.3.10.2 Attributes

**Table 6.3.10.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `generatedAlarmResultList` | M | T | F | T | T |
| `faultManagementCCLReportTime` | M | T | F | T | T |

#### 6.3.10.3 Attribute constraints

None.

#### 6.3.10.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.11 GeneratedAlarmResult <<dataType>>

#### 6.3.11.1 Definition

This data type represents the alarm result information generated by the CCL, which is a list of attributes that describe the result of the Fault Management for each alarm.

#### 6.3.8.2 Attributes

**Table 6.3.11.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `alarmId` | M | T | F | T | F |
| `alarmClearedStatus` | M | T | F | T | F |
| `identifiedRootCauseInformation` | M | T | F | T | F |
| `enhancedCorrelationInformation` | M | T | F | T | F |

#### 6.3.11.3 Attribute constraints

None.

#### 6.3.11.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.12 CCLPurpose <<dataType>>

#### 6.3.12.1 Definition

This data type represents a single purpose that describes what a CCL can do. The purpose is a list of characteristics that describe the capabilities of the CCL.

#### 6.3.12.2 Attributes

**Table 6.3.12.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| | | | | | |
| **Attributes related to role** | | | | | |
| | | | | | |

#### 6.3.12.3 Attribute constraints

None.

#### 6.3.12.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.13 CCLScopeCoordinationCapability <<dataType>>

#### 6.3.13.1 Definition

This data type represents the information and a capability of the `ConflictManagementAndCoordinationEntity` for Coordinating CCL instances to handle different CCL conflicts.

- The attribute `coordinatedScopeTypes` indicates the type of scopes for which the coordination is undertaken.

The logic needed for coordinating different scopes is different so each set of scopes to be coordinated must be of the same scope. The `ConflictManagementAndCoordinationEntity may have multiple` CCLScopeCoordinationCapability(s) differentiated by the type of scope that is being coordinated.

- The attribute `toBeCoordinatedScope` contains the set of CCL scopes that the coordinationEntity coordinates to ensure that they do not conflict. A CCL that requires its scopes to be evaluated for conflicts can add its scope into the list of coordinated scopes.

The attribute `detectedScopeConflict` indicates the list of conflicts that have been detected. Each conflict includes an indication for the type of conflict event, which in this case is ScopeConflict. It also has an indication for whether it is a potential conflict or an actual conflict that is observed.

`The fullCoordinatedScopeSpace` attribute indicates the full scope which is to be considered by the CoordinationEntity when selecting sub-allocations to different CCL instances.

#### 6.3.13.2 Attributes

**Table 6.3.13.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLCoordinationCapabilityID` | M | T | T | T | T |
| `coordinatedScopeTypes` | O | T | F | F | T |
| `fullCoordinatedScopeSpace` | M | T | T | T | T |
| `toBeCoordinatedCCLScopes` | M | T | T | T | T |
| `detectedScopeConflicts` | M | T | F | T | T |
| `detectedTriggerConflicts` | M | T | F | T | T |
| `detectedMetricValueConflicts` | M | T | F | T | T |

#### 6.3.13.3 Attribute constraints

None.

#### 6.3.13.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.14 CCLTriggerCoordinationCapability <<dataType>>

#### 6.3.14.1 Definition

This data type represents the information on a single set of CCL scope coordinated by the coordination entity. The `ScopeCoordinationSet` includes the type of scope to be coordinated, the set of Scopes to be coordinated and information on whether a Scope conflict is observed or not.

#### 6.3.14.2 Attributes

**Table 6.3.14.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLCoordinationCapabilityID` | M | T | T | T | T |
| `toBeCoordinatedPrecedentCCLs` | M | T | T | F | T |

#### 6.3.14.3 Attribute constraints

None..

#### 6.3.14.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

## 6.3.15 CCLActionCoordinationCapability <<datatype>>

#### 6.3.15.1 Definition

This defines the functionality for coordinating of CCL actions among CCLs to detect, avoid or resolve potential and real concurrent and non-concurrent actions conflicts.

- The CCLActionConflictsHandling datatype includes a `toBeCoordinatedActionPlans` attribute which is the list that contains information on the different action plans that the `coordinationEntity` attempts to resolve for direct action conflict.

- A CCL that requires its action plan to be evaluated for conflicts can notify its plan to the coordinationEntity which then be added to an appropriate list of `toBeCoordinatedActionPlans`. The CCL coordination entity checks the submitted configuration changes against other previous configuration changes from other CCLs (that have been executed) to see if there are any potential conflicting actions based on the provided information. This ensures to check planned configuration changes against actions that have already been executed.

- The `cCLParameterValuesUsefulness` attribute indicates how useful specific values of a parameter are good for the desired outcomes of a given CCL. On the other hand, the `cCLinterestInConflictParameter` attribute indicates the level of interest that the CCL has in the parameter – regardless of how useful specific values contribute to fulfilling that interest

- Given a list of CCLs whose plans are evaluated for concurrent or non-concurrent actions conflicts, the `ComputedCompromizePlans` attribute indicates the compromise action plans that are recommended by the coordinationEntity for each CCL. The `ComputedCompromizePlan` may include a sequence in which the actions may be executed.

The CCL has a detectedActionConflicts attribute that holds the list of detected conflicts in the set of action plans that have been evaluated.

- The `conflictMonitoringContext` attribute at a CCL A indicates the scope on which another CCL B has recently taken actions and for which that CCL B has limits in performance change that (called tolerenceLimits) that should be maintained by CCL A in that scope. The limited are added to each action plan that is executed.

#### 6.3.15.2 Attributes

**Table 6.3.15.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `cCLCoordinationCapabilityID` | M | T | T | T | T |
| `toBeCoordinatedActionPlans` | M | T | T | T | T |
| `detectedActionConflicts` | M | T | T | F | T |
| `cCLParameterValuesUsefulness` | M | T | T | F | T |
| `cCLinterestInConflictParameter` | M | T | T | F | T |
| `conflictMonitoringContext` | M | T | T | F | T |
| `computedCompromizePlans` | M | T | T | F | T |

#### 6.3.15.3 Attribute constraints

None

#### 6.3.15.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

## 6.3.16 CCLMetricValueCoordinationCapability <<datatype>>

### 6.3.16.1 Definition

This data type represents the information and a capability of the `ConflictManagementAndCoordinationEntity` for Coordinating CCL instances to handle different CCL conflicts.

### 6.3.16.2 Attributes

**Table 6.3.16.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| cCLCoordinationCapabilityID | M | T | T | T | T |
| proposedReviseddActionPlan | M | T | F | T | T |
| observedMetricValueConflicts | M | T | F | F | T |
| actionPlanFailedCriteria | M | T | F | F | T |
| TrustedCCLs | M | T | F | F | T |
| flipflopMetrics | M | T | T | F | T |

### 6.3.16.3 Attribute constraints

None.

### 6.3.D2.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

## 6.3.17 ScopeConflict <<datatype>>

### 6.3.17.1 Definition

This data type represents the information on a scope conflict.

Each conflict includes an indication in `ConflictType` attribute for whether it is a potential conflict or an actual conflict that is observed.

The `ConflictType` indicates the type of conflict that has been observed, i.e., either a potential conflict or an actual conflict.

### 6.3.17.2 Attributes

**Table 6.3.17.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| conflictID | M | T | T | F | T |
| conflictingCCLs | M | T | T | F | T |
| conflictScope | M | T | T | F | T |
| ConflictType | M | T | T | F | T |

### 6.3.17.3 Attribute constraints

None

#### 6.3.17.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.18 TriggerConflict <<datatype>>

#### 6.3.18.1 Definition

This data type represents the information on a trigger conflict.

Each conflict includes an indication in `ConflictType` attribute for whether it is a potential conflict or an actual conflict that is observed.

#### 6.3.18.2 Attributes

**Table 6.3.15.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `conflictID` | M | T | T | F | T |
| `conflictingCCLs` | M | T | T | F | T |
| `ConflictType` | M | T | T | F | T |

#### 6.3.18.3 Attribute constraints

None

#### 6.3.18.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.19 ActionConflict <<datatype>>

#### 6.3.19.1 Definition

This defines the information related with an action conflict among two or more CCLs.

Each conflict includes an indication in `ConflictType` for whether it is a potential conflict or an actual conflict that is observed.

#### 6.3.19.2 Attributes

**Table 6.3.19.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `conflictID` | M | T | T | F | T |
| `conflictingCCLId` | M | T | T | F | T |
| `conflictingActions` | M | T | T | F | T |
| `ConflictType` | M | T | T | F | T |

#### 6.3.19.3 Attribute constraints

None

#### 6.3.19.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.20 MetricValueConflict <<datatype>>

#### 6.3.20.1 Definition

This data type represents the information on a metric-value conflict.

Each conflict includes an indication in `ConflictType` attribute for whether it is a potential conflict or an actual conflict that is observed.

#### 6.3.20.2 Attributes

**Table 6.3.20.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `conflictID` | M | T | T | F | T |
| `conflictingCCLs` | M | T | T | F | T |
| `conflictingMetrics` | M | T | T | F | T |
| `ConflictType` | M | T | T | F | T |
| `correlatedOscillationMetrics` | M | T | T | F | T |

#### 6.3.20.3 Attribute constraints

None

#### 6.3.20.4 Notifications

- The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.21 ActionPlan <<datatype>>

#### 6.3.21.1 Definition

This data type represents the an action plan from a CCL instance.
For an action, a CCL B has limits in performance change (called tolerenceLimits) that should be maintained by any other CCL A taking action in the same scope. The limited are added to each action plan that is executed.

#### 6.3.21.2 Attributes

**Table 6.3.21.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `actionPlanID` | M | T | T | T | T |
| `cCLID` | M | T | T | T | T |
| `actions` | M | T | T | T | T |
| `toleranceLimits` | M | T | T | T | T |

#### 6.3.21.3 Attribute constraints

None.

#### 6.3.21.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.22 CCLActionConflictsHandling <<datatype>>

#### 6.3.22.1 Definition

- This defines the handling of CCL action conflict between the two existing CCLs.

#### 6.3.22.2 Attributes

**Table 6.3.22.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `conflictInformation` | M | T | T | F | T |
| `conflictResolution` | M | T | T | F | T |
| `targetCCL` | M | T | F | F | T |

#### 6.3.22.3 Attribute constraints

None

#### 6.3.22.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.23 ActionConflictResolution <<datatype>>

#### 6.3.23.1 Definition

This defines the information related with conflict resolution configured by the MnS Consumer.

#### 6.3.23.2 Attributes

**Table 6.3.23.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `conflictingCCLId` | M | T | T | F | T |
| `cCLRequirementBreachPercentage` | M | T | F | F | T |

#### 6.3.23.3 Attribute constraints

None

#### 6.3.23.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

## 6.3.24 MetricBreachInformation <<data type>>

### 6.3.24.1 Definition

This defines the requirements breach information related with the CCL.

### 6.3.24.2 Attributes

**Table 6.3.24.2-1**

| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
|---|---|---|---|---|---|
| `breachedMetricIdentification` | M | T | F | F | T |
| `breachTime` | M | T | F | F | T |
| `mitigationAction` | M | T | F | F | T |

### 6.3.24.3 Attribute constraints

None

### 6.3.24.4 Notifications

The subclause 6.5 of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

## 6.4 Attribute definitions

### 6.4.1 Attribute properties

**Table 6.4.1-1**

| Attribute Name | Documentation and Allowed Values | Properties |
|---|---|---|
| scopeType | It indicates the type of scope that represented by the particular scope instance.<br><br>allowedValues: CCL_MEASUREMENT_SCOPE, CCL__TARGETED__SCOPE, CCL_CONTROL_SCOPE, CCL_IMPACT_SCOPE<br><br>Editor's Note: The allowed values will be revisited | type: Enum<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| ScopeDescription | It indicates the description of the scope that is instantiated or being informed about. It is defined according to the ScopeDefinition in TS28.561 | type: ScopeDefinition<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| objectParameters | It indicates the list of parameters on the objects in the ScopeDescription which are part of the scope. This applies when the scope is of type measurement scope or control scope.<br><br>allowedValues: string | type: String<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| coordinationCapability | It indicates a capability of a coordination entity to coordinate CCL conflicts | type: CoordinationCapability<br>multiplicity: *<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| cCLCoordinationCapabilityID | It indicates an identifier for a specific CCL conflicts coordination capability | type: String<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| closedControlLoopRefList | It indicates a list of DN for ClosedControlLoop Instances.<br><br>allowedValues: N/A | type: DN<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLScopeCoordinationCapability | It indicates a CCL scope assignment and conflict coordination capacity | type: CCLScopeCoordinationCapability<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLTriggerCoordinationCapability | It indicates a specific type of CCL trigger coordination functionality of the ConflictManagementAndCoordinationEntity | type: CCLTriggerCoordinationCapability<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| cCLActionCoordinationCapability | It indicates a specific type of CCL conflict coordination functionality of the ConflictManagementAndCoordinationEntity | type: CCLActionConflictsHandling<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLMetricValueCoordinationCapability | It indicates a specific type of CCL conflict coordination functionality of the ConflictManagementAndCoordinationEntity | type: CCLMetricValueCoordinationCapability<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| coordinatedCCLsScopes | It indicates the scopes of the CCL that are coordinated by the coordinationEntity<br><br>It is a pair <string_1, string_2 > where string_1 is the DN of a CCL being coordinated and string_2 the DN of that CCL's CCLScope. | type: pair <string, string ><br>multiplicity: 2 ..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| operationalState | It indicates the operational state of the ClosedControlLoop instance. It describes whether the resource is installed and partially or fully operable (Enabled) or the resource is not installed or not operable (Disabled).<br><br>AllowedValues; Enabled/Disabled<br><br>allowedValues: "ENABLED", "DISABLED".<br>The meaning of these values is as defined in 3GPP TS 28.625 [8] and ITU-T X.731 [9]. | type: ENUM<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: Disabled<br>isNullable: False |
| administrativeState | It indicates the administrative state of the ClosedControlLoop instance. It describes the permission to use or the prohibition against using the ClosedControlLoop instance. The administrative state is set by the MnS consumer.<br><br>AllowedValues; Locked/Unlocked<br><br>allowedValues: "LOCKED", "UNLOCKED".<br>The meaning of these values is as defined in 3GPP TS 28.625 [8] and ITU-T X.731 [9]. | type: ENUM<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: Locked<br>isNullable: False |
| cCLComponentsInfo | It indicates information on the constituent components of a CCL.<br><br>allowedValues: N/A | type: CCLComponentInfo<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLComponentId | It indicates the identifier of a CCL component. It is the DN of an object instantiated to act as a component of the CCL | type: DN<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLSteps | It indicates the CCL steps or functionality that is accomplished by a CCL component.<br><br>allowedValues: DATA_COLLECTION, ANALYSIS, DECISION, EXECUTION | type: Enum<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| `faultManagementAlarmIdList` | It describes the list of IDs of alarms to be managed by Fault Management CCL.<br><br>allowedValues: A list of alarmIds as specified in TS 28.111 [4], clause 7.4.1 | type: List<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: True |
| `faultManagementTimeWindow` | It describes the information of a time window (including start and end time) specified by the consumer for fault management to carry out troubleshooting and to clear the alarms.<br><br>allowedValues: timeWindow as defined in 3GPP TS 28.622 [5], clause 4.4.1 | type: TimeWindow<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: True |
| `faultManagementBackUpObjectRequirement` | It describes whether to back-up the alarmed object is required by the consumer before fault management.<br><br>allowedValues: True, False | type: Boolean<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `faultManagementIsolateObjectRequirement` | It describes whether to isolate the alarmed object from interaction with other objects is required by the consumer before fault management.<br><br>allowedValues: True, False | type: Boolean<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `clearUserId` | It carries the identity of the Fault Management CCL who is the consumer that invokes the clearAlarms operation.<br><br>allowedValues: clearUserId as defined in 3GPP TS 28.111 [4], clause 7.4.1 | type: string<br>multiplicity: 0..1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `faultManagementCCLReport` | It describes the Fault Management CCL report.<br><br>allowedValues: Not Applicable | type: FaultManagementCCLReport<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `generatedAlarmResultList` | It describes the list of generated alarm results<br><br>allowedValues: A list of `GeneratedAlarmResult` | type: List<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| | | |
| `faultManagementCCLReportTime` | It describes the time when the `FaultManagementCCLReport` is created.<br><br>allowedValues: `DateTime` as specified in TS 28.622 [5]. | type: DateTime<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `alarmId` | It identifies an AlarmRecord as specified in TS 28.111 [4]<br><br>allowedValues: A string as specified in TS 28.111 [4] | type: string<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `alarmClearedStatus` | It describes whether an alarm is cleared by the Fault Management CCL when the identified root cause is resolved.<br><br>allowedValues: True, False | type: Boolean<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| identifiedRootCauseInformation | It describes root cause information identified by the Fault Management CCL.<br><br>allowedValues: String | type: string<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| enhancedCorrelationInformation | It describes the list of correlated alarm Ids identified by the Fault Management CCL<br><br>allowedValues: A list of alarmId | type: List<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| cCLActionConflictsHandling | This defines the handling of CCL action conflict between the two existing CCLs. | Type: CCLActionConflictsHandling<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| detectedActionConflict | This indicates the information related with a detected conflict CCL. It is a list of conflicts among a set of action plans that have been evaluated. Each entry is a pair of plans that are conflicting. | Type: ActionConflict<br>multiplicity: *<br>isOrdered: True<br>isUnique: False<br>defaultValue: None<br>isNullable: False |
| conflictResolution | This defines the information related with conflict resolution. | Type: ActionConflictResolution<br>multiplicity: *<br>isOrdered: True<br>isUnique: False<br>defaultValue: None<br>isNullable: False |
| targetCCL | The identification of the CCL that need to be deleted or updated to resolve conflict. This will be decided as per the information ConflictResolution. | Type: Dn<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| conflictingCCLId | This indicates the CCL identification | Type: Dn<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| conflictingActions | This provides the set of actions that have been taken by the CCL as part of the Execute step. | Type: String<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLPriority | This provides the priority of the CCL. This will be the numerical value between 1 to 10, with 1 being the least priority. | Type: String<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| cCLMetricBreachPercentage | It defines the breach percentage per metric in terms of how bad the metric(s) is breached. For example, if the metric of guaranteed throughput is 200mbps and the actual throughput is coming to be 100mbps then the breach percentage would be 50%. The CCL that have higher percentage of breach will be prioritized | Type: Integer<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| cCLComponentList | It indicates the list of components acting as steps of the CCL, each either a MnF or a MnS producer whose services can be part of the CCL. The cCLComponent may have a role among MONITOR; ANALYSIS; DECISION; EXECUTION. Or OTHER. OTHER. Is used for example in the cases where a components fulfils more than 1 role or where the role can be simply described by the four options.<br><br>The cCLComponents are sequenced, i.e., cCLComponents is an ordered list. For example, if there are 2 steps that contribute to the analysis role, it is necessary to show how those steps are sequenced. The order in which they are listed indicates the order in which their services should be chained to complete the CCL | type: CCLComponent<br>multiplicity: 1..*<br>isOrdered: True<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLType | It indicates a type or Category of CCL that is to be instantiated or dynamically composition. It indicates the kind of capability that will be accomplished by the CCL instance, e.g. ENERGYOPTIMIZATION, SLICEASSURANCE, etc.<br><br>The specific details, characteristics and behavior of a CCL for a given CCL type are then written into the CCL purpose.<br><br>Note: The allowed values are FFS | type: String<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| cCLComponentRole | It indicates a role accomplished by CCL component.<br><br>AllowedValues: MONITOR; ANALYSIS; DECISION; EXECUTION, OTHER. Is used for example in the cases where a components fulfils more than 1 role or where the role can be simply described by the four options | type: Enum<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLComponentIdentification | It indicates the entity accomplishing the component.<br><br>It may be the DN of an MOI or the combination of URI and DN that can be used to fulfil that role. | Type: String<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLInstanceIdentifier | This defines the specific CCL instance | Type: Dn<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| satisfactionScore | The numerical value from 1 to 10 (1 being the worst), providing the consumer satisfaction with the CCL. | Type: Integer<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| metricBreachInformation | This defines the goalrequirement breach information related with the CCL. | Type: MetricBreachInformation<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| breachedMetricIdentification | This defines the goalrequirement which got breached | Type: String<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| `breachedTime` | This defines the time of the goalrequirement breach | Type: DateTime<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `mitigationAction` | This defines the configuration actions that was performed by the CCL execution to mitigate the breach. | Type: String<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `cCLInstantiationTrigger` | This defines dynamic closed control loop invocation criteria that can be configured by the consumer. The producer will instantiate an CCL based on the criteria defined. | Type: TriggerConditionDescriptor<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `cCLCompositionTrigger` | This defines dynamic closed control loop composition criteria that can be configured by the consumer. The producer will compose an CCL based on the criteria defined. | Type: TriggerConditionDescriptor<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `closedControlLoopRef` | This refers to the CCL that is composed or instantiated using triggers. | Type: Dn<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `cCLActionTrigger` | This defines the criteria/conditions under which the CCL is allowed to take actions. | Type: CCLTrigger<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `desiredBehavior` | This will define the corresponding behavior of the CCL. The behaviours can be represented by an ENUM to include:<br><br>- DECISION_ACTIVATION: The CCL executes the recommendations that it derives on to the network.<br><br>- NOTIFY_RCOMMENDATION: The CCL starts processing input to derive recommendations but without the corresponding actions executed on the network. Instead, the recommendation is notified to the consumer who then considers whether it should be applied or not.<br><br>- DO_NOTHING: do not do anything. | Type: ENUM<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `scopeOutcomes` | It indicates the set of outcomes to be coordinated for a given scope as part of scope coordination. It is a pair <A,B> where A is the metric and B the desired outcome on that metric. | Type: pair<string, Real><br>multiplicity: 1...*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `conflictID` | It identifies a conflict event | type: Integer<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| `conflictingCCLs` | It identifies the set of CCLs that are conflicting | type: DN<br>multiplicity: 2<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `conflictScope` | It indicates the scope for which two or more CCLs are conflicting. | Type: `ScopeDefinition`<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `ConflictType` | It indicates the type of conflict that has been observed, i.e., either a potential conflict or an actual conflict.<br><br>allowedValues: POTENTIAL_CONFLICT; ACTUAL_CONFLICT | Type: ENUM<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `coordinatedScopeTypes` | It indicates the types of scopes under consideration for coordination by a scope coordination functionality.<br><br>allowedValues: CCLMEASUREMENTSCOPE, CCLTARGETSCOPE, CCLCONTROLSCOPE, CCLIMPACTSCOPE, CCLMONITOREDSCOPE | Type: ENUM<br>multiplicity: 1 ..5<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `fullCoordinatedScopeSpace` | It indicates the full scope which is to be considered by the CoordinationEntity when selecting sub-allocations to different CCL instances. | Type: Scope<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `toBeCoordinatedCCLScopes` | It indicates the list of scopes which the coordinatinEntity is responsible for coordinating to ensure they have no conflicts. A CCL that requires its scope to be evaluated for conflicts can add its scope set into the list of scopes sets | Type: CCLScope<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `detectedScopeConflicts` | It indicates the list of scope conflicts that are detected by the coordinationEntity. Each entry is of type: scope conflict | Type: ScopeConflict<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `detectedTriggerConflicts` | It indicates the list of trigger conflicts that are detected by the coordinationEntity. Each entry is of type: TriggerConflict | Type: TriggerConflict<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| `precedent`Entities | It indicates the set of instances of CCLs or other functionality that should be executed before the CCL | Type: DN<br>multiplicity: 1 ..*<br>isOrdered: True<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLhierarchyList | It indicates the ordered list of CCL instances defining the order in which CCLs should be executed. It is an ordered list where the first entry is the one to be executed first. | Type: DN<br>multiplicity: 1 ..*<br>isOrdered: True<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| `toBeCoordinatedPrecedentCCLs` | It indicates the set of instances of CCLs or other functionality that need to be coordinated | Type: DN<br>multiplicity: 1 ..*<br>isOrdered: False<br>isUnique: False<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| detectedScopeConflicts | It indicates the list of scope conflicts that are detected by the coordinationEntity. Each entry is of type: ScopeConflict. | Type: ScopeConflict<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| actionPlanID | It identifies an actionPlan generated by a CCL | type: string<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| cCLID | It identifies the DN of a CCL that has generated an actionPlan | type: DN<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| actions | It indicates the CM changes proposed a CCL | type: PlannedConfigurationDescriptor<br>multiplicity: 1 ..*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| toBeCoordinatedActionPlans | It indicates the list of action plans which the coordinatinEntity is responsible for coordinating to ensure they have no conflicts. A CCL that requires its action plan to be evaluated for conflicts can notify its plan to the coordinationEntity to then be added to an appropriate list of toBeCoordinatedActionPlans. Each list includes plans with related (or same) scope in managed objects and time. | Type: ActionPlan<br>multiplicity: *<br>isOrdered: False<br>isUnique: False<br>defaultValue: None<br>isNullable: False |
| cCLParameterValuesUsefulness | It indicates the relative goodness of different values of the parameter to the CCL. It a list of pairs <A, B> where A is a value of CCL control parameter and B is an integer indicating the usefulness of value A. B is in the scale [0:100], where “0” indicates that the value is useless while “100” indicates that the functionality of the CCL completely depends on that value.<br><br>allowedValues: [0, 100] | Type: pair<string,integer><br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| cCLinterestInConflictParameter | It indicates CCL’s relative interest in the parameter. It is a measure of how useful different parameters are to the objectives of the CCL, regardless of how useful specific values of those parameters contribute to fulfilling those objectives.<br><br>allowedValues: [0, 100] | Type: integer<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| conflictMonitoringContext | It indicates the scope that one CCL B notify to another CCL A to monitor and ensure to maintain the performance within some stated limits. It is written by the CCL B into coordinatinEntity as the pair pair<actionID, Scope> where actionID is the identifier of a previous action that has been taken by a CCL and Scope is the scope which that CCL wants other CCLs to maintain within certain limits | Type: pair<actionID, Scope><br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| toleranceLimits | It indicates the limits within which the compromise on the parameters and metrics can still be acceptable. It is an integer indicting the acceptable percentage change in the values on parameters in a specific action plan.<br><br>allowedValues: [0, 100] | Type: integer<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| ComputedCompromizePlans | It indicates the compromise action plans that are recommended by the coordinationEntity for each CCL. It is list with each entry a pair <CCL_ID, compPlan> where CCL_ID is the identifier of a CCL for which a compromise plan has been computed, and compPlan is the proposed compromise plan | Type: ActionPlan<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| desiredMetrics | It indicates the set of metrics that the CCL intends to optimize. These need to be coordinated among several CCLs, e.g. so that 2 CCLs don't aim to optimize the same metric | Type: string<br>multiplicity: 1...*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| proposedReviseddActionPlan | It indicates a compromise action plan proposed by the coordination entity for the case where the action plan executed by a CCL resulted in metric value conflict | Type: ActionPlan<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| actionPlanFailedCriteria | It indicates criteria which an action plan for which an action plan failed and caused metric value conflicts. | Type: string<br>multiplicity: 1...*<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| TrustedCCLs | It indicates the list of CCL that have performed consistently well and have achieved full trust that not further check of their actions is necessary. | Type: DN<br>multiplicity: *<br>isOrdered: False<br>isUnique: True<br>defaultValue: None<br>isNullable: False |
| observedMetricValueConflicts | It indicates the list of observed metric value conflicts | Type:<br>MetricValueConflict<br>multiplicity: *<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| correlatedOscillationMetrics | It indicates the metrics noted to be experiencing correlated oscillations | Type: string<br>multiplicity: 1..*<br>isOrdered: False<br>isUnique:<br>TruedefaultValue: None<br>isNullable: False |
| conflictingMetrics | It indicates the list of metrics that are in conflict | Type: string<br>multiplicity: 1...*<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| detectedMetricValueConflicts | It indicates the list of MetricValueConflicts that are detected by the coordinationEntity. Each entry is of type: MetricConflict | Type:<br>MetricValueConflict<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |
| toleranceLimits | It indicates the limits within which the compromise on the parameters and metrics can still be acceptable. It is an integer indicting the acceptable percentage change in the values on parameters in a specific action plan.<br><br>allowedValues: [0, 100] | Type: integer<br>multiplicity: 1<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

| | | |
|---|---|---|
| `flipflopMetrics` | It indicates the list of metrics that are observed by a CCL as flip flopping. It is a pair <objDN, ffmetric> where objDN is DN of the managed object whose metric is flipflopping and ffmetric is identifier of the flip flopping metric. | Type: pair <DN, string><br>multiplicity: *<br>isOrdered: N/A<br>isUnique: N/A<br>defaultValue: None<br>isNullable: False |

## 6.5 Common notifications

### 6.5.1 Configuration notifications

This clause presents a list of notifications, defined in TS 28.532 [3], that an MnS consumer may receive. The notification header attribute `objectClass/objectInstance` shall capture the DN of an instance of a class defined in the present document.

**Table 6.2.1.5.1-1**

| Name | Qualifier | Notes |
|---|---|---|
| `notifyMOICreation` | O | -- |
| `notifyMOIDeletion` | O | -- |
| `notifyMOIAttributeValueChanges` | O | -- |

# 7 Procedures

## 7.1 Procedure for conditional trigger/instantiation of CCLs

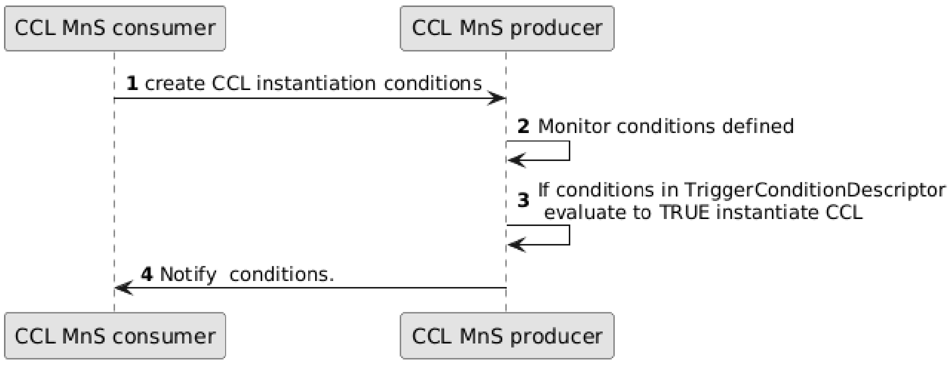

CCL MnS consumer
CCL MnS producer
1 create CCL instantiation conditions
2 Monitor conditions defined
3 If conditions in TriggerConditionDescriptor evaluate to TRUE instantiate CCL
4 Notify conditions.
CCL MnS consumer
CCL MnS producer

**Figure 7.1-1: Procedure and interactions for conditional trigger/instantiation of CCLs**

Step 0: There exists an object exposing the MnS producer responsible for instantiating CCLs. This object may be represented by a subnetwork or managed element.

Step 1: The MnS consumer creates on the MnS producer responsible for instantiating CCLs the set of conditions to be evaluated for instantiation of the CCL. These conditions are created as an instance of TriggerConditionDescriptor defined in 28.572. TriggerConditionDescriptor describes the conditions that should be evaluated including performance, provisioning and fault management conditions. The performance conditions includes managed object, measurement/KPI name and the trigger value. The provisioning conditions includes the managed object, location, event and time of the provisioning events. The fault conditions includes managed object, alarmSeverityThreshold and alarmTypeThreshold.

Step 2: The MnS producer monitors the network to detect when the conditions defined in TriggerConditionDescriptor evaluate to TRUE.

Step 3: If conditions in TriggerConditionDescriptor evaluate to TRUE, the MnS producer instantiates the CCL.

Step 4: For the instantiated CCL, the MnS producer may notify the conditions that triggered the CCL.

## 7.2 Procedure for conditional composition of CCLs

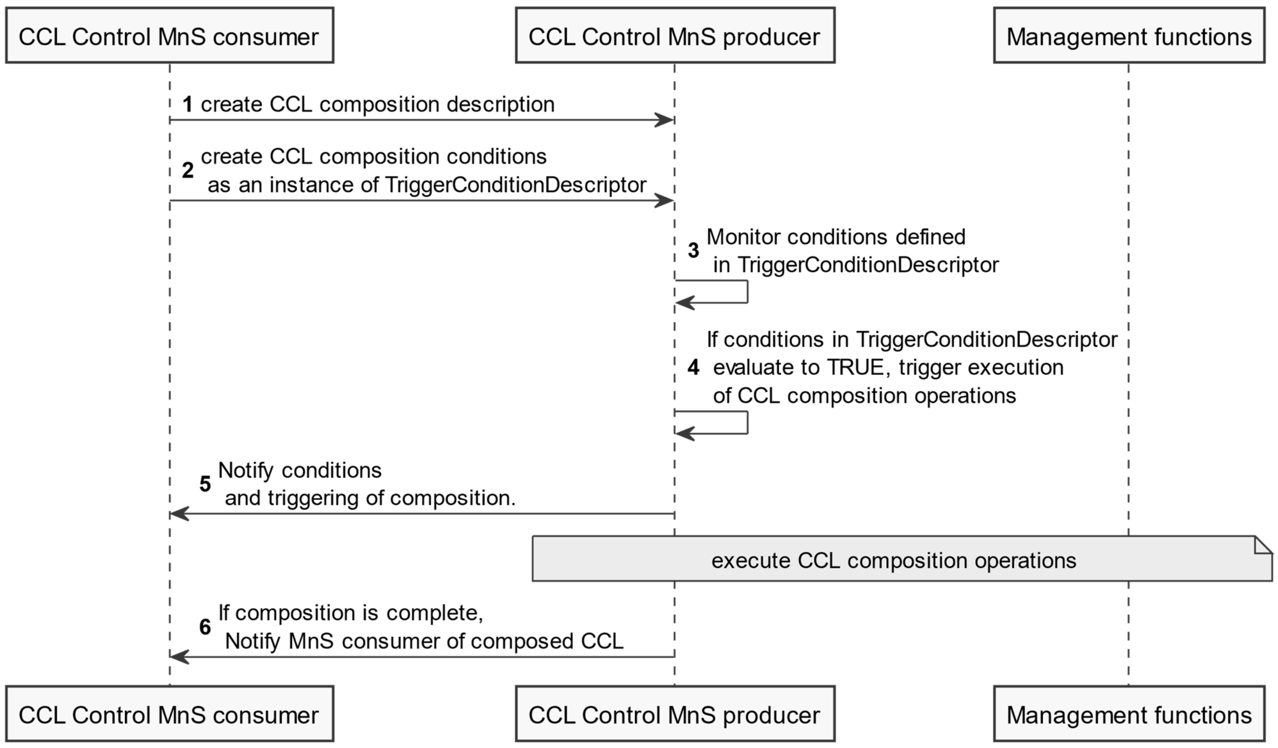

CCL Control MnS consumer
CCL Control MnS producer
Management functions
1 create CCL composition description
2 create CCL composition conditions as an instance of TriggerConditionDescriptor
3 Monitor conditions defined in TriggerConditionDescriptor
4 If conditions in TriggerConditionDescriptor evaluate to TRUE, trigger execution of CCL composition operations
5 Notify conditions and triggering of composition.
execute CCL composition operations
6 If composition is complete, Notify MnS consumer of composed CCL

**Figure 7.2-1: Procedure and interactions for conditional composition of CCLs**

Step 0: There exists an object exposing the MnS producer responsible for instantiating CCLs. This object may be represented by a subnetwork or managed element.

Step 1: The MnS consumer creates on the MnS producer responsible for instantiating CCLs the CCL composition operations description which contains the details on the provisioning actions to be undertaken – in this case the operations for composing the CCL. These may include

- createMOI operations for instantiating the objects to be used as components of the closed control loop, e.g., a PMJob to be used to collect data

- modifyMOI operations for configuring the instantiated components to enable them to operate as a single loop, e.g., to configure the PMJob to deliver data to an analytics instance

Step 2: The MnS consumer creates on the MnS producer responsible for instantiating CCLs the set of conditions to be evaluated for composing the CCL. These conditions are created as an instance of TriggerConditionDescriptor defined in 28.572. TriggerConditionDescriptor describes the conditions that should be evaluated including performance, provisioning and fault management conditions

Step 3: The MnS producer monitors the network to detect when the conditions defined in TriggerConditionDescriptor evaluate to TRUE.

Step 4: If conditions in TriggerConditionDescriptor evaluate to TRUE, the MnS producer triggers execution of CCL composition operations.

Step 5: For the triggered CCL composition, the MnS producer may notify the conditions that triggered the composition or the composed CCL.

Step 6: The MnS producer executes the CCL composition operations through interaction with other management functions and services. When the composition is complete, the MnS producer may notify the MnS consumer of composed CCL.

## 7.3 CCL Performance Monitoring

When the PA (Performance Assurance)MnS consumer notices that a slice or network performance is degrading, it may require to know information about available CCLs that have the requirements related to this performance degradation. This may imply that the performance of the related CCL is not as expected. This requires performance management to be done on the available CCL including further actions such as evaluating and updating closed control loops. The metrics for assessing performance  of CCLs, for example, total number of occurrences of a requirementbreach, time taken by CCL to meet a breached requirement, total number of conflicts occurred by a CCL are defined in clause 8. A procedure for performance management of CCLs involving these performance metrics is described below

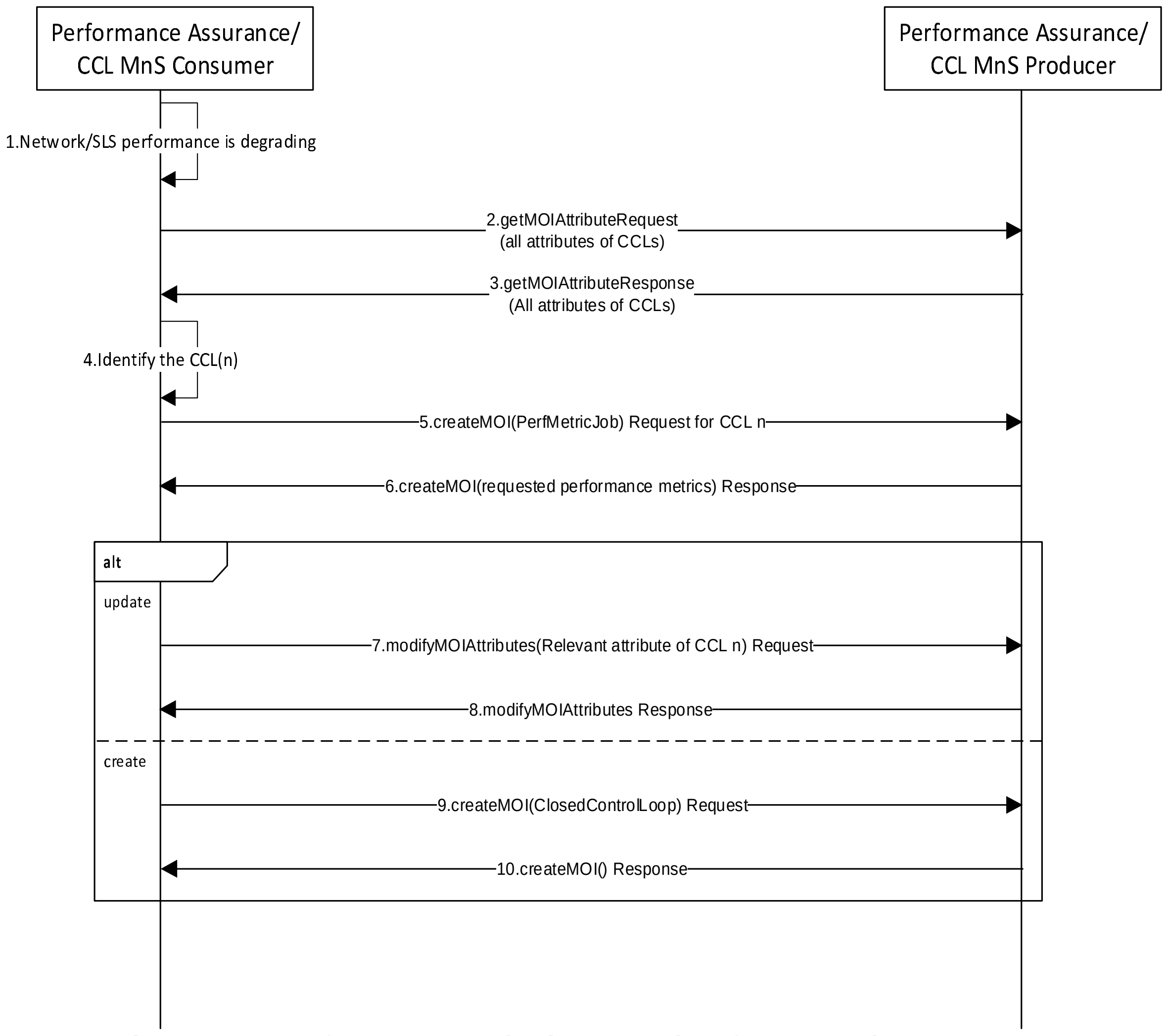

Performance Assurance/ CCL MnS Consumer
Performance Assurance/ CCL MnS Producer
1.Network/SLS performance is degrading
2.getMOIAttributeRequest (all attributes of CCLs)
3.getMOIAttributeResponse (All attributes of CCLs)
4.Identify the CCL(n)
5.createMOI(PerfMetricJob) Request for CCL n
6.createMOI(requested performance metrics) Response
alt
update
7.modifyMOIAttributes(Relevant attribute of CCL n) Request
8.modifyMOIAttributes Response
create
9.createMOI(ClosedControlLoop) Request
10.createMOI() Response

**Figure 7.3-1: Performance monitoring procedure for a closed control loop**

Step 1. PA/CCL MnS consumer notices that a certain performance metric of a SLS or a network starts degrading.

Step 2. PA/CCL MnS consumer sends getMOIAttributeRequest message to PA/CCL MnS producer for getting information about all CCLs attributes.

Step 3. PA/CCL MnS producer provides this information of all CCLs to the consumer in getMOIAttributeResponse message.

Step 4. PA/CCL MnS consumer identifies the CCL (n) which is responsible for maintaining the performance of slice or network.

Step 5. PA/CCL MnS consumer sends createMOI(PerfMetricJob) request to PA/CCL MnS producer for obtaining status of following performance metrics for that particular CCL(n) as defined in clause 8 - TotalAssuranceGoalBreach, TimeCorrectiveGoalMeet, TotalCclConflicts_Filter.

Step 6. PA/CCL MnS producer provides requested performance metric values via createMOI() Response message to PA/CCL MnS consumer.

Step 7. PA/CCL MnS consumer has two choices – either to update the existing CCL n (of step 4) to create a new CCL for the same. If PA/CCL MnS consumer chooses to modify an existing CCL, it sends a modifyMOIAttributes request message for that CCL or it can also update by sending changeMOIs request message to PA/CCL MnS producer.

Step8. Accordingly, PA/CCL MnS producer sends modifyMOIAttributes Response or changeMOIs response message to PA/CCL MnS consumer for the updated attributes of CCL n.

Step9. If PA/CCL MnS consumer chooses to create a new CCL, it does so by sending createMOI Request message to PA/CCL MnS producer.

Step10. PA/CCL MnS producer provides createMOI() Response message for the newly created CCL MOI to PA/CCL MnS consumer.

## 7.4 CCL decision escalation

To enable escalation, there has to be entities to which decision can be escalated, called escalation recipients. These are mainly closed control loops but other decision makers, e.g. AIML inference functions could be used as escalation recipients. The CCL which wishes to escalate a decision is named an escalator CCL.

To enable escalation, each CCL contains an attribute identifying an entity acting as an escalation recipient to which a decision is escalated. The CCL also contains an attribute for defining the condition that triggers the escalation. For example, the CCL may trigger escalation when its level of confidence in the derived decision is below some threshold, in which case the confidence threshold is the condition for triggering the escalation. The confidence threshold attribute enables the CCL to autonomously make decisions for each situation and context based on its computed confidence level in the given situation. If the confidence level is lower than the confidence threshold the decision is escalated otherwise the decision is executed.

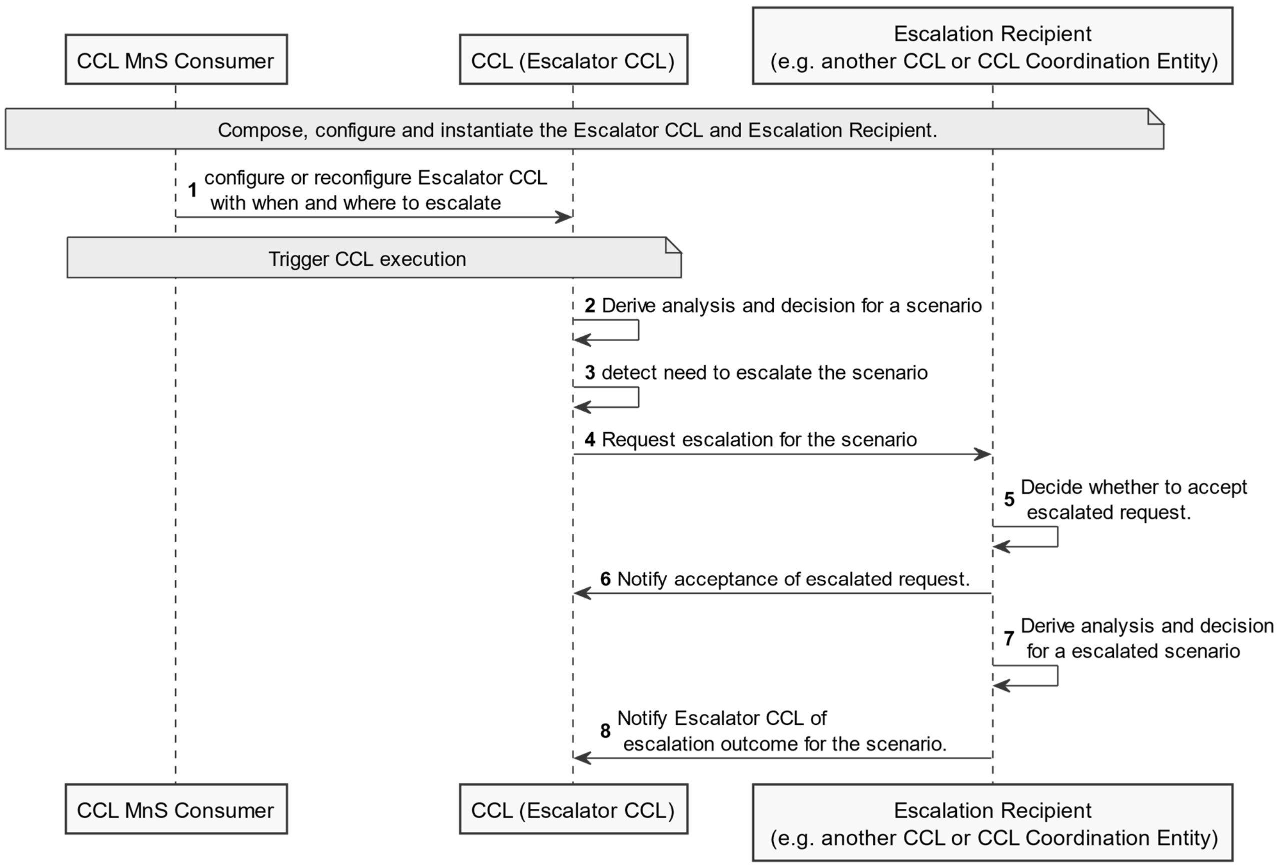

CCL MnS Consumer
CCL (Escalator CCL)
Escalation Recipient (e.g. another CCL or CCL Coordination Entity)
Compose, configure and instantiate the Escalator CCL and Escalation Recipient.
1 configure or reconfigure Escalator CCL with when and where to escalate
Trigger CCL execution
2 Derive analysis and decision for a scenario
3 detect need to escalate the scenario
4 Request escalation for the scenario
5 Decide whether to accept escalated request.
6 Notify acceptance of escalated request.
7 Derive analysis and decision for a escalated scenario
8 Notify Escalator CCL of escalation outcome for the scenario.
CCL MnS Consumer
CCL (Escalator CCL)
Escalation Recipient (e.g. another CCL or CCL Coordination Entity)

**Figure 7.4-1: Procedure and interactions for CCL decision escalation**

Step 0. The escalator CCL and the escalation recipient are composed, configured and instantiated.

Step 1. The MnS consumer configure the escalator CCL with information about the conditions under which to escalate and when to escalate to (the escalation recipient ). The escalator CCL can the trigger escalation either on its own or based on extra information form the Mns consumer.

Step 2. The escalator CCL executes analysis and decision making for a scenario. If the escalator CCL is confident with its decision it executes as normal

Step 3. The escalator CCL detect the need to escalate, e.g., for the case where it is not confident with its decision, the lack of confidence is the indicator of a scenario that should be escalated.

Step 4. When a CCL requires an escalation, escalator CCL instantiates a request for escalation on the escalation recipient.. The escalation request includes:

- An attribute for proposed CM change as a plan containing the configuration management changes that has been proposed by the escalator CCL.

- An attribute for the context and conditions describing decision constraints observed by the escalating CCL in making the decision(s).

Step 5. Based on the information in the escalation request, the capabilities of the escalation recipient as well as its observations of the evaluated network state, the escalation recipient decides whether it can undertake the escalation or not.

Step 6. The escalation recipient can notify (send an acceptance of) the escalation request.

Step 7. For an accepted escalation request, the escalation recipient derives an outcomes for the request

Step 8. The escalation recipient provides the outcomes to the escalator CCL, by writing it into an escalation outcome attribute on the escalation recipient. The outcome may then be written into an equivalent attribute on the escalator CCL, i.e., the escalator CCL contains an attribute for the escalation outcome to which the escalation recipient writes its computed escalation outcome. The escalation recipient contains an attribute for an escalation outcomes report in which it writes the derived outcomes for each corresponding escalation request. This can then be notified to the escalator CCL which subsequently reads it to obtain the recommendations.

The escalation outcome indicates whether the escalator CCL should take any action and what that action is. Accordingly, it contains an ENUM attribute to indicate what should be done by the escalator CCL, with the values:

- "DONOTHING"- indicating that the escalator CCL does not need to take any action, i.e. the escalation recipient is addressing the scenario.
- "APPLYACTION"- indicating that the escalator CCL should apply a specific set of actions proposed by the escalation recipient. The action is written into a proposed-actions attribute of the escalation outcome, which is of the type plan according to TS 28.572[6].
- "APPLYGUIDANCE"- indicating that the escalator CCL should compute a new CM change based on the guidance from the escalation recipient. The guidance is written into the proposed-actions attribute.

## 7.5 CCL-impact assessment and metric conflicts resolution

A CCL (called the actor-CCL) may not know the full scope that its actions will impact. And this may also not be known by the CCL coordination entity, In that case, the impact can be collected from the entities that have been affected by the CCL’s actions - jointly called impacted entities. The CCL contains an attribute, called executedAction attribute, which contains information indicating that an action has been taken that may affect the other CCLs (thus requesting feedback on how much impact there has been); and the CCL-action-impact time indicating the time when the affected entities should provide feedback. Any entity which may be impacted by the CCL actions (e.g. e.g. the CCL coordination entity or other CCLs) subscribes to be notified of changes to the executedAction and the related CCL-action-impact time.

After an action, the CCL updates the executedAction so that notifications are sent to the subscribed entities to indicate that if the entity is affected, it should provide its feedback on the effect in a time not exceeding the CCL-action-impact time. The notification may also be sent to the CCL coordination entity which then notifies that respective affected entities, e.g. other CCLs or other management functions.

The Impacted entity computes its observed impact in form of an index, called the Action Quality Indicator (AQI), that describes and quantifies the observed impact, i.e. it indicates the degree to which the action was good or bad to their objectives. The Action Quality Indicator is an integer in the range [0,10] where "0" indicates that the action was completely unacceptable and should never be reused in that context while "10" indicates that the action had very good outcomes for the reporting Impacted entity (e.g. the affected CCL). An index is used instead of sending specific metrics measured by each Impacted entity because specific metrics would require the actor-CCL to understand all the different metrics in exactly the same way as the Impacted entities do, which is not guaranteed to always be true. The AQI is specific to each CCL and to each scenario thar the CCL evaluates - since it is used to check how good or bad an action was for that CCL in that scenario. Accordingly, its computation would vary depending on the CCL and scenario but can be computed in a uniform way as a weighted sum of normalized KPIs) of that CCL.

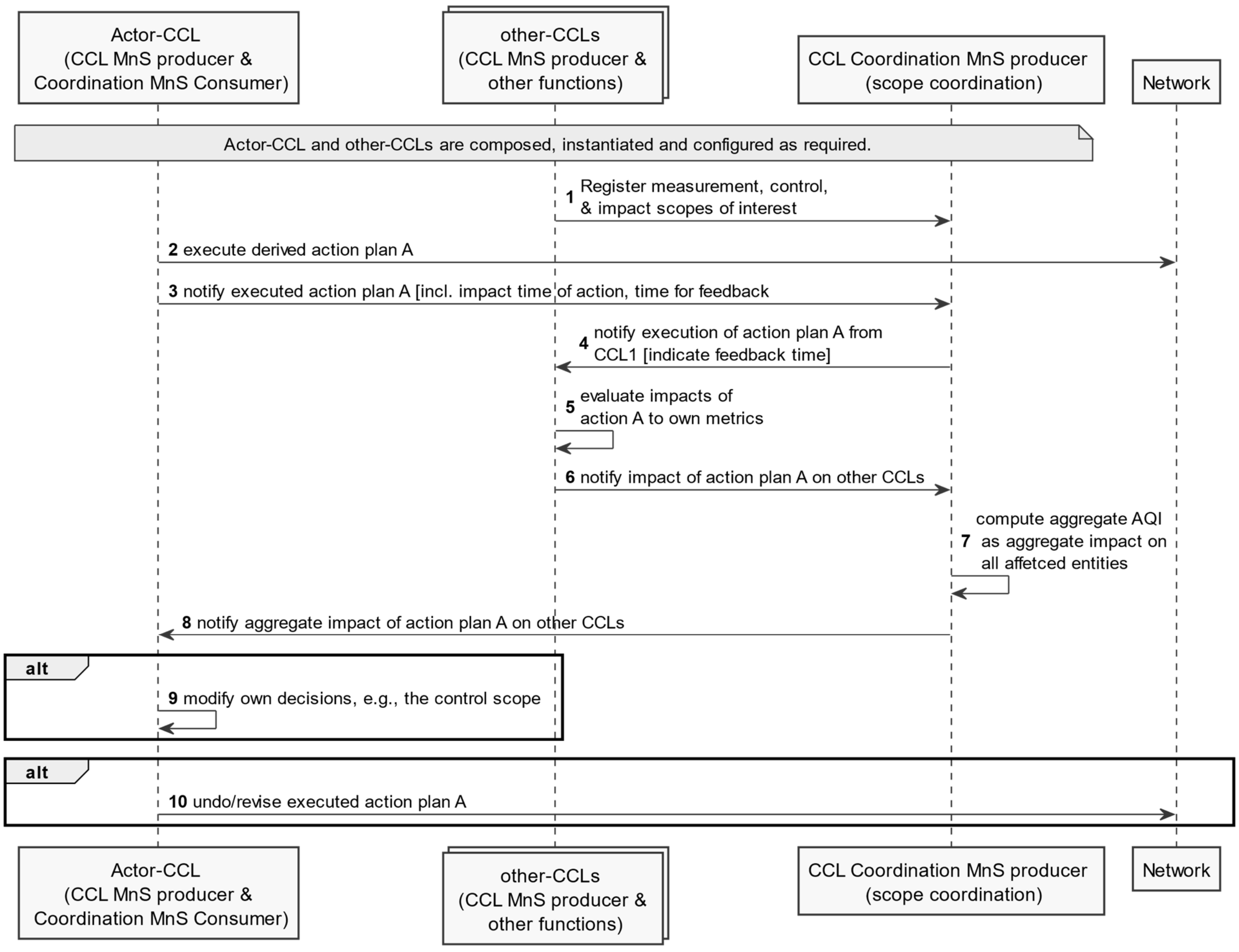

Actor-CCL (CCL MnS producer & Coordination MnS Consumer)
other-CCLs (CCL MnS producer & other functions)
CCL Coordination MnS producer (scope coordination)
Network
Actor-CCL and other-CCLs are composed, instantiated and configured as required.
1 Register measurement, control, & impact scopes of interest
2 execute derived action plan A
3 notify executed action plan A [incl. impact time of action, time for feedback
4 notify execution of action plan A from CCL1 [indicate feedback time]
5 evaluate impacts of action A to own metrics
6 notify impact of action plan A on other CCLs
7 compute aggregate AQI as aggregate impact on all affetced entities
8 notify aggregate impact of action plan A on other CCLs
alt
9 modify own decisions, e.g., the control scope
alt
10 undo/revise executed action plan A

**Figure 7.5-1: CCL-impact assessment and actual metric-value conflicts resolution**

Step 0. The set of CCLs are composed, configured and instantiated.

Step 1 The CCLs register their scopes of interest to the coordination entity including the scopes where they take measurements, take control actions as well as where their actions are expected to impact. Where applicable, the scope have also been coordinated to ensure there are no conflicts for desired impacted scopes, the desired outcomes on the impacted scopes, cross impacts between measurement and control scopes.

Step 2. The acting CCL derives and executes an action plan onto the network.

Step 3. After an action, the CCL updates the executedAction so that notifications are sent to the CCL coordination entity to indicate that if an entity is affected, it should provide its feedback on the effect in a time not exceeding the CCL-action-impact time.

Step 4. The CCL coordination entity then notifies that respective affected entities, e.g. other CCLs or other management functions

Step 5. The impacted entity collects information on its metrics, (e.g., a using PM job), based on which, it computes its observed impact in form of an index, called the Action Quality Indicator (AQI), that describes and quantifies the observed impact, i.e. it indicates the degree to which the action was good or bad to their objectives.

Step 6. The impacted entity sends the AQI to the coordination entity within the CCL-action-impact time that was notified by the actor-CCL. The Action Quality Indicator is delivered to the actor-CCL in one of two ways:

- Impacted entity writes the AQI into an equivalent attribute called “reported AQIs” on the coordination entity, i.e., the coordination entity contains an attribute for the Action Quality Indicator to which each impacted entity writes its computed AQI. The reported AQIs attribute is a list to which each affected entity appends a value.

- The impacted entity contains an attribute for an observed AQI in which it writes the computed AQI. This can then be notified to the coordination entity which subsequently reads it to obtain the AQI for that impacted entity.

Step 7. The coordination entity determines the aggregate impact on all affected entities. To enable the CCL coordination entity to determine how much impact actor-CCL had on all the other CCLs together, the CCL coordination entity aggregates the impact based on the reported Action Quality Indicators from the respective impacted entities. The AQI can be computed as a weighted average of the AQI sent by the individual impacted entities. The AQI is delivered to the actor-CCL in one of two ways:

- The coordination entity writes the AQI into an equivalent attribute called “reported AQIs” on the actor-CCL, i.e., the actor-CCL contains an attribute for the Action Quality Indicator to which coordination entity writes the computed aggregate AQI.

- The CCL coordination entity contains an attribute for the aggregate AQI in which the coordination entity writes the computed aggregate AQI that is computed from the AQIs reported by the multiple affected CCLs. On modifying this aggregate AQI attribute, this can then be notified to the actor-CCL which subsequently reads it to obtain the aggregate AQI.

Step 8. The coordination entity sends the aggregate AQI to actor-CCL which is then used by the actor-CCL to decide an appropriate action to minimise the impact.

Step 9. The actor-CCL evaluates the impacts and if needed, the way it makes its decisions. For example, the actor-CCL can adjust the control scope (i.e., the acceptable range of values) on a given parameter.

Step 10. actor-CCL can revise the previous actions if needed. If it is computed by the CCL coordination entity, the coordination entity notifies it to the actor-CCL and may also propose a response action, e.g. to reverse the action that was taken.

NOTE: The data models for executedAction, reportedAQIs need to be extended.

## 7.6 CCL Scope conflicts avoidance, detection and resolution

To coordinate scope assignments, a CCL coordination functionality, say in CCL Coordination entity, needs a capability to coordinate the scope assignment across multiple CCLs, say called the scope assignment coordination capability. The scope assignment coordination capability considers a defined full scope space Sp and a set of scope rules to define the best scope to be assigned to each CCL. An example rule may be that the defined CCL scope should not overlap. The rules may for example be defined by an operator or can be implementation specific depending on the types of CCLs that are to be configured.

Each CCL has four scopes - the measurement scope, target scope, control scope and impact scope, all of which can configured by the MnS consumer, i.e., an operator or the CCL Coordination entity may derive the required scope and configure it onto the CCL. There maybe different rules that each scope definition should adhere to for a given use case. The CCLs register their scopes with the CoordinationEntity (e.g., for the case where the scope is not defined by the CCL Coordination entity). The notification of the CCL scope to the CCL Coordination entity triggers an evaluation of potential conflict, i.e. whether those scopes are likely to conflict with the scopes of another CCL. The potential conflicts can be confirmed as actual conflicts by the CCL or the Coordination entity which then triggers resolution by computing a new reassignment of scopes.

To assign scopes, the scope coordination capability Applies the scope assignment rules defined in the scope coordination capability and divides the scope space into regions such that each region is matched to a CCL in a way that maximizes fulfilment of the assignment rules. The For example, if the benefit is to avoid overlaps, the subregions are assigned to the different CCLs in a way that ensures no overlaps and that all the scope space has been assigned.

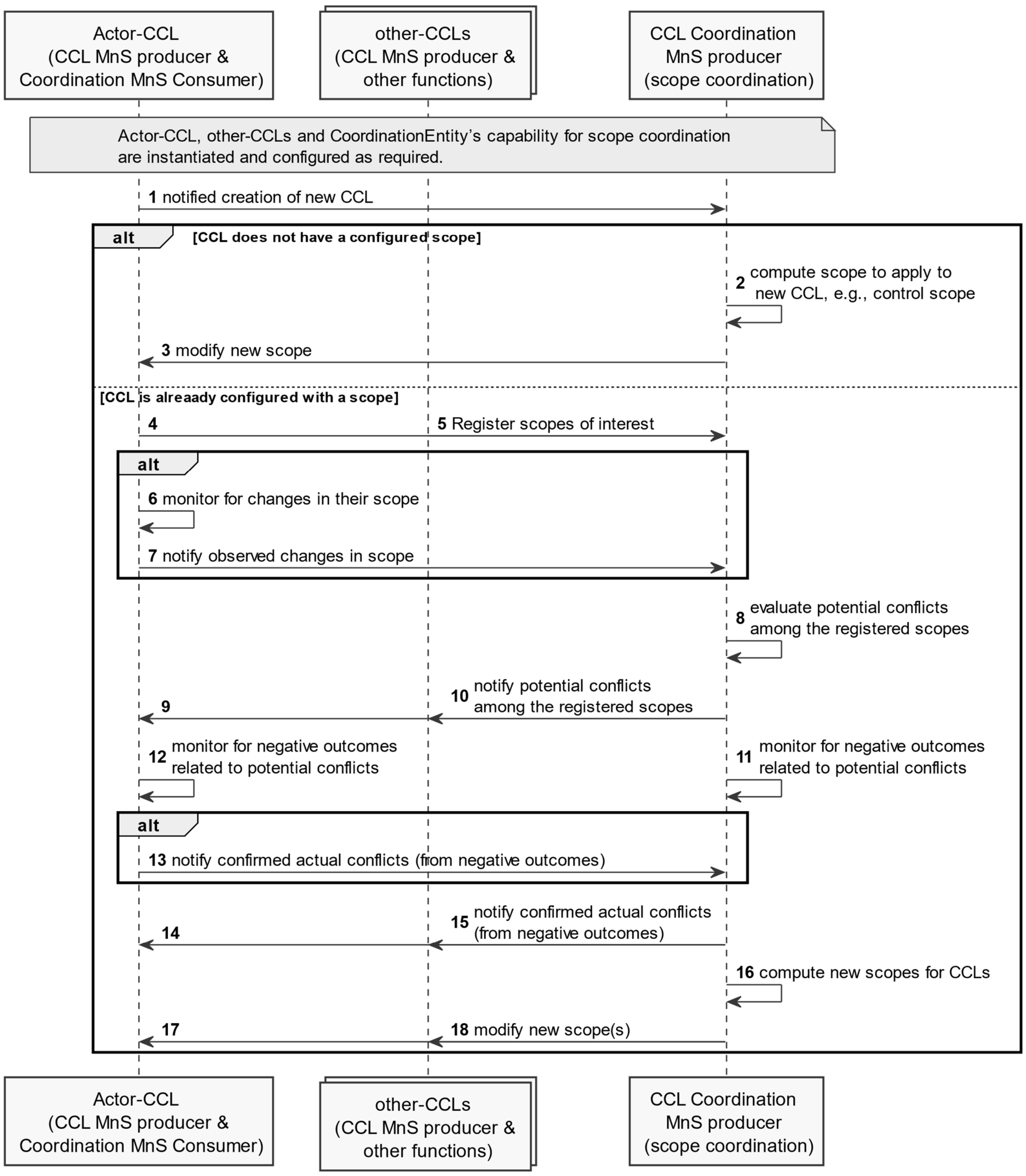

Actor-CCL
(CCL MnS producer &
Coordination MnS Consumer)
other-CCLs
(CCL MnS producer &
other functions)
CCL Coordination
MnS producer
(scope coordination)
Actor-CCL, other-CCLs and CoordinationEntity's capability for scope coordination are instantiated and configured as required.
1 notified creation of new CCL
alt
[CCL does not have a configured scope]
2 compute scope to apply to new CCL, e.g., control scope
3 modify new scope
[CCL is alreaady configured with a scope]
4
5 Register scopes of interest
alt
6 monitor for changes in their scope
7 notify observed changes in scope
8 evaluate potential conflicts among the registered scopes
9
10 notify potential conflicts among the registered scopes
12 monitor for negative outcomes related to potential conflicts
11 monitor for negative outcomes related to potential conflicts
alt
13 notify confirmed actual conflicts (from negative outcomes)
14
15 notify confirmed actual conflicts (from negative outcomes)
16 compute new scopes for CCLs
17
18 modify new scope(s)

**Figure 7.5-1: CCL-impact assessment and actual metric-value conflicts resolution**

Step 0-1. The CoordinationEntity's capability for scope coordination is instantiated and configured ( e.g., with the rules for evaluating and coordinating scopes for different use cases)

Step 0-2. The set of CCLs are composed, configured and instantiated;

Step 1. Instantiation of a new CCL is notified to the CoordinationEntity.

Alternative: the CCL does not have a configured scope and the CoordinationEntity needs to assign the scope

Step 2. the CoordinationEntity computes the scope to be applied by the new CCL, e.g., it divides the scope space into regions matched to the CCLs e.g. to ensure no overlaps and that all the scope space has been assigned.

Step 3. the CoordinationEntity notifies the new CCL of the assigned scope

Otherwise

Step 4, 5. The CCLs register their scopes of interest to the coordination entity including the scopes where they take measurements, take control actions as well as where their actions are expected to impact. The CCL may register by writing into the `toBeCoordinatedCCLScopes` attribute of the CoordinationEntity.

Step 6. The CCLs monitor for changes in their scope to detect misalignments in scope

Step 7. If the scope is changed, the CCL registers the observed changes in the scope to the CoordinationEntity's scope coordination capability. The CCL registers differences between what was configured and the actual scopes e.g., if the considered scope for taking measurement data are affected by the actions of another CCL. The CCL may register by writing into the `toBeCoordinatedCCLScopes` attribute of the CoordinationEntity.

Step 8. A registration of scope or scope changes triggers the CoordinationEntity to evaluate if there are any potential conflicts among the registered scopes,

Step 9, 10. If scope conflicts are potential detected, the CoordinationEntity notifies the CCLs of the potential conflicts, so that both the CCLs and the CoordinationEntity monitor to see if potential scope conflicts results into actual conflicts. The CoordinationEntity adds a new entry in the detectedScopeConflicts list with a value of POTENTIAL_CONFLICT for the conflictType .

Step 11, 12. CCLs and the CoordinationEntity monitor to see if there are negative outcomes.

Step 13. If negative outcomes are observed by a CCL, the CCLs notifies the CoordinationEntity of the confirmed scope conflicts. The CCL updates the conflictType value of the detectedScopeConflicts entry from potential to actual conflict.

Step 14, 15. Alternatively, if the negative outcomes are observed by the CoordinationEntity, the CoordinationEntity notifies all affected CCLs of the confirmed scope conflicts. The CoordinationEntity updates the conflictType value of the detectedScopeConflicts form potential to actual conflict.

Step 16. The CoordinationEntity computes new scopes to be applied by the different CCLs, e.g., similar to an initial assignment.

Step 17, 18. If there are CCLs whose scope should be revised, the CoordinationEntity notifies the CCLs whose scope is revised of the newly computed scope.

## 7.7 CCL Trigger-time conflicts avoidance, detection and resolution

CCL could require to operate in a hierarchy. For example, to ensure that handovers are always optimal, a CCL on handover optimization may need to be triggered every after a CCL on Energy saving has been executed to be sure that there are appropriate handover relations even when some cells may have been disabled. The handover CCL would be in lower hierarchy to the Energy saving CCL. Each CCL (say CCL-A) has an operational profile (described in the CCL purpose) that includes the level of hierarchy and describing characteristics under which the CCL-A operates, e.g. when or after which precedent CCLs this CCL-A should be executed.

A CCL may be involved in more than 1 hierarchies or within a single hierarchy, the CCL may relate to multiple other CCLs operating on related scope in one or more domains, e.g., on the same or related managed objects. The CoordinationEntity needs to configure the appropriate hierarchy for the CCLs considering the different relationships. For a CCL C3 with relationships to 2 other CCLs C1 and C2, considering the hierarchies defined in the operational profiles P1 and P2 of the CCLs C1 and C2, the CoordinationEntity evaluates the description of CCL C3 against at least one of the profiles P1 and P2 and accordingly determines and configures the operational profile of CCL C3.

There are cases where it is not clear which CCL should be triggered, e.g. if there are multiple degraded KPIs. The CoordinationEntity may evaluate the network state of a given network scope to diagnose what the problem might be occurring. Alternatively, it may obtain an analytics report for that problem. For the identified problem, the CoordinationEntity finds the most appropriate CCL to trigger. The CoordinationEntity queries the capabilities of the available CCLs to match the identified problem to one of the CCLs. The identified CCL is then triggered to find an appropriate response for the problem.

To coordinate within each hierarchy, when a CCL C1 generates an action, it needs to informs the related CCL C2 of such an action, so that CCL C2 the considers the actions of CCL C1 in determining C2's actions to be taken on the shared or related managed scope. Accordingly, C1 actions should be notified to the CoordinationEntity so that the CoordinationEntity indicates them to C2 when triggering C2.

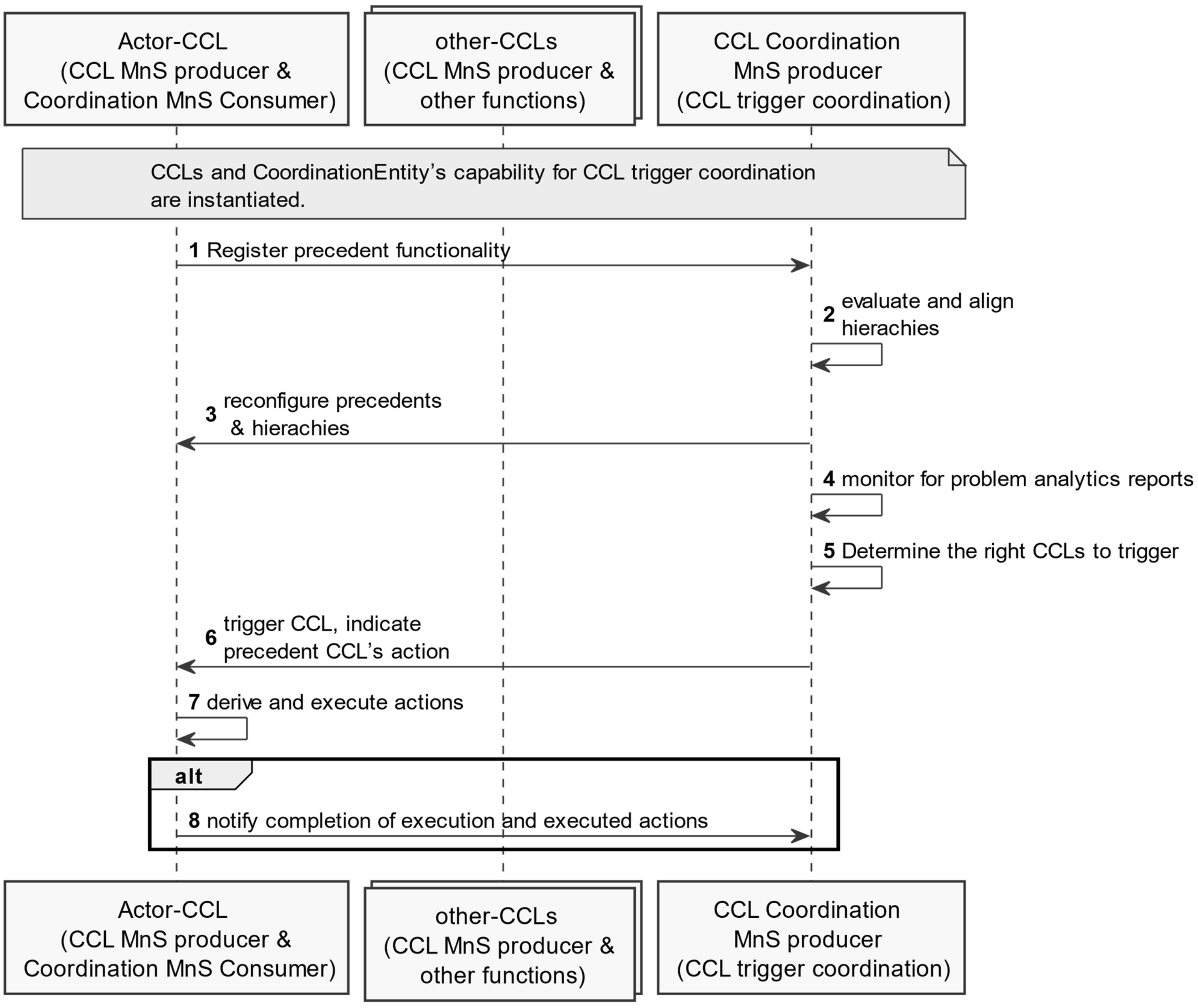

Actor-CCL (CCL MnS producer & Coordination MnS Consumer)
other-CCLs (CCL MnS producer & other functions)
CCL Coordination MnS producer (CCL trigger coordination)
CCLs and CoordinationEntity's capability for CCL trigger coordination are instantiated.
1 Register precedent functionality
2 evaluate and align hierachies
3 reconfigure precedents & hierachies
4 monitor for problem analytics reports
5 Determine the right CCLs to trigger
6 trigger CCL, indicate precedent CCL's action
7 derive and execute actions
alt
8 notify completion of execution and executed actions

**Figure 7.5-1: CCL-impact assessment and actual metric-value conflicts resolution**

Step 0. The CoordinationEntity's capability for CCL trigger coordination is instantiated and configured. The set of CCLs are composed, configured and instantiated but not triggered to evaluate the network or execute actions.

Step 1. Each CCL registers its precedent functionality which is the set of higher hierarchy automation functionality or CCLs, for which after their execution this CCL should be executed.

Step 2. The CoordinationEntity evaluates the sets of precedent functionality to align hierarchies of the CCLs and determine if there is need to configure the hierarchies.

Step 3. If reconfiguration is needed, the CoordinationEntity (re)configures the operational profiles of the CCLs, e.g. hierarchies and relations among the CCLs.

Step 4. The CoordinationEntity analyses network problem scope or obtains analytics report on network problem.

Step 5. If a problem is identified, the CoordinationEntity evaluates what the most appropriate CCL to be triggered should be.

Step 6. If the CoordinationEntity has identified a new CCL to trigger or a previous execution in a hierarchy has been completed, the CoordinationEntity triggers the CCL identified as most appropriate, i.e., it toggles the `administrativeState` from LOCKED to UNLOCKED.

Step 7. The triggered CCL generates and executes its desired action

Step 8. The triggered CCL notifies its action to the CoordinationEntity for onward transfer when triggering lower hierarchy CCLs

## 7.8 CCL concurrent-actions conflicts avoidance, detection and resolution

Since each CCL focuses on a smaller scope of the network problem space, several CCLs may need to be executed. To avoid collision of their actions in a given network scope, the CCLs can be explicitly scheduled by the CCL coordination entity using the CCL Trigger-time conflicts avoidance, detection and resolution mechanisms in clause 7.8.

Even then it may not be the case that only 1 CCL is active within a given scope. To minimize conflicts (e.g. where the scopes overlap), the CCLs align their action plans through the CoordinationEntity that identifies possibilities for potential conflicts based on which it selects which action plan should be executed and when to minimize the potential conflicts. The CoordinationEntity acts as supervisory action-critic functionality that oversees the actions of the different CCLs to look out to good performance across the several CCLs. It receives desired action from the CCLs, evaluates them to see if they overlap with other proposed changes from other CCLs; and what their likely effects may be.

The CCLs inform the CoordinationEntity about their respective action plans. The action plans contain information of target resources, scheduled time for execution, and may include other additional information such as historical results of the proposed actions. The CoordinationEntity assesses each plan to determine the likely impacts. An example analytics involves discretizing the state of the network into discrete scenarios onto which the planned actions are superimposed and then marked with particular performance. Example discrete scenarios can be whether the network ends in a state of low traffic and normal performance or scenario of normal traffic and anomalous performance. Where there are likely conflicts (i.e., likely undesired impacts), the CoordinationEntity decides the changes that should be executed on the network to minimize or avoid concurrent actions on the same resources.

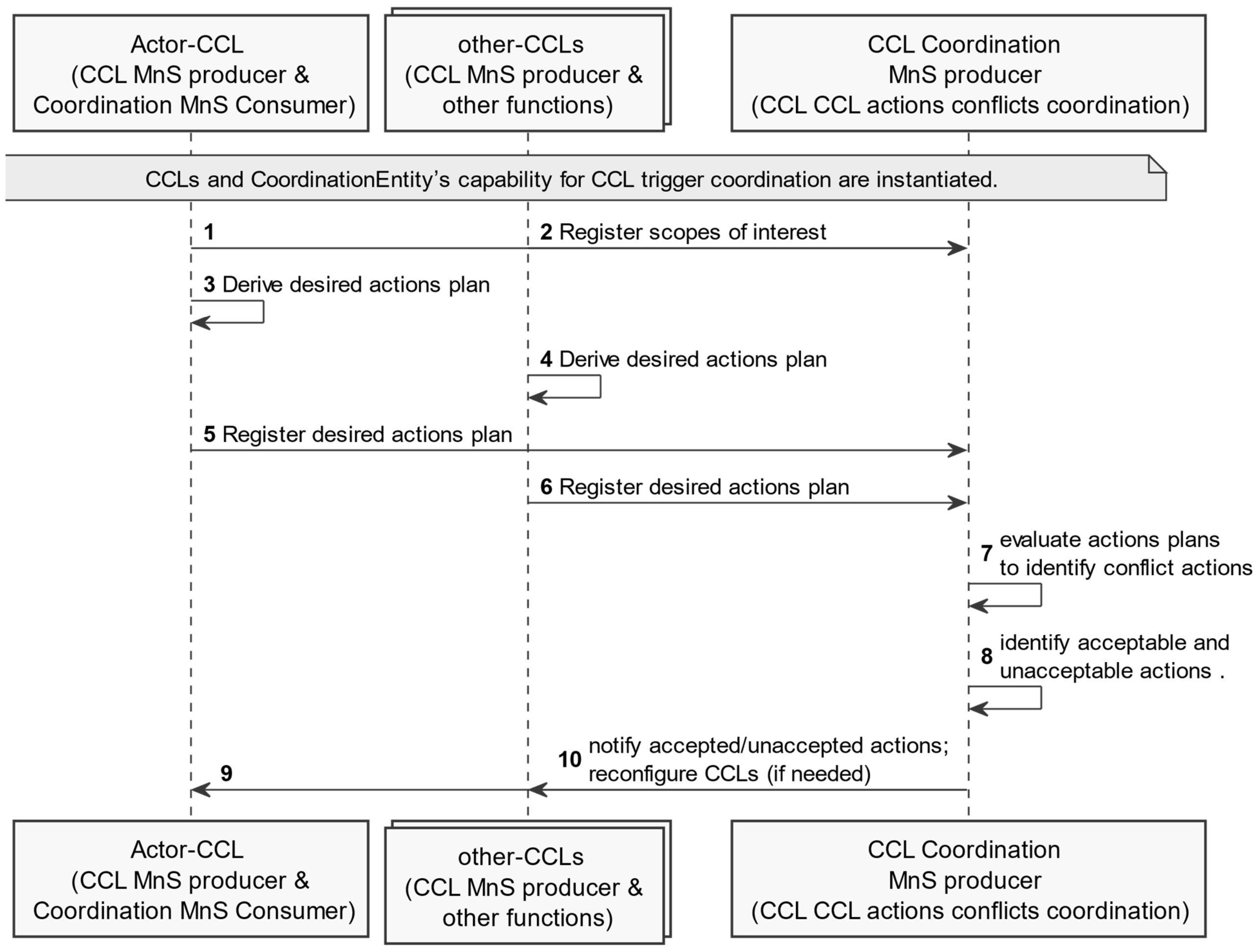

Actor-CCL (CCL MnS producer & Coordination MnS Consumer)
other-CCLs (CCL MnS producer & other functions)
CCL Coordination MnS producer (CCL CCL actions conflicts coordination)
CCLs and CoordinationEntity's capability for CCL trigger coordination are instantiated.
1
2 Register scopes of interest
3 Derive desired actions plan
4 Derive desired actions plan
5 Register desired actions plan
6 Register desired actions plan
7 evaluate actions plans to identify conflict actions
8 identify acceptable and unacceptable actions .
10 notify accepted/unaccepted actions; reconfigure CCLs (if needed)
9

**Figure 7.8-1: CCL-impact assessment concurrent actions conflicts resolution**

Step 0. The set of CCLs and the CoordinationEntity's capability for actions conflicts coordination is instantiated and configured ( e.g., with the rules for evaluating and coordinating scopes for different use cases).

Step 1,2 The CCLs register their scopes of interest to the coordination entity including the scopes where they take measurements, take control actions as well as where their actions are expected to impact. Where applicable, the scope have also been coordinated to ensure there are no conflicts for desired impacted scopes, the desired outcomes on the impacted scopes, cross impacts between measurement and control scopes.

Step 3. The CCLs derive their desired action plans that needs to be coordinated prior to execution. The action plan is the combination of a set of actions that can be taken and the scopes under which those actions can be applied

Step 5,6. The CCLs register their desired action plans to the CoordinationEntity. The CCL writes into the desiredCCLActions attribute on the CoordinationEntity. The CCL may add the desired actions onto the toBeCoordinatedActionPlans.

Step 7. The CoordinationEntity assesses the plans to see if they overlaps with one another; and what the likely effects of the overlaps may be.

Step 8. If potential conflicts are detected (e.g., from likely undesired impacts), the CoordinationEntity decides the changes that should be executed, e.g., be based on the priorities of the CCLs required metric values

Step 9.The CCL coordination entity then provides feedback to the CCL instance (s) regarding their recommended actions,including information on which actions can be executed or not and on the expected effects of the CCLs actions. Feedback may also include redefining the allowed control parameter spaces and ranges of the individual

CCLs (i.e. which parameters the CCL should not control any further or the range in which the CCL may set the value of a control parameter).

## 7.9 CCL non-concurrent actions conflicts avoidance, detection and resolution

### 7.9.1 Detection and avoidance of non-concurrent actions conflicts

Non-concurrent actions conflicts differ from concurrent actions conflicts in that for non-concurrent actions conflicts, the other CCLs have already taken their actions and are considered stable when an actor CCL initiates its actions. Accordingly, the potential non-concurrent actions conflicts cannot be observed from the desired action plans, but can be detected from overlaps with previously executed actions. The CCL intending to take an action, sends its desired action plans to the coordination entity prior to execution of those configuration changes. The coordination entity checks the configuration changes against other CCLs (that have been executed) to detect potential conflicting actions. If overlaps are detected, the potential direct-actions conflicts can be avoided if the CCL coordination entity notifies the detected conflict(s) to the related CCLs which then adjust the planned configurations to a new set that could have less conflicts.

The potential conflicts can be confirmed as actual via their counter-productiveness due to known or unknown interdependence between their actions, e.g., when they change the same parameter one after the other. A CCL instance A that is likely to be affected, needs to monitor a specific scope or context that could be affected by another CCL instance B. And CCL B knows that CCL A may take actions that could affect the same scope as CCL B. CCL B can provide a conflict monitoring context/scope to CCL A informing CCL A about CCL B's latest actions on the managed entity and its tolerance limits that should be maintained for the parameters and metrics in this managed entity.

Based on scopes of interest registered by the CCLs, the CoordinatorEntity informs the other CCLs instances which have related scopes. The CCL with previous actions inform the actor CCL (via the CoordinationEntity) about their latest actions on the managed entity and their tolerance w.r.t to its parameters and metrics in this managed entity. CCL A observes the conflict monitoring context, so that if it observes the violations of the said tolerances, it reports the conflict to the CCL B– again either directly to B or via the coordination CCL.

NOTE: If CCL A is able to predict violation prior to activating the actions, CCL A inform CCL B of the predicted impact at the time when the action is being activated. Otherwise, CCL a first observes the impacts and ten informs CCL B.

### 7.9.2 Resolution of potential non-concurrent actions conflicts

If the conflict is confirmed, i.e., two or more CCLs want different values for the same parameter and the parameter cannot be assigned to only one CCL, the CoordinationEntity should compute a compromise value for the parameter, a value which can be considered to be equally good for all the CCLs. To ensure that the CoordinationEntity understands the importance of the parameter to each CCL, the CCLs provide their usefulness of the parameter to the coordinator CCL. The usefulness provided by a CCL shows the relative goodness of different values of the parameter to the CCL in a pre-defined scale, e.g. [0:1]. Since all the CCLs used the same scale, when the CCL coordinator selects a parameter value, it can clearly understand how important this value is for each CCL. The CoordinationEntity can then derive the compromise values which is then (provided to the CCLs to be) executed onto the managed object. An example way to compute the compromise is to use the Nash Social Welfare Function since it provides equal fairness to all competing entities.

A compromise based only on usefulness does not consider the relative (level of) interest of the CCLs in the parameter. To account for the interests, the CCLs should provide to the CCL coordinator their relative interest in the parameter, so that the computed compromise value accounts for the combined interests of the CCLs. The relative interest may be computed based on a fixed scale. For example, for a CCL on cell interference management on a scale of [0-10], a cell's transmit power has a goodness of say 9 than the cells load which has a goodness of 3.

NOTE 1: The CCL coordination entity does not have to calculate the compromise value all the time as this requires information exchange among the CCLs and computational energy. It should be possible to configure the CCL coordination entity such that it calculates the compromise values only when certain conditions are met. The CCL coordination entity should be able to expose required services to the MnS consumer to configure such conditions.

NOTE 2: For a given CCL, the usefulness may be equivalent to the level of interest, but it is not always the case. It is possible that a CCL has high interest in a parameter that has low usefulness.

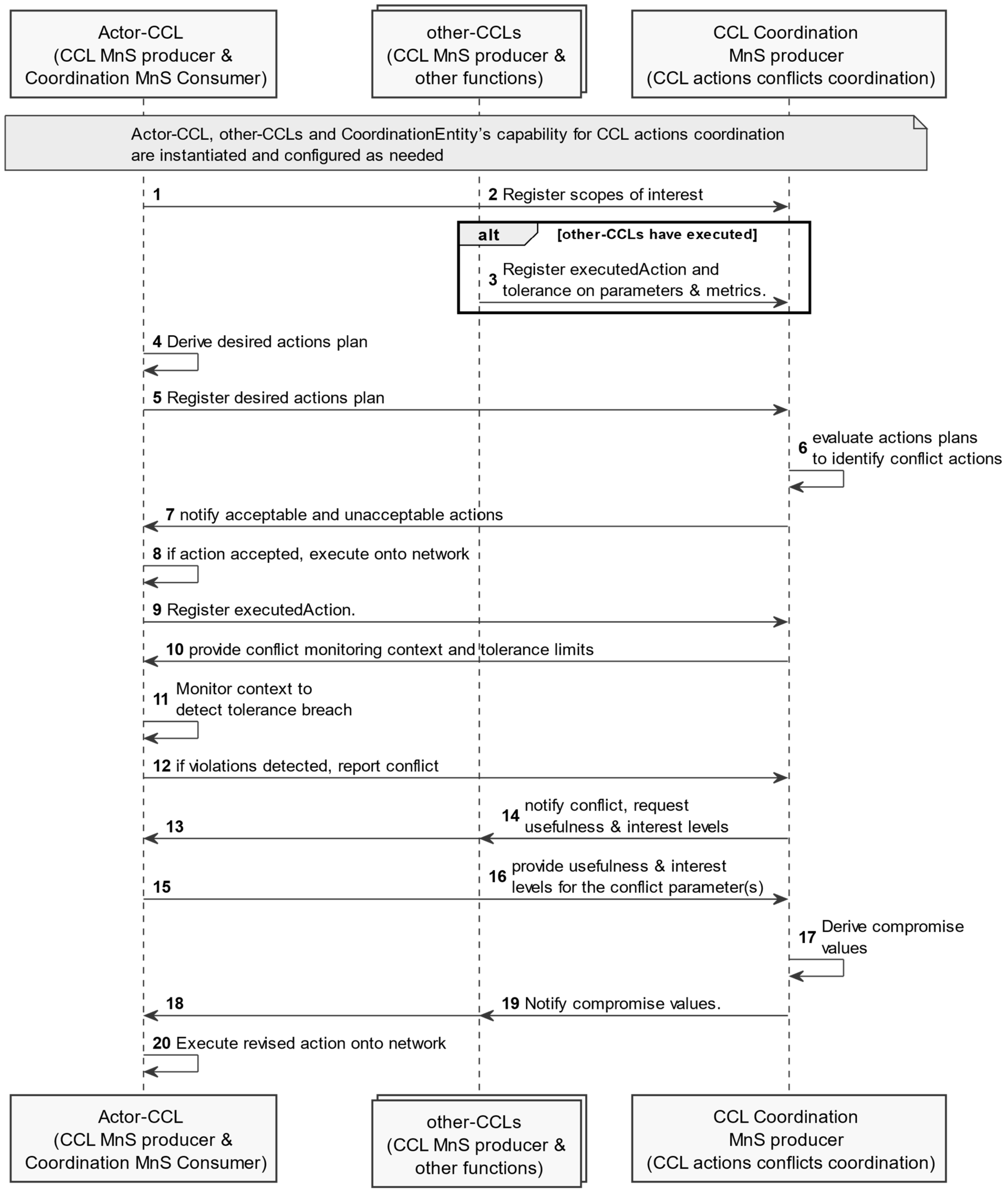

Actor-CCL (CCL MnS producer & Coordination MnS Consumer)
other-CCLs (CCL MnS producer & other functions)
CCL Coordination MnS producer (CCL actions conflicts coordination)
Actor-CCL, other-CCLs and CoordinationEntity's capability for CCL actions coordination are instantiated and configured as needed
1
2 Register scopes of interest
alt [other-CCLs have executed]
3 Register executedAction and tolerance on parameters & metrics.
4 Derive desired actions plan
5 Register desired actions plan
6 evaluate actions plans to identify conflict actions
7 notify acceptable and unacceptable actions
8 if action accepted, execute onto network
9 Register executedAction.
10 provide conflict monitoring context and tolerance limits
11 Monitor context to detect tolerance breach
12 if violations detected, report conflict
13
14 notify conflict, request usefulness & interest levels
15
16 provide usefulness & interest levels for the conflict parameter(s)
17 Derive compromise values
18
19 Notify compromise values.
20 Execute revised action onto network

**Figure 7.9-1: CCL-impact assessment and actual metric-value conflicts resolution**

Step 0. The set of CCLs and the CoordinationEntity's capability for actions conflicts coordination is instantiated and configured ( e.g., with the rules for evaluating and coordinating scopes for different use cases)

Step 1,2. The CCLs register their scopes of interest to the coordination entity including the scopes where they take measurements, take control actions as well as where their actions are expected to impact. Where applicable, the scope have also been coordinated to ensure there are no conflicts for desired impacted scopes, the desired outcomes on the impacted scopes, cross impacts between measurement and control scopes.

Step 3. if previous CCLs have executed actions, the CCLs register to the CoordinationEntity, the executedAction and their tolerance on the related parameters and metrics. The tolerance indicates the performance limits which CCL B would like CCL A to respect. CCL A (the context recipient CCL) should work within these bounds, i.e. its actions should not violate the said tolerances to avoid counter-productiveness.

Step 4. The actor CCL derives its desired action plan that needs to be coordinated prior to execution. The action plan is the combination of a set of actions that can be taken and the scopes under which those actions can be applied

Step 5. The actor CCL registers its desired action plans to the CoordinationEntity. The CCL writes into the desiredCCLActions attribute on the CoordinationEntity.

Step 6. The CoordinationEntity evaluates the action plans against previously executed and accepted actions if the new action plan overlaps with those plans; and what the likely effects of the overlaps may be.

Step 7. The CoordinationEntity provides feedback to the actor CCL instance indicating if the action is not accepted (the action overlaps with previous actions indicating a potential conflict) or is accepted.

Step 8. If the action is accepted, the CCL executes action on to the network

Step 9. The actor CCL registers its executedAction to the CoordinationEntity.

Step 10. If the action is accepted and executed, the CoordinationEntity informs the actor CCL of the conflict monitoring context and related performance tolerance limits of the related CCLs. This serves as the request to the actor CCL to monitor the provided context for counter productiveness.

Step 11. The actor CCL monitors the provided context to see if it observes violations of the said tolerances

Step 12. If violations of the tolerances are observed, the actor CCL reports the conflict to the CoordinationEntity for onward notification to the other CCLs.

Step 13, 14. The CoordinationEntity indicates all CCLs that a conflict is observed. This is an indication that the conflict needs to be resolved, so the CCLs should provide their usefulness and level of interest on the values for the parameters in conflict.

Step 15, 16. The CCLs provide their usefulness and level of interest for the parameter to the CoordinationEntity to be used to compute a compromise.

Step 17. The CoordinationEntity derives the compromise values, e.g., using the Nash Social Welfare Function

Step 18, 19. The CoordinationEntity notifies the compromise value to the actor CCL to be executed onto the managed object.

Step 20. The CCL may re-execute the revised action on to the network

## 7.10 CCL metric-value conflicts avoidance and detection

### 7.10.1 Avoiding concurrent and non-concurrent metric-values conflicts

Each CCL has a control scope including a set of metrics. The metrics ma have prioritization among them, e.g. a handover optimization CCL may have more interest (higher priority) in controlling Cell individual offsets compared to controlling antenna tilts. To support detection and avoidance of potential non-concurrent metric-value conflicts, if the CCL has been pre-configured e.g., by the operator with the expected outcomes, the CCL may register its desired metrics, their priorities and outcomes with the CCLCoordinationEntity. This triggers the first evaluation for potential conflict, i.e. whether these metrics and outcomes are likely to conflict with those of another CCL.

Subsequently, potential metric-value conflicts are avoided using likely-impact of planned actions. For any CCL, large and frequent changes to network parameters may affect network stability since they increase the probability of occurrence of conflicts, i.e. avoiding making unnecessary configuration changes to the managed objects guarantees network stability and minimize the probability of conflicts between CCLs. This may then imply that executing large changes, e.g. to quickly improve the performance, in case of a poor decision, may also result in significant degradation. So, it is preferred to take small smooth changes in the case where the impact is not so clear, and only make the large changes when the CCL is sure that the impact is positive.

In case of a plan that results in a conflict, the CoordinationEntity sends its decision and possibly the failed criteria to the CCL - to either be executed or to be used to compute better decisions. It is assumed that based on feedback on the quality of its decisions, the CCL updates its decision-making engine and repeats the decision evaluation process. Then if the CCL has consistently made good large action-decisions, the coordinator CCL can consider the CCL as trusted to make such large decisions. The coordinator CCL informs the CCL that the CCL has consistently made good decisions and achieved its ultimate trust.

## 7.10.2 Detecting concurrent metric-values conflicts

For metric-values conflicts where actions are executed in a short interval form one another, detection can be possible. Two CCLs (CCL1 and CCL2) may optimize 2 target metrics Y1 and Y2, e.g. one intending to ensure "HO failure is < 2 %" while the other wants "SINR > 10dB". Due to coupling between Y1 and Y2, actions to optimize these by CCLs may lead to correlated oscillations/degradations in Y1 or Y2. The correlated oscillations indicate a potential conflict, but the CCLs may not see the oscillations in the metric that is not of their interest. The CoordinationEntity analyses the behavior of Y1 and Y2 to see if there are correlated oscillations as result of actions by any of the CCLs which then indicates potential conflict between CCL1 and CCL2. When the oscillations are observed, the CoordinationEntity informs the related CCLs (i.e. CCL1 and CCL2) about the detected potential conflict.

For detected potential conflict the CCL coordination service producer needs to confirm that it is an actual harmful conflict. This can be determined based on the severity of degradation in the performance metrics of the related CCLs. The threshold to determine the severity may be defined by the MnS consumer (e.g. the operator or coordinator CCL). If the degree of degradation is higher than the threshold then it is a confirmed conflict that requires resolution. Otherwise, no action is needed.

## 7.10.3 Detecting non-concurrent metric-values conflicts

For actions that are not executed within the same time frame, there are no correlated oscillations in the metrics, so the CCLs should detect potential conflicts themselves. The CCLs attempt to fulfil desired outcomes, and where they ae unable to, the CCL sends feedback to the CoordinationEntity indicating which outcomes on which metrics cannot be fulfilled. A CCL may for example indicate that there are ping-pong effects on a target, i.e. whenever the target is pushed in a given direction, it flips back to a previous state. The flipflop is an indication of a potential conflict which the CCL should notify to the CoordinationEntity. The CCL should notify the CoordinationEntity, e.g., the response could be that “desired outcomes on metric x cannot be achieved because it causes problems on higher priority metric y.”. Based on the feedback, the CoordinationEntity can confirm the existence of conflict, e.g. that other CCLs are requesting to readjust related parameters. The CoordinationEntity derives recommendations to the CCL including whether the CCL should change the prioritisation of its desired control metrics. The CoordinationEntity notifies the proposed changes to the CCL including setting control metrics or their priorities.

Note the resolution of concurrent and non-concurrent metric-values can apply the procedure for CCL-impact assessment and metric conflicts resolution as described in clause 7.5.

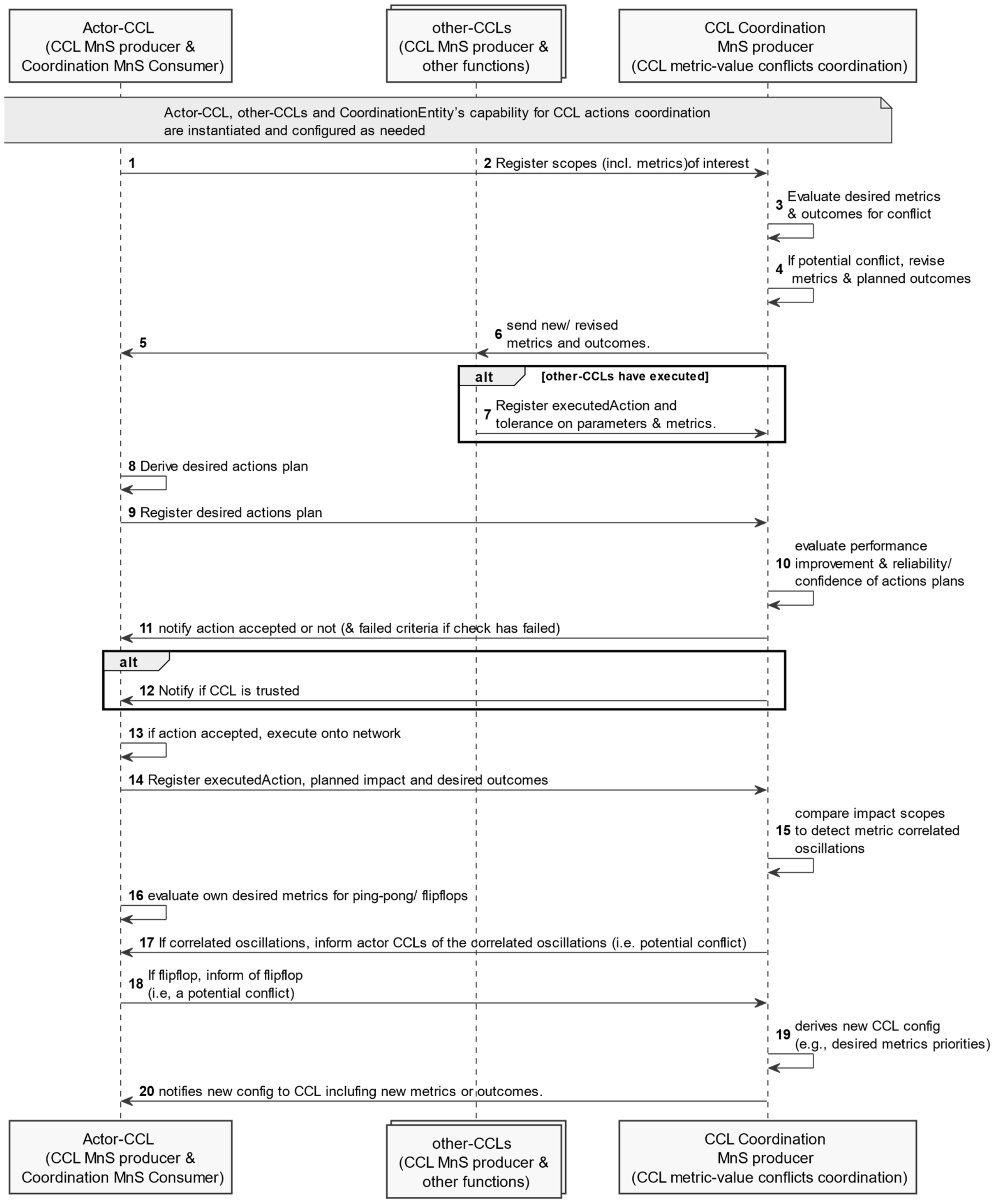

Actor-CCL (CCL MnS producer & Coordination MnS Consumer)
other-CCLs (CCL MnS producer & other functions)
CCL Coordination MnS producer (CCL metric-value conflicts coordination)
Actor-CCL, other-CCLs and CoordinationEntity's capability for CCL actions coordination are instantiated and configured as needed
1
2 Register scopes (incl. metrics)of interest
3 Evaluate desired metrics & outcomes for conflict
4 If potential conflict, revise metrics & planned outcomes
5
6 send new/ revised metrics and outcomes.
alt [other-CCLs have executed]
7 Register executedAction and tolerance on parameters & metrics.
8 Derive desired actions plan
9 Register desired actions plan
10 evaluate performance improvement & reliability/ confidence of actions plans
11 notify action accepted or not (& failed criteria if check has failed)
alt
12 Notify if CCL is trusted
13 if action accepted, execute onto network
14 Register executedAction, planned impact and desired outcomes
15 compare impact scopes to detect metric correlated oscillations
16 evaluate own desired metrics for ping-pong/ flipflops
17 If correlated oscillations, inform actor CCLs of the correlated oscillations (i.e. potential conflict)
18 If flipflop, inform of flipflop (i.e, a potential conflict)
19 derives new CCL config (e.g., desired metrics priorities)
20 notifies new config to CCL inclufing new metrics or outcomes.

**Figure 7.5-1: CCL metric-value conflicts avoidance and detection**

Step 0 The set of CCLs and CoordinationEntity's capability for metric-values conflicts coordination is instantiated and configured ( e.g., with the rules for evaluating and coordinating scopes for different use cases)

Step 1,2. The CCLs register their scopes of interest to the coordination entity including the scopes where they take measurements, take control actions, where their actions are expected to impact and their desired outcomes on those impact scopes. Where applicable, the scope have also been coordinated to ensure there are no conflicts for desired impacted scopes, the desired outcomes ion the impacted scopes, cross impacts between measurement and control scopes.

Step 3. The CCL coordination entity evaluates the metrics of interest and desired outcomes to see if they conflict with other CCLs. For example, based on the defined general objectives for the network scope (e.g. derived form an intent), the CCLCoordinationEntity may select the appropriate metrics and outcomes for the CCL.

Step 4. In case of a potential conflict, the CCL coordination entity derives revisions in the assigned metrics of interest and planned outcomes, to minimize contradictions or conflicts among the metrics and outcomes.

Step 5,6. The CCL coordination entity sends the selected new or revised metrics and outcomes to each CCL.

Step 7. Previous CCLs that have executed actions have registered to the CoordinationEntity, their executed actions, the scopes they expect to impact and their desired outcomes on those impact scopes.

Step 8. The actor CCL derives its desired action plan on to the network

Step 9. The actor CCL registers to the CoordinationEntity its desired action plan and the expected impact of that action plan (its claimed/predicted performance improvement) and reliability/confidence in that action/decision to be evaluated for potential significant degradation, i.e., that the actions are no unnecessarily too large.

Step 10 The coordinator CCL evaluates the claimed performance improvement and reliability/confidence to determine if the action should be allowed or not to avoid counter-productive actions - CCL making large changes, should have high reliability/ confidence and significant improvement in performance.

Step 11. The CoordinationEntity sends to the actor CCL its decision and the failed criteria in case the action plan has failed the evaluation For this, the CoordinationEntity updates the `proposedReviseddActionPlan` which is then notified to the respective CCL

Step 12. The coordinator CCL may also inform the CCL that the CCL has consistently made good decisions and achieved its ultimate trust. The CCL would not need to recheck its decision for appropriateness of the step change . For this the coordinationEntity updates the TrustedCCLs attribute with the DN of the CCL that has achieved full trust. The change is then notified to the CCL

Step 13. If the action is accepted, the actor CCL executes its desired action plan on to the network

Step 14. The actor CCL registers the executed action plans to the CoordinationEntity including the scopes they expect to impact and their desired outcomes on those impact scopes. The CCL writes into the desiredCCLActions attribute on the CoordinationEntity.

Step 15. The CoordinationEntity evaluates the impact scopes of the previous CCLs to detect metric oscillations which indicate a potential conflict.

Step 16. The CCL evaluates its desired metrics to see if there are ping-pong/ flipflop effects. The flipflop is an indication of a potential conflict which the CCL should notify to the CoordinationEntity.

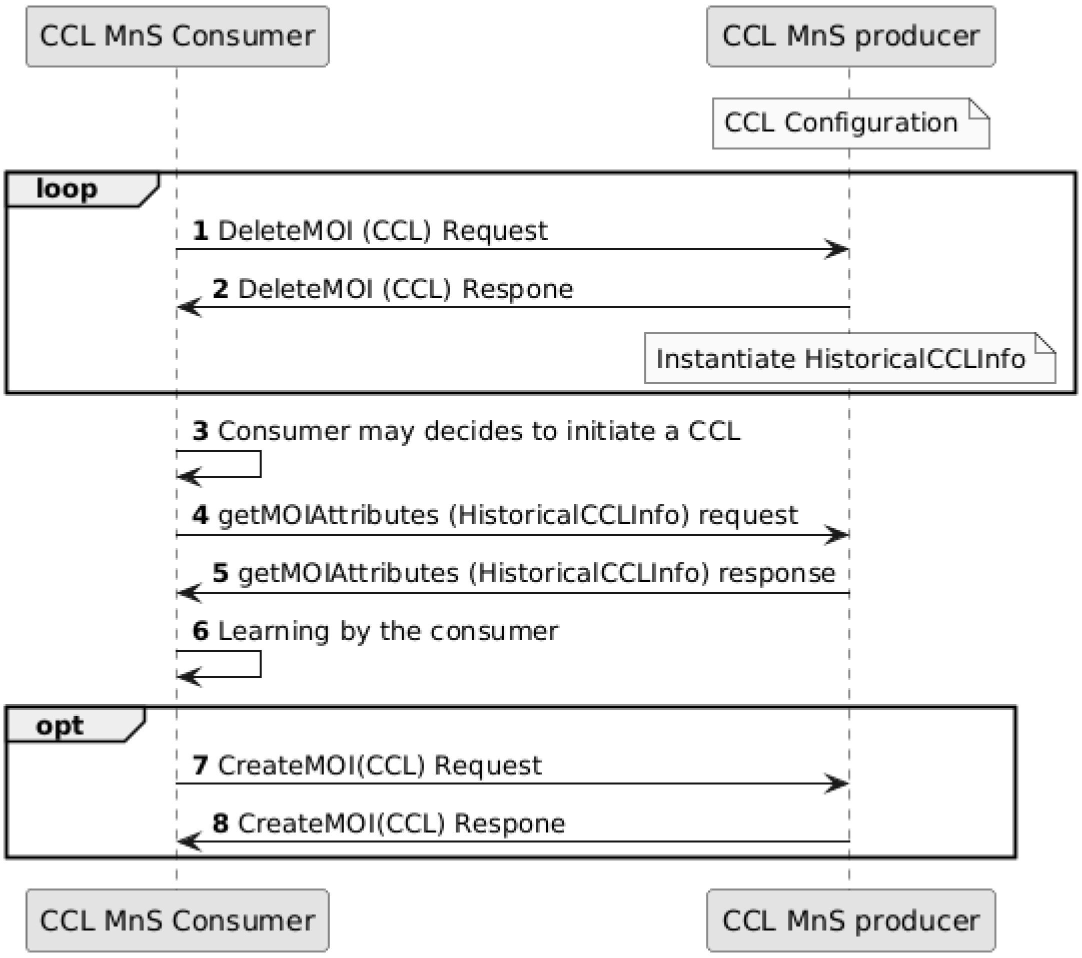
Step 17. In case of correlated oscillations, the CoordinationEntity informs the actor CCLs of the correlated oscillations indicating a potential conflict. For this, the coordinationEntity updates the `correlatedOscillation` attribute in the `metricValueConflict` added to the list of `observedMetricValueConflicts.` It then notifies the CCL of the `metricValueConflict`

Step 18. In case of flipflop, the CCL informs the CoordinationEntity of the flipflop indicating a potential conflict. For this the CCL adds the metric that is flipflopping as an entry in the flipflopMetrics attribute of the coordination entity.

Step 19. The CoordinationEntity derives recommendations to the CCL including whether the CCL should change the prioritisation of its desired control metrics

Step 20. The CoordinationEntity notifies the proposed changes to the CCL including setting control metrics or their priorities.

## 7.11 CCL creation based on Historical CCL data



CCL MnS Consumer
CCL MnS producer
CCL Configuration
loop
1 DeleteMOI (CCL) Request
2 DeleteMOI (CCL) Respone
Instantiate HistoricalCCLInfo
3 Consumer may decides to initiate a CCL
4 getMOIAttributes (HistoricalCCLInfo) request
5 getMOIAttributes (HistoricalCCLInfo) response
6 Learning by the consumer
opt
7 CreateMOI(CCL) Request
8 CreateMOI(CCL) Respone
CCL MnS Consumer
CCL MnS producer

**Figure 7.11-1: CCL creation based on Historical CCL data**

Producer instantiate and provision a CCL as defined in 3GPP TS 28.536

1. Consumer send DeleteMOI request for a CCL.
2. Producer sends a response. Producer either instantiate or modify the HistoricalCCLInfo MOI with the information related with CCL being deleted.
3. Consumer may decides to initiate a CCL. Before that it would like to understand the historical CCL information.
4. It send getMOIAttributes for HistoricalCCLInfo MOI to read the information captured.
5. Producer send a response
6. Consumer develops the learning based on the historical CCL information received based on the HistoricalCCLInfo MOI attributes.
7. Based on the learning, the consumer send a createMOI request to create a new CCL. It enables the newly created CCL to move from a reactive mode to a proactive mode, where it anticipates and prevents problems based on historical trends and patterns. This proactive approach enhances network optimization, issue prevention and improves the overall efficiency of network operations.
8. Producer send a response.

# 8 CCL Performance Metrics

The performance metrics to evaluate performance of a CCL for its optimal execution are defined in order to enable operators to track the effectiveness of closed loop automation, identify areas for improvement, and make informed adjustments to CCL functionalities.

## 8.1 Total number of occurrences of a requirementbreach:

a) This measurement provides the total number of occurrences when a requirement(e.g., a metrix), as defined in CCL is breached during an observation time period i.e.granularityPeriod.

b) CC.

c) This is measured by counting each incidence when a requirementis breached and incrementing the corresponding counter by one for each such occurrence within an observation time period i.e. granularityPeriod.

d) An integer value.

e) The measurement name has the form TotalRequirementBreach.

f) ClosedControlLoop

g) Valid for packet switched traffic.

h) 5GS.

## 8.2 Time taken by CCL to meet a breached requirement:

a) This measurement provides the time taken by a CCL to meet a breached requirementafter activating it again.

b) DER.

c) This is measured by considering the time stamp when a requirementis breached and subtracting it from the time stamp when that requirementis met after activating the CCL with required changes.

d) Each measurement is an integer representing the mean delay in milliseconds.

e) The measurement name has the form TimeBreachedRequirementRecovery.

f) ClosedControlLoop

g) Valid for packet switched traffic.

h) 5GS.

## 8.3 Total number of conflicts occurred by a CCL:

a) This measurement provides the total number of conflicts that occur between a CCL under consideration and any other CCL during an observation time period i.e. granularityPeriod.

b) CC.

c) This is measured by counting each incidence when conflict occurs between a CCL under consideration and the other CCL and incrementing the corresponding counter by one for each such occurrence within an observation time period i.e. granularityPeriod

d) An integer value.

e) The measurement name has the form TotalCclConflicts_Filter, where filter is either Implicit or Explicit. Implicit represents the action conflict i.e. conflict between two existing CCL and explicit represents the explicit conflict i.e. conflict between an existing CCL and a requested CCL.

f) ClosedControlLoop

g) Valid for packet switched traffic.

h) 5GS.

# 9 Stage 3 definition for Closed Control Loop

## 9.1 RESTful HTTP-based solution set

The RESTful HTTP-based solution set for generic provisioning management service is defined in clause 12.1.1 in 3GPP TS 28.532 [3]. Corresponding className is ClosedControlLoop.

Following is the SS to support CCL lifecycle management based on Table 12.1.1.1.1-1 in TS 28.532 [3].

**Table x.1-1: SS to support CCL lifecycle management**

| CCL lifecycle management | IS operation | HTTP Method | Resource URI |
|---|---|---|---|
| Create an CCL | createMOI operation | PUT | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{ClosedControlLoop}={id}<br><br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLReport}={id}<br><br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLScope}={id} |
| Delete an CCL | deleteMOI operation | DELETE | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{ClosedControlLoop}={id}<br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLReport}={id}<br><br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLScope}={id} |
| Modify an CCL | modifyMOIAttributes operation | PUT<br>PATCH | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{ClosedControlLoop}={id}<br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLReport}={id}<br><br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLScope}={id} |
| Query an CCL | getMOIAttributes operation | GET | {MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{in ClosedControlLoop}={id}<br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLReport}={id}<br><br>{MnSRoot}/ProvMnS/{MnSVersion}/{URI-LDN-first-part}/{CCLScope}={id} |

## 9.2 OpenAPI specification

### 9.2.1 OpenAPI document for provisioning MnS

The OpenAPI/YAML definitions for provisioning MnS are specified in 3GPP Forge, refer to clause 4.3 (OpenAPI Definitions) of TS 28.623 [17] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag_Rel18_SA104/".

Directory: OpenAPI

File: TS28532_ProvMnS.yaml

### 9.2.2 OpenAPI document for CCL NRM

The OpenAPI/YAML definitions for CCL NRM are specified in 3GPP Forge , refer to clause 4.3 (OpenAPI Definitions) of TS 28.623 [17] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag_Rel18_SA104/".

Directory: OpenAPI

File: CCLNrm.yaml

# Annex A (informative): UML code for model diagrams

## A.1 UML code for CCL management model diagrams

This annex contains the PlantUML source code for the NRM diagrams defined in clause 6.2 of the present document.

### A.1.1 CCL NRM fragment (Figure 6.2.1-1)

```
@startuml
skinparam ClassStereotypeFontStyle normal
skinparam ClassBackgroundColor White
skinparam shadowing false
skinparam monochrome true
hide members
hide circle

class ManagedEntity <<ProxyClass>>
class ClosedControlLoop <<InformationObjectClass>>
class CCLPurpose << ProxyClass >>
class CCLScope << InformationObjectClass >>
class CCLReport <<InformationObjectClass>>
class CCLComponent<<InformationObjectClass>>

ManagedEntity "1" *-- "*" ClosedControlLoop: <<names>>

ClosedControlLoop "1" <--> "*" CCLPurpose
ClosedControlLoop "1" *-- "*" CCLScope: <<names>>
ClosedControlLoop "1" *-- "*" CCLReport: <<names>>

ManagedEntity "1" *-- "*" CCLComponent: <<names>>
ClosedControlLoop "1" -r-> "*" CCLComponent


note left of ManagedEntity
   Represents the following IOCs:
     SubNetwork or
     ManagedElement
  end note
note top of CCLPurpose
  Can be any of these CCL purposes:
    NetworkProblemRecovery
    FaultManagement
    ...
end note
@enduml
```

**Source code for Figure 6.2.1-1 CCL NRM fragment**

### A.1.2 NRM fragment for Coordination entity (Figure 6.2.1-2)

```
@startuml
skinparam ClassStereotypeFontStyle normal
skinparam ClassBackgroundColor White
skinparam shadowing false
skinparam monochrome true
hide members
hide circle

class ManagedEntity <<ProxyClass>>
class ConflictManagementAndCoordinationEntity <<InformationObjectClass>>
class CoordinationCapability <<dataType>>
class ClosedControlLoop <<InformationObjectClass>>

ManagedEntity "1" *-- "1" ConflictManagementAndCoordinationEntity: <<names>>
ConflictManagementAndCoordinationEntity "1" -r- "*" CoordinationCapability
ClosedControlLoop "*" -r- "*" ConflictManagementAndCoordinationEntity
```

```
note left of ManagedEntity
   Represents the following IOCs:
     Subnetwork or
     ManagedElement
  end note

note top of CoordinationCapability
   Represents the following capabilities:      ScopeCoordinationCoordination
      TriggerCoordination
      ActionExecutionCoordination
      DirectActionsCoordination

end note

@enduml
```

**Source code for Figure 6.2.1-2 NRM fragment for Conflict management and Coordination entity**

## A.1.3 NRM fragment for CCLTrigger (Figure 6.2.1-3)

```
@startuml
skinparam ClassStereotypeFontStyle normal
skinparam ClassBackgroundColor White
skinparam shadowing false
skinparam monochrome true
hide members
hide circle
class ManagedEntity <<ProxyClass>>
class CCLTrigger<<InformationObjectClass>>
ManagedEntity "1" *-- "*" CCLTrigger: <<names>>
note left of ManagedEntity
   Represents the following IOCs:
     SubNetwork or
     ManagedElement
  end note
@enduml
```

**Source code for Figure 6.2.1-3 NRM fragment for CCLTrigger**

## A.1.4 NRM fragment for Historical CCL (Figure 6.2.1-4)

```
@startuml
skinparam ClassStereotypeFontStyle normal
skinparam ClassBackgroundColor White
skinparam shadowing false
skinparam monochrome true
hide members
hide circle
class ManagedEntity <<ProxyClass>>
class HistoricalCCLInfo<<InformationObjectClass>>
ManagedEntity "1" *-- "1" HistoricalCCLInfo: <<names>>
note left of ManagedEntity
   Represents the following IOCs:
     SubNetwork or
     ManagedElement
  end note
@enduml
```

**Source code for Figure 6.2.1-4 NRM fragment for CCLTrigger**

## A.2 CCL inheritance relationships (Figure 6.2.2-1)

```
@startuml
skinparam ClassStereotypeFontStyle normal
skinparam ClassBackgroundColor White
skinparam shadowing false
skinparam monochrome true
hide members
hide circle
```

```
class Top << InformationObjectClass >>
class ClosedControlLoop <<InformationObjectClass>>
class CCLReport <<InformationObjectClass>>
class CCLScope <<InformationObjectClass>>
class ConflictManagementAndCoordinationEntity <<InformationObjectClass>>
class CCLComponent<<InformationObjectClass>>
class CCLTrigger<<InformationObjectClass>>
class HistoricalCCLInfo<<InformationObjectClass>>

Top <|-- ClosedControlLoop
Top <|-- ConflictManagementAndCoordinationEntity
Top <|-- CCLScope
Top <|-- CCLReport
Top <|-- CCLComponent
Top <|-- CCLTrigger
Top <|-- HistoricalCCLInfo
ClosedControlLoop -[hidden]-> CCLScope
ClosedControlLoop -[hidden]-> CCLTrigger
ClosedControlLoop -[hidden]-> HistoricalCCLInfo
ClosedControlLoop -[hidden]-> CCLComponent
@enduml
```

**Source code for Figure 6.2.2-1 CCL inheritance relationships**

# Annex B (informative): UML code for procedure diagrams

## B.1 UML code for CCL coordination procedure diagrams

This annex contains the PlantUML source code for the procedure diagrams in clause 7 of the present document.

## B.2 Procedure for conditional instantiation of CCLs (Figure 7.1-1)

```
@startuml Procedure for conditional composition of CCLs
skinparam Shadowing false
autonumber
skinparam monochrome true
participant "CCL MnS consumer" as CMC
participant "CCL MnS producer" as CMP
CMC -> CMP: create CCL instantiation conditions
CMP -> CMC: Monitor conditions defined
CMP -> CMP: If conditions in TriggerConditionDescriptor\n evaluate to TRUE instantiate CCL
CMP -> CMC: Notify  conditions.
@enduml
```

**PlantUML source code for Figure 7.1-1 Procedure for conditional instantiation of CCLs**

## B.2 Procedure for conditional composition of CCLs (Figure 7.2-1)

```
@startuml Procedure for conditional composition of CCLs
skinparam Shadowing false
autonumber
skinparam monochrome true

participant "CCL Control MnS consumer" as MNSCS
participant "CCL Control MnS producer" as MNSPD
participant "Management functions" as MNFs

MNSCS -> MNSPD: create CCL composition desription
MNSCS -> MNSPD: create CCL composition conditions\n as an instance of TriggerConditionDescriptor
MNSPD -> MNSPD: Monitor conditions defined\n in TriggerConditionDescriptor
MNSPD -> MNSPD: If conditions in TriggerConditionDescriptor\n evaluate to TRUE, trigger execution\n
of CCL composition operations
MNSPD -> MNSCS: Notify  conditions\n and triggering of composition.
Note over MNSPD, MNFs: execute CCL composition operations
MNSPD -> MNSCS: If composition is complete,\n Notify MnS consumer of composed CCL

@enduml
```

**PlantUML source code for Figure 7.2-1 Procedure for conditional composition of CCLs**

## B.3 CCL decision escalation procedure (Figure 7.4-1)

```
B.2.1   CCL decision escalation procedure (Figure 7.6-1)
@startuml avoidance of potential action-execution-time conflicts - Information on detected conflict
skinparam Shadowing false
autonumber
skinparam monochrome true

participant "CCL MnS Consumer" as MNSCS
participant "CCL (Escalator CCL)" as ESCCL
participant "Escalation Recipient\n (e.g. another CCL or CCL Coordination Entity)" as ESCRP
```

```
Note over MNSCS, ESCRP: Compose, configure and instantiate the Escalator CCL and Escalation 
Recipient.

MNSCS -> ESCCL: configure or reconfigure Escalator CCL\n with when and where to escalate 
Note over MNSCS,ESCCL: Trigger CCL execution
ESCCL -> ESCCL: Derive analysis and decision for a scenario
ESCCL -> ESCCL: detect need to escalate the scenario

ESCCL -> ESCRP: Request escalation for the scenario
ESCRP -> ESCRP: Decide whether to accept\n escalated request.

ESCRP -> ESCCL: Notify  acceptance of escalated request.
ESCRP -> ESCRP: Derive analysis and decision\n for an escalated scenario
ESCRP -> ESCCL: Notify  Escalator CCL of\n escalation outcome for the scenario.

@enduml
```

**PlantUML source code for Figure 7.4-1 CCL NRM fragment**

## B.4 CCL-impact assessment and metric conflicts resolution on unknown or unbounded impact-scope (Figure 7.5-1)

```
@startuml CCL-impact assessment and metric conflicts resolution on unknown or unbounded impact-scope
skinparam Shadowing false
autonumber
skinparam monochrome true

participant "Actor-CCL \n (CCL MnS producer & \n Coordination MnS Consumer)" as CL1
collections "other-CCLs \n (CCL MnS producer & \n other functions)" as CL2
participant "CCL Coordination MnS producer \n (scope coordination)" as xCL
participant "Network" as Net

Note over CL1, xCL: Actor-CCL and other-CCLs are composed, instantiated and configured as required.

CL2 -> xCL: Register measurement, control, \n& impact scopes of interest

CL1 -> Net: execute derived action plan A

CL1 -> xCL: notify executed action plan A [incl. impact time of action, time for feedback
xCL -> CL2: notify execution of action plan A from \nCCL1 [indicate feedback time]

CL2 -> CL2: evaluate impacts of \naction A to own metrics
CL2 -> xCL: notify impact of action plan A on other CCLs

xCL -> xCL: compute aggregate AQI\n as aggregate impact on\n all affected entities
xCL -> CL1: notify aggregate impact of action plan A on other CCLs

Alt
  CL1 -> CL1: modify own decisions, e.g.,  the control scope
end

Alt
  CL1 -> Net: undo/revise executed action plan A
end
@enduml
```

**PlantUML source code for Figure 7.5-1 CCL NRM fragment**

## B.5 CCL Scope conflicts avoidance, detection and resolution (Figure 7.6-1)

```
@startuml CCL Scope conflicts avoidance, detection and resolution
skinparam Shadowing false
autonumber
skinparam monochrome true
!pragma teoz true

participant "Actor-CCL \n (CCL MnS producer & \n Coordination MnS Consumer)" as CL1
collections "other-CCLs \n (CCL MnS producer & \n other functions)" as CL2
```

```
participant "CCL Coordination \nMnS producer \n (scope coordination)" as xCL

Note over CL1, xCL: Actor-CCL, other-CCLs and CoordinationEntity’s capability for scope
coordination \nare instantiated and configured as required.

CL1 -> xCL: notified creation of new CCL

Alt CCL does not have a configured scope
  xCL -> xCL: compute scope to apply to \n new CCL, e.g., control scope
  xCL -> CL1: modify new scope
else CCL is alreaady configured with a scope

CL1 -> xCL:
& CL2 -> xCL: Register scopes of interest

Alt
CL1 -> CL1: monitor for changes in their scope
CL1 -> xCL: notify observed changes in scope
End

xCL -> xCL: evaluate potential conflicts \namong the registered scopes
xCL -> CL1:
& xCL -> CL2: notify potential conflicts \namong the registered scopes

xCL -> xCL: monitor for negative outcomes \nrelated to potential conflicts
& CL1 -> CL1: monitor for negative outcomes \nrelated to potential conflicts
alt
CL1 -> xCL: notify confirmed actual conflicts (from negative outcomes)

end
xCL -> CL1:
& xCL -> CL2: notify confirmed actual conflicts \n(from negative outcomes)

xCL -> xCL: compute new scopes for CCLs
xCL -> CL1:
& xCL -> CL2: modify new scope(s)

@enduml
```

**PlantUML source code for Figure 7.6-1 CCL-Scope conflicts avoidance, detection and resolution**

## B.6 CCL Trigger-time conflicts avoidance, detection and resolution (Figure 7.7-1)

```
@startuml CCL Trigger-time conflicts avoidance, detection and resolution
skinparam Shadowing false
autonumber
skinparam monochrome true
!pragma teoz true

participant "Actor-CCL \n (CCL MnS producer & \n Coordination MnS Consumer)" as CL1
collections "other-CCLs \n (CCL MnS producer & \n other functions)" as CL2
participant "CCL Coordination \nMnS producer \n (CCL trigger coordination)" as xCL

Note over CL1, xCL: CCLs and CoordinationEntity’s capability for CCL trigger coordination
\nare instantiated.

CL1 -> xCL: Register precedent functionality

xCL -> xCL: evaluate and align \nhierachies
xCL -> CL1: reconfigure precedents \n & hierachies

xCL -> xCL: monitor for problem \nanalytics reports
xCL -> xCL: Determine the right \nCCLs to trigger

xCL -> CL1: trigger CCL, indicate \nprecedent CCL’s action

CL1 -> CL1: derive and execute actions
```

```
alt
CL1 -> xCL: notify completion of execution and executed actions

@enduml
```

**PlantUML source code for Figure 7.7-1 CCL- Trigger-time conflicts avoidance, detection and resolution**

## B.7 CCL concurrent actions conflicts avoidance, detection and resolution (Figure 7.8-1)

```
@startuml CCL CCL actions conflicts, detection and resolution
skinparam Shadowing false
autonumber
skinparam monochrome true
!pragma teoz true

participant "Actor-CCL \n (CCL MnS producer & \n Coordination MnS Consumer)" as CL1
collections "other-CCLs \n (CCL MnS producer & \n other functions)" as CL2
participant "CCL Coordination \nMnS producer \n (CCL CCL actions conflicts coordination)"
as xCL

Note over CL1, xCL: CCLs and CoordinationEntity’s capability for CCL trigger coordination
\nare instantiated.

CL1 -> xCL:
& CL2 -> xCL: Register scopes of interest

CL1 -> CL1: Derive desired actions plan
CL2 -> CL2: Derive desired actions plan

CL1 -> xCL: Register desired actions plan
CL2 -> xCL: Register desired actions plan

xCL -> xCL: evaluate actions plans \nto identify conflict actions
xCL -> xCL: identify acceptable and \nunacceptable actions .
xCL -> CL1:
& xCL -> CL2: notify accepted/unaccepted actions; \nreconfigure CCLs (if needed)

@enduml
```

**PlantUML source code for Figure 7.8-1 CCL coordination to avoid, detect and resolve CCL-concurrent actions conflicts**

## B.8 CCL non-concurrent actions conflicts CCL coordination to avoid, detect and resolve (Figure 7.9-1)

```
@startuml CCL CCL actions conflicts, detection and resolution
skinparam Shadowing false
autonumber
skinparam monochrome true
!pragma teoz true

participant "Actor-CCL \n (CCL MnS producer & \n Coordination MnS Consumer)" as CL1
collections "other-CCLs \n (CCL MnS producer & \n other functions)" as CL2
participant "CCL Coordination \nMnS producer \n (CCL actions conflicts coordination)" as
xCL
```

```
Note over CL1, xCL: Actor-CCL, other-CCLs and CoordinationEntity’s capability for CCL
actions coordination \nare instantiated and configured as needed

CL1 -> xCL:
& CL2 -> xCL: Register scopes of interest

Alt other-CCLs have executed
CL2 -> xCL: Register executedAction and \ntolerance on parameters & metrics.
End

CL1 -> CL1: Derive desired actions plan
CL1 -> xCL: Register desired actions plan

xCL -> xCL: evaluate actions plans \nto identify conflict actions
xCL -> CL1: notify acceptable and unacceptable actions

CL1 -> CL1: if action accepted, execute onto network
CL1 -> xCL: Register executedAction.
xCL -> CL1: provide conflict monitoring context and tolerance limits

CL1 -> CL1: Monitor context to \ndetect tolerance breach
CL1 -> xCL: if violations detected, report conflict
xCL -> CL1:
& xCL -> CL2: notify conflict, \nrequest usefulness & \ninterest levels

CL1 -> xCL:
& CL2 -> xCL: provide usefulness & interest \nlevels for the conflict parameter(s)
xCL -> xCL: Derive compromise \nvalues
xCL -> CL1:
& xCL -> CL2: Notify compromise values.
CL1 -> CL1: Execute revised action onto network

@enduml
```

**PlantUML source code for Figure 7.9-1 CCL coordination to avoid, detect and resolve CCL non-concurrent actions conflicts**

## B.9 CCL metric-value conflicts avoidance and detection (Figure 7.F-1)

```
@startuml CCL CCL actions conflicts, detection and resolution
skinparam Shadowing false
autonumber
skinparam monochrome true
!pragma teoz true

participant "Actor-CCL \n (CCL MnS producer & \n Coordination MnS Consumer)" as CL1
collections "other-CCLs \n (CCL MnS producer & \n other functions)" as CL2
participant "CCL Coordination \nMnS producer \n (CCL metric-value conflicts
coordination)" as xCL

Note over CL1, xCL: Actor-CCL, other-CCLs and CoordinationEntity’s capability for CCL
actions coordination \nare instantiated and configured as needed

CL1 -> xCL:
& CL2 -> xCL: Register scopes (incl. metrics)of interest

xCL -> xCL: Evaluate desired metrics \n& outcomes for conflict
xCL -> xCL: If potential conflict, revise \n metrics & planned outcomes
xCL -> CL1:
& xCL -> CL2: send new/ revised \nmetrics and outcomes.

Alt other-CCLs have executed
CL2 -> xCL: Register executedAction and \ntolerance on parameters & metrics.
End

CL1 -> CL1: Derive desired actions plan
CL1 -> xCL: Register desired actions plan
```

```
xCL -> xCL: evaluate performance \nimprovement & reliability/\nconfidence of actions
plans
xCL -> CL1: notify action accepted or not (& failed criteria if check has failed)
alt
xCL -> CL1: Notify if CCL is trusted
end

CL1 -> CL1: if action accepted, execute onto network
CL1 -> xCL: Register executedAction,  planned impact and desired outcomes
xCL -> xCL: compare impact scopes \nto detect metric correlated \noscillations
CL1 -> CL1: evaluate own desired metrics for ping-pong/ flipflops

xCL -> CL1: If correlated oscillations, inform actor CCLs of the correlated oscillations
(i.e. potential conflict)
CL1 -> xCL: If flipflop, inform of flipflop \n(i.e, a potential conflict)
xCL -> xCL: derives new CCL config \n(e.g., desired metrics priorities)
xCL -> CL1: notifies new config to CCL inclufing new metrics or outcomes.

@enduml
```

**PlantUML source code for Figure 7.9-1 CCL coordination to avoid and detect CCL metric-value conflicts**

## B.10 CCL creation based on Historical CCL data (Figure 7.11-1)

```
@startuml CCL creation based on Historical CCL data
skinparam Shadowing false
autonumber
skinparam monochrome true

participant "CCL MnS Consumer" as CL1
participant "CCL MnS producer" as CL2


Note over CL2: CCL Configuration

Loop
  CL1 -> CL2: DeleteMOI (CCL) Request
  CL2 -> CL1: DeleteMOI (CCL) Respone
  Note over CL2: Instantiate HistoricalCCLInfo
end

CL1 -> CL1: Consumer may decides to initiate a CCL
CL1 -> CL2: getMOIAttributes (HistoricalCCLInfo) request
CL2 -> CL1: getMOIAttributes (HistoricalCCLInfo) response
CL1 -> CL1: Learning by the consumer

Opt
  CL1 -> CL2: CreateMOI(CCL) Request
  CL2 -> CL1: CreateMOI(CCL) Respone
end
@enduml
```

**PlantUML source code for Figure 7.11-1 CCL creation based on Historical CCL data**

# Annex D: Change history

| **Change history** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2025-02 | - | n/a | - | - | - | Initial skeleton | 0.0.0 |
| 2025-02 | SA5#159 | S5-251107<br>S5-251137<br>S5-251123<br>S5-251124<br>S5-250944<br><br>S5-250945<br><br>S5-250943 | pCR | - | - | 28.567-001<br>pCR TS28.567 TS clause structure<br>pCR TS28.567 Concepts and overview<br>pCR TS28.567 Dynamic composition of CCLs<br>Rel-19 pCR 28.567 CCL creation based on historical data UC and Requirements<br>Rel-19 pCR TS 28.567 Add use case and requirements for Performance monitoring of Closed control loop<br>Rel19 pCR TS28.567 Add use case and requirements for CCL for Fault Mgmt | 0.1.0 |
| 2025-04 | SA5#160 | S5-251904<br>S5-251905<br>S5-251906<br>S5-251907<br>S5-251909<br>S5-251911<br>S5-251912<br><br>S5-252030<br>S5-252033<br><br>S5-252034 | pCR | - | - | Rel-19 pCR 28.567 Triggered CCL UC and Requirements<br>CCL Trigger conflicts<br>CCL Goal-targets and actions conflicts<br>CCL Indirect target conflicts<br>Informative Annex on CCL conflict handling approach<br>CCL decision escalation<br>pCR TS 28.567 Add use case and requirements for network performance problem recovery<br>CCL General NRM<br>Rel-19 pCR TS 28.567 Procedure and metrics for CCL performance monitoring<br>Coordinating CCLs with other functions | 0.2.0 |
| 2025-05 | SA5#161 | S5-252491<br><br>S5-252846<br>S5-252848<br>S5-252849<br>S5-252850<br>S5-252851<br>S5-252852<br><br>S5-252853<br>S5-252854<br>S5-252855<br>S5-252856<br>S5-252857<br>S5-253007<br>S5-253008<br>S5-253009<br>S5-253010<br><br>S5-253020 | pCR | | | pCR TS 28.567 Correct information models for CCL management in clause 6.2.1<br>CCL coordination NRM<br>CCL concurrent-actions conflicts<br>CCL scope conflicts<br>Solution for Dynamic CCLs<br>Rel-19 pCR 28.567 solution for conflict management<br>Pseudo-CR on Rel-19 TS 28.567 Add Stage-2 Solutions for Fault Management CCL<br>Rel-19 pCR TS 28.567 Metrics for CCL performance monitoring<br>CCL-impact assessment and resolution<br>Procedure for CL decision escalation<br>CCL-impact assessment, feedback and resolution<br>Definitions of CCL conflicts<br>Control of CCLs<br>Example realizations of CCLs<br>CCL DraftTS Rapporteur corrections<br>Pseudo-CR on Rel-19 TS 28.567 Add Definitions of terms and abbreviations for closed control loop<br>Rel-19 pCR 28.567 solution for triggered CCL | 0.3.0 |
| 2025-08 | SA5#162 | S5-253486<br><br>S5-253531<br><br>S5-253599<br>S5-253602<br>S5-253604<br>S5-253608<br>S5-253858<br>S5-253859<br>S5-253860<br>S5-253861<br><br>S5-253862<br>S5-253863<br>S5-253864<br>S5-253865<br>S5-253866<br>S5-253867 | | | | Pseudo-CR on Rel-19 TS 28.567 Corrections on Fault Management CCL Use-case<br>Rel-19 pCR TS 28.567 Update the attribute definition in clause 6.4.1<br>Rel-19 pCR 28.567 Model imports<br>Rel-19 pCR 28.567 Adding missing descriptions<br>Rel-19 pCR 28.567 Fixing References<br>Rel-19 pCR 28.567 Fixing notifications<br>Rapporteur cleanup and corrections<br>Restructure clauses<br>Rel-19 pCR 28.567 CCL Stage 3<br>Rel-19 pCR TS 28.567 Remove the descriptions related to target and goal<br>CCL scope conflict coordination NRMs<br>CCL trigger conflict coordination NRMs2<br>CCL Action conflict coordination NRMs<br>CCL metric-value conflict coordination NRMs<br>Rel-19 pCR 28.567 Historical CCL<br>Rel-19 pCR 28.567 defining CCLTrigger | 0.4.0 |
| 2025-09 | SA#109 | SP-251036 | | | | Presented for information and approval | 1.0.0 |
| 2025-09 | SA#109 | | | | | Upgrade to change control version | 19.0.0 |

# History

| Document history | | |
|---|---|---|
| V19.0.0 | October 2025 | Publication |
| | | |
| | | |
| | | |
| | | |