**3GPP TSG-SA5 Meeting #162 *S5-253334***

Gothenburg, Sweden, 25 – 29 August 2025

**Source: Nokia**

**Title: Pseudo-CR on CCL Scope** **conflicts Coordination NRM**

**Document for: Approval**

**Agenda item: 6.19.4.1**

**Spec: 3GPP TS28.567**

**Version: 0.3.0**

**Work Item: Closed Control Loop Management**

**Comments**

This pCR is to add NRM and procedures for CCL Scope conflicts Coordinationas was agreed in the CCL study in TR28. 867

**Proposed Changes**

\* \* \* First Change \* \* \* \*

### 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.



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.

\* \* \* Second Change \* \* \* \*

# 6 Model

## 6.1 Imported and associated information entities

TBD

### 6.1.1 Imported information entities and local labels

TBD

### 6.1.2 Associated information entities and local labels

TBD

## 6.2 Class diagram

### 6.2.1 Relationships

A diagram of a computer program

AI-generated content may be incorrect.

Figure 6.2.1-1: Relations for common information models for CCLmanagement

Editor’s Note: The handling of Goal, targets or objectives for the general closed control loops is FFS

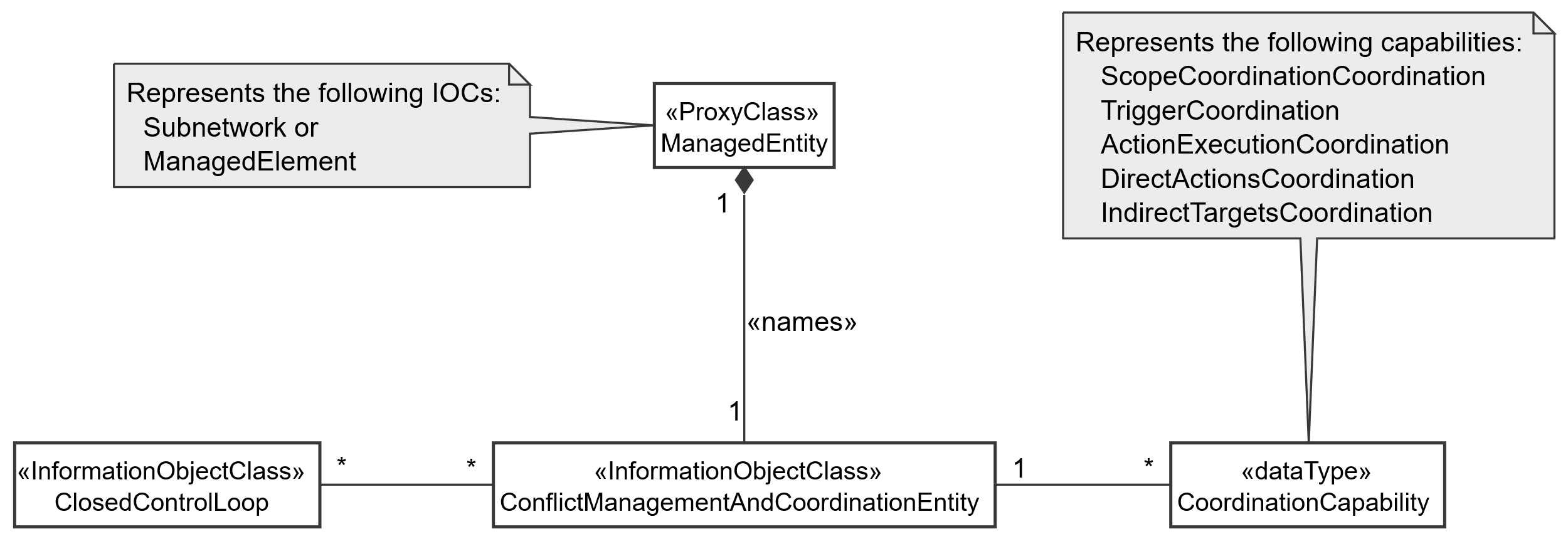


Figure 6.2.1-2: NRM fragment for conflict management and Coordination entity

### 6.2.2 Inheritance

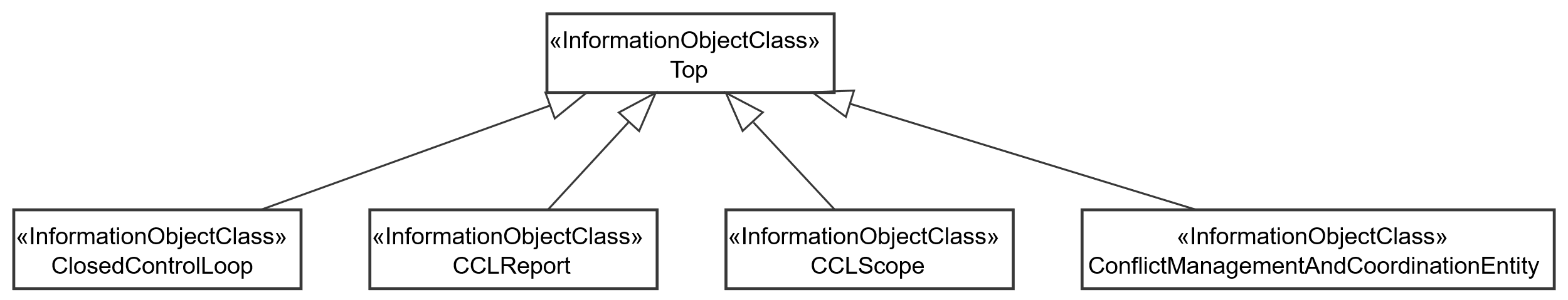


Figure 6.2.2-1: Inheritance Hierarchy for Closed Control Loops and for conflict management and Coordination entity

## 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 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 cCLComponents indicates the list of components which are combined to create a CCL.

The attribute 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 CCLControlLoop IOC includes attributes inherited from Top IOC (defined 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 |
| **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 6.1 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 6.1 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 6.1 are valid for this IOC, without exceptions.

### 6.3.4 ConflictManagementAndCoordinationEntity

#### 6.3.4.1 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.

#### 6.3.4.2 Attributes

Table 6.3.4.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | Support Qualifier | isReadable | isWritable | isInvariant | isNotifyable |
| cCLScopecoordinationCapability | M | T | T | F | T |
| coordinatedCCLsScopes | M | T | T | F | T |
| cCLActionConflictsHandling | M | T | T | F | T |
| **Attribute related to role** |  |  |  |  |  |
|  |  |  |  |  |  |

#### 6.3.4.3 Attribute constraints

None

#### 6.3.4.4 Notifications

The common notifications defined in clauses 6.1 are valid for this IOC, without exceptions.

### 6.3.5 FaultManagement <<IOC>>

#### 6.3.5.1 Definition

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

#### 6.3.5.2 Attributes

**Table 6.3.5.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

#### 6.3.5.3 Attribute constraints

**Table 6.3.5.3-1**

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

#### 6.3.5.4 Notifications

None.

### 6.3.6 CCLComponentInfo <<dataType>>

#### 6.3.6.1 Definition

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

#### 6.3.6.2 Attributes

Table 6.3.6.2-1

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

#### 6.3.6.3 Attribute constraints

None.

#### 6.3.6.4 Notifications

The common notifications defined in clauses 6.1 are valid for this IOC, without exceptions.

### 6.3.7 CCLComponent <<dataType>>

#### 6.3.7.1 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.

#### 6.3.7.2 Attributes

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

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

#### 6.3.7.3 Attribute constraints

None

#### 6.3.7.4 Notifications

The common notifications defined in subclause 4.1.2.5 are valid for this IOC, without exceptions or additions.

### 6.3.8 FaultManagementCCLReport <<dataType>>

#### 6.3.8.1 Definition

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

#### 6.3.8.2 Attributes

**Table 6.3.8.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **S** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| GeneratedAlarmResultList | M | T | F | T | T |
| FaultManagementCCLReportTime | M | T | F | T | T |

#### 6.3.8.3 Attribute constraints

None.

#### 6.3.8.4 Notifications

None.

### 6.3.9 GeneratedAlarmResult <<dataType>>

#### 6.3.9.1 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 6.3.9.2-1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

#### 6.3.9.3 Attribute constraints

None.

#### 6.3.9.4 Notifications

None.

### 6.3.10 CCLPurpose <<dataType>>

#### 6.3.10.1 Definition

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

#### 6.3.10.2 Attributes

Table 6.3.10.2-1

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

#### 6.3.10.3 Attribute constraints

None.

#### 6.3.10.4 Notifications

The common notifications defined in clauses 6.1 are valid for this IOC, without exceptions..

### 6.3.11 CCLScopeCoordinationCapability <<dataType>>

#### 6.3.11.1 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.

#### 6.3.11.2 Attributes

Table 6.3.11.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

#### 6.3.11.3 Attribute constraints

None.

#### 6.3.11.4 Notifications

The common notifications defined in clauses 6.1 are valid for this IOC, without exceptions.

### 6.3.12 CCLActionConflictsHandling <<datatype>>

#### 6.3.12.1 Definition

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

#### 6.3.12.2 Attributes

Table 6.3.12.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

#### 6.3.12.3 Attribute constraints

None

#### 6.3.12.4 Notifications

The common notifications defined in subclause 4.1.2.5 are valid for this IOC, without exceptions or additions.

### 6.3.A ScopeConflict <<datatype>>

#### 6.3.A.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.A.2 Attributes

Table 6.3.A.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.A.3 Attribute constraints

None

#### 6.3.A.4 Notifications

The subclause 6.x of the <<IOC>> using this <<dataType>> as one of its attributes, shall be applicable.

### 6.3.13 ConflictInformation <<datatype>>

#### 6.3.13.1 Definition

This defines the information related with a conflicting CCLs that have been detected.

#### 6.3.13.2 Attributes

Table 6.3.13.2-1

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

#### 6.3.13.3 Attribute constraints

None

#### 6.3.13.4 Notifications

The common notifications defined in subclause 4.1.2.5 are valid for this IOC, without exceptions or additions.

### 6.3.14 ActionConflictResolution <<datatype>>

#### 6.3.14.1 Definition

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

#### 6.3.14.2 Attributes

Table 6.3.14.2-1

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

#### 6.3.14.3 Attribute constraints

None

#### 6.3.14.4 Notifications

The common notifications defined in subclause 4.1.2.5 are valid for this IOC, without exceptions or additions.

## 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.  allowedValues: CCL\_MEASUREMENT\_SCOPE, CCL\_TARGET\_SCOPE, CCL\_CONTROL\_SCOPE, CCL\_IMPACT\_SCOPE  Editor’s Note: The allowed values will be revisited | type: Enum  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  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  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  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.  allowedValues: string | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| coordinationCapability | It indicates a capability of a coordination entity to coordinate CCL conflicts | type: CoordinationCapability  multiplicity: \*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| cCLCoordinationCapabilityID | It indicates an identifier for a specific CCL conflicts coordination capability | type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| closedControlLoopRefList | It indicates a list of DN for ClosedControlLoop Instances.  allowedValues: N/A | type: DN  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLScopeCoordinationCapability | It indicates a CCL scope assignment and conflict coordination capability | type: CCLScopeCoordinationCapability  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| coordinatedCCLsScopes | It indicates the scopes of the CCL that are coordinated by the coordinationEntity  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 >  multiplicity: 2 ..\*  isOrdered: False  isUnique: True  defaultValue: None  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).  AllowedValues; Enabled/Disabled  allowedValues: "ENABLED", "DISABLED".  The meaning of these values is as defined in 3GPP TS 28.625 [14] and ITU-T X.731 [15]. | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: Disabled  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.  AllowedValues; Locked/Unlocked  allowedValues: "LOCKED", "UNLOCKED".  The meaning of these values is as defined in 3GPP TS 28.625 [14] and ITU-T X.731 [15]. | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: Locked  isNullable: False |
| cCLComponentsInfo | It indicates information on the constituent components of a CCL.  allowedValues: N/A | type: CCLComponentInfo  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLComponentId | It indicates the identifier of a CCL component. It is the DN of a object instantiated to act as a component of the CCL | type: DN  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLSteps | It indicates the CCL steps or functionality that is accomplished by a CCL component.  allowedValues: DATA\_COLLECTION, ANALYSIS, DECISION, EXECUTION | type: Enum  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| FaultManagementAlarmIdList | It describes the list of IDs of alarms to be managed by Fault Management CCL.  allowedValues: A list of alarmIds as specified in TS 28.111 [4], clause 7.4.1 | type: List  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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.  allowedValues: timeWindow as defined in 3GPP TS 28.622 [5], clause 4.4.1 | type: TimeWindow  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: True |
| FaultManagementBackUpObjectRequirement | It describes whether to back-up the alarmed object is required by the consumer before fault management.  allowedValues: True, False | type: Booelan  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| FaultManagementIsolateObjectRequirement | It describes whether to isolate the alarmed object from interaction with other objects is required by the consumer before fault management.  allowedValues: True, False | type: Booelan  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| clearUserId | It carries the identity of the Fault Management CCL who is the consumer that invokes the clearAlarms operation.  allowedValues: clearUserId as defined in 3GPP TS 28.111 [4], clause 7.4.1 | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| FaultManagementCCLReport | It describes the Fault Management CCL report.  allowedValues: Not Applicable | type: FaultManagementCCLReport  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| GeneratedAlarmResultList | It describes the list of generated alarm results  allowedValues: A list of GeneratedAlarmResult | type: List  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| GeneratedAlarmResult | It describes the result for each alarmId listed in FaultManagemetAlarmIdList  allowedValues: Not Applicable | type: GeneratedAlarmResult  multiplicity: 1..\*  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| FaultManagementCCLReportTime | It describes the time when the FaultManagementCCLReport is created.  allowedValues: DateTime as specified in TS 28.622 [5]. | type: DateTime  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| alarmId | It identifies an AlarmRecord as specified in TS 28.111 [4]  allowedValues: A string as specified in TS 28.111 [4] | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| alarmClearedStatus | It describes whether an alarm is cleared by the Fault Management CCL when the identified root cause is resolved.  allowedValues: True, False | type: Booelan  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| identifiedRootCauseInformation | It describes root cause information identified by the Fault Management CCL.  allowedValues: String | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| enhancedCorrelationInformation | It describes the list of correlated alarm Ids identified by the Fault Management CCL  allowedValues: A list of alarmId | type: List  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| cCLActionConflictsHandling | This defines the handling of CCL action conflict between the two existing CCLs. | Type: cCLActionConflictsHandling  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| conflictInformation | This defines the information related with a conflicting CCL. | Type: ConflictInformation  multiplicity: \*  isOrdered: True  isUnique: False  defaultValue: None  isNullable: False |
| conflictResolution | This defines the information related with conflict resolution. | Type: ConflictResolution  multiplicity: \*  isOrdered: True  isUnique: False  defaultValue: None  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  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| conflictingCCLId | This indicates the CCL identification | Type: Dn  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| conflictingActions | This provides the set of actions that have been taken by the CCL as part of the Execute step. | Type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  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  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| cCLComponentList | It indicates the list of components ating 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 caes where a components fulfile more than 1 role or where the role can be siml y described by the four options.  The cCLComponents are sequenced, i.e., cCLComponents is an ordred 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  multiplicity: 1..\*  isOrdered: True  isUnique: True  defaultValue: None  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.  The specific details, characteristics and behavior of a CCL for a given CCL type are then written into the CCL purpose.  Editor’s Note: Documentation and Allowed values will be revisited | type: String  multiplicity: 1  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLComponentRole | It indicates a role accomplished by CCL component.  AllowedValues: MONITOR; ANALYSIS; DECISION; EXECUTION, OTHER. Is used for example in the caes where a components fulfile more than 1 role or where the role can be siml y described by the four options | type: Enum  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLComponentIdentification | It indicates the entity accomplishing the component.  It may be the the DN of an MOI or the combination of URI and DN that can be used to fulfil that role. | Type: String  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLActionTrigger | This defines the criteria/conditions under which the CCL is allowed to take actions. | Type: CCLTrigger  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| desiredBehavior | This will define the corresponding behavior of the CCL. The behaviors can be represented by an ENUM to include:  - DECISION\_ACTIVATION: The CCL executes the recommendations that it derives on to the network.  - 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.  - DO\_NOTHING: do not do anything. | Type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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>  multiplicity: 1...\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| conflictID | It identifies a conflict event | type: Integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| conflictingCCLs | It identifies the set of CCLs that are conflicting | type: DN  multiplicity: 2  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| conflictScope | It indicates the scope for which two or more CCLs are conflicting. | Type: ScopeDefinition  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| ConflictType | It indicates the type of conflict that has been observed, i.e., either a potential conflict or an actual conflict.  allowedValues: POTENTIAL\_CONFLICT; ACTUAL\_CONFLICT | Type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| coordinatedScopeTypes | It indicates the types of scopes under consideration for coordination by a scope coordination functionality.  allowedValues: CCLMEASUREMENTSCOPE, CCLTARGETSCOPE, CCLCONTROLSCOPE, CCLIMPACTSCOPE, CCLMONITOREDSCOPE | Type: ENUM  multiplicity: 1 ..5  isOrdered: False  isUnique: True  defaultValue: None  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  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  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  multiplicity: 1..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |

\* \* \* Second Change \* \* \* \*

# 7 Procedures

## 7.1 Procedure for conditional trigger/instantiation of CCLs

A diagram of a product

AI-generated content may be incorrect.

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 conditios to be evaluated for instantiation of the CCL. These conditions are created as an instance of TriggerConditionDescriptor defned 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

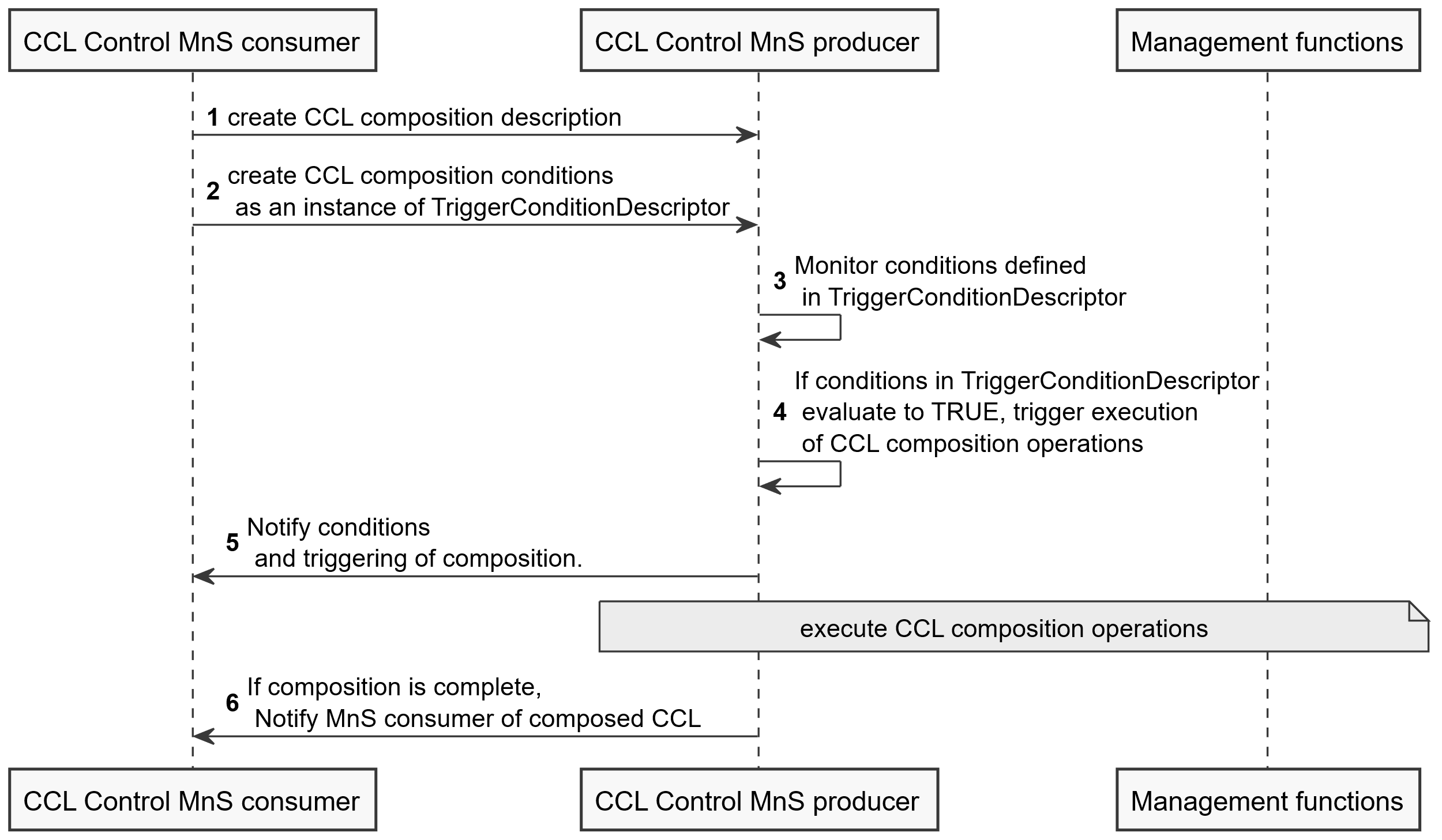


Figure 7.2-1: Procedure and interactions for conditional composition of CCLs

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 desription 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 instantiatng the objects to be used as components of the closd control loop, e.g., a PMJob to be used to collect data

- modifyMOI operations for configuring the instantiated components to enable thenm oprate as a single loop, e.g., to configure te PMJob to delvier data to an analytics sinstance

Step 2: The MnS consumer creates on the MnS producer responsible for instantiating CCLs the set of conditios to be evaluated fot composing the CCL. Thse conditions are created as an instance of TriggerConditionDescriptor defned 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 triggere dteh 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 goals 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 goal breach, time taken by CCL to meet a breached goal, 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



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 achieve the desired goal or 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 for the desired goal, 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.

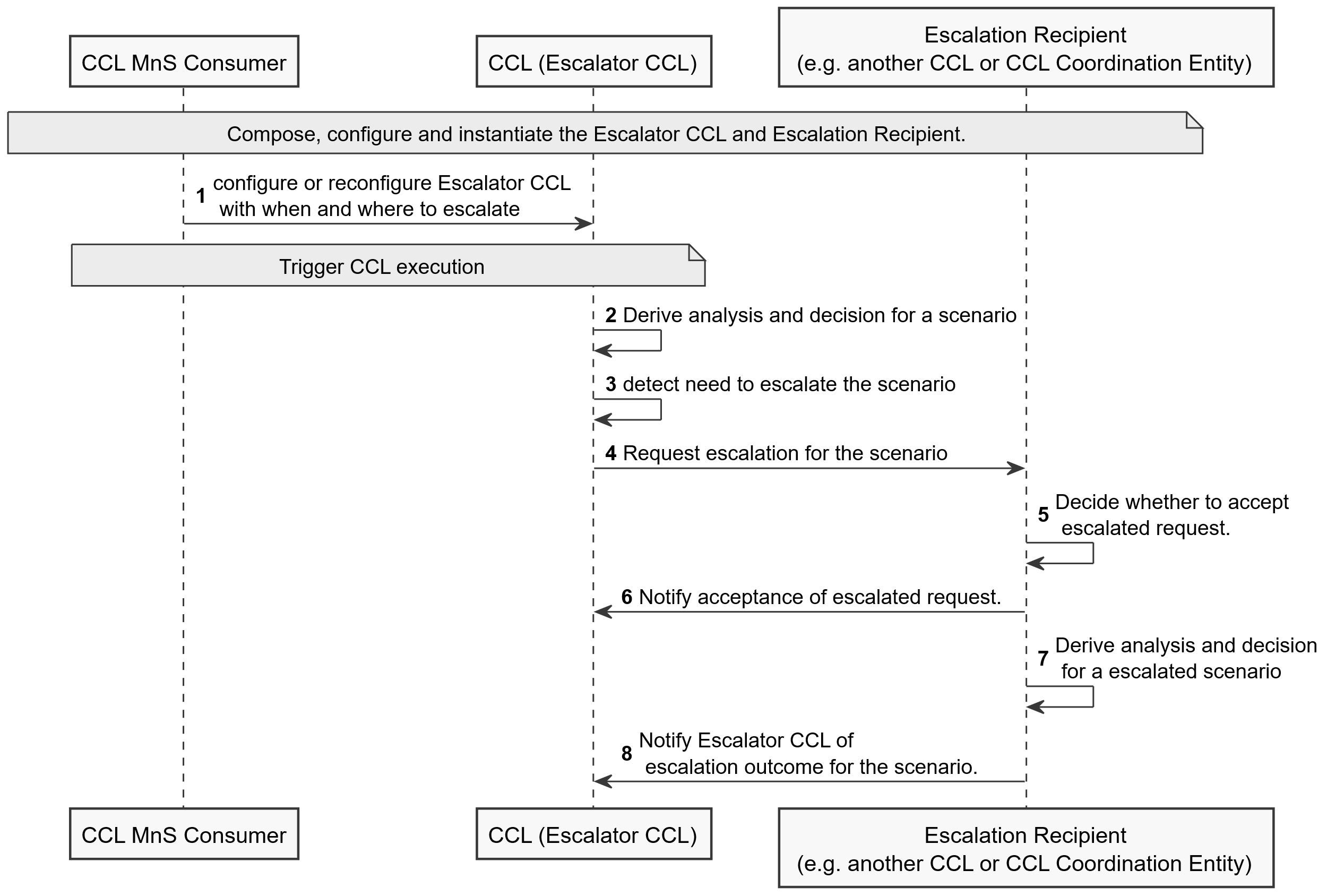


Figure 7.4-1: Procedure and interactions for CCL decision escalation

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 acceptane 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 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.

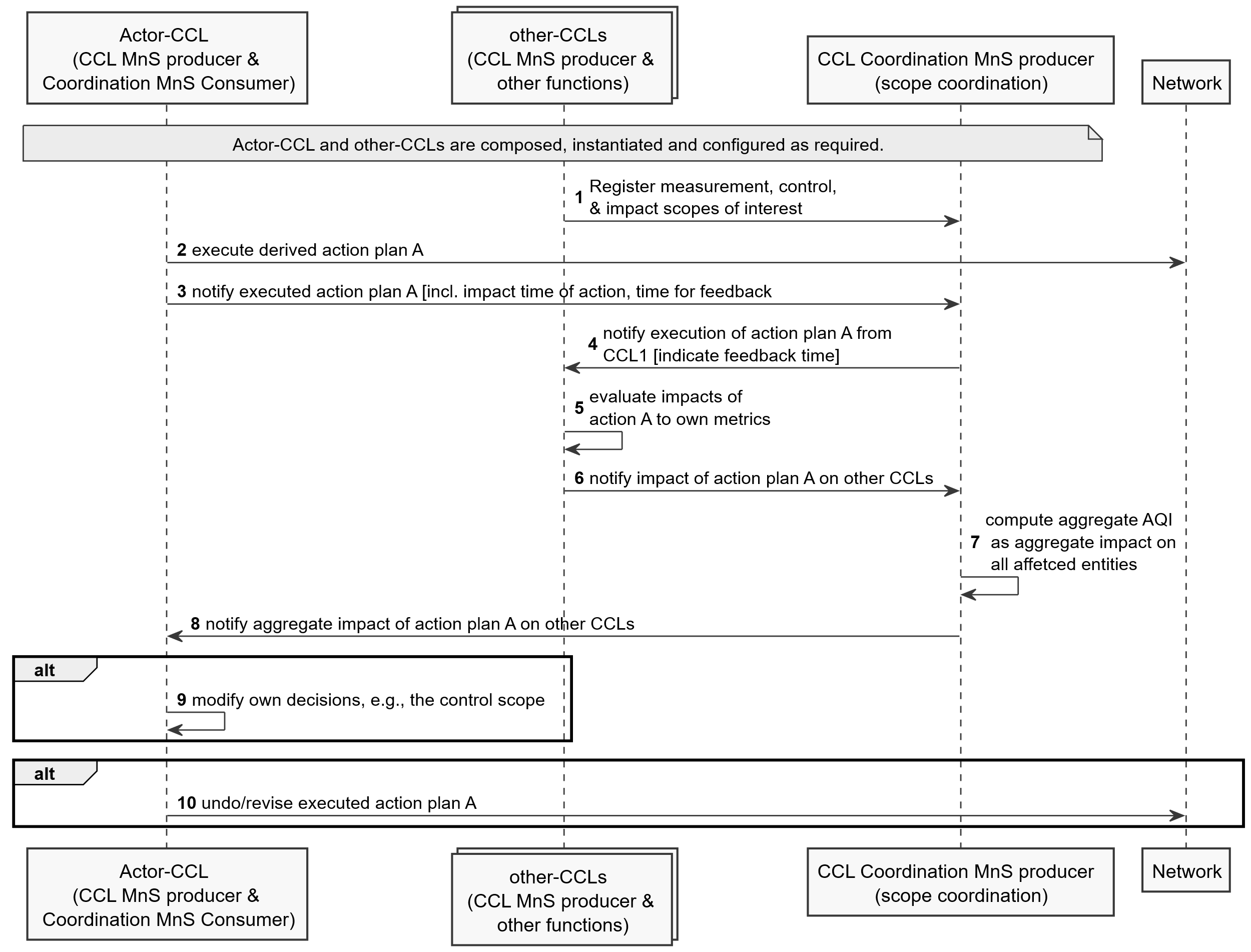


Figure 7.5-1: CCL-impact assessment and actual metric-value conflicts resolution

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.A 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.

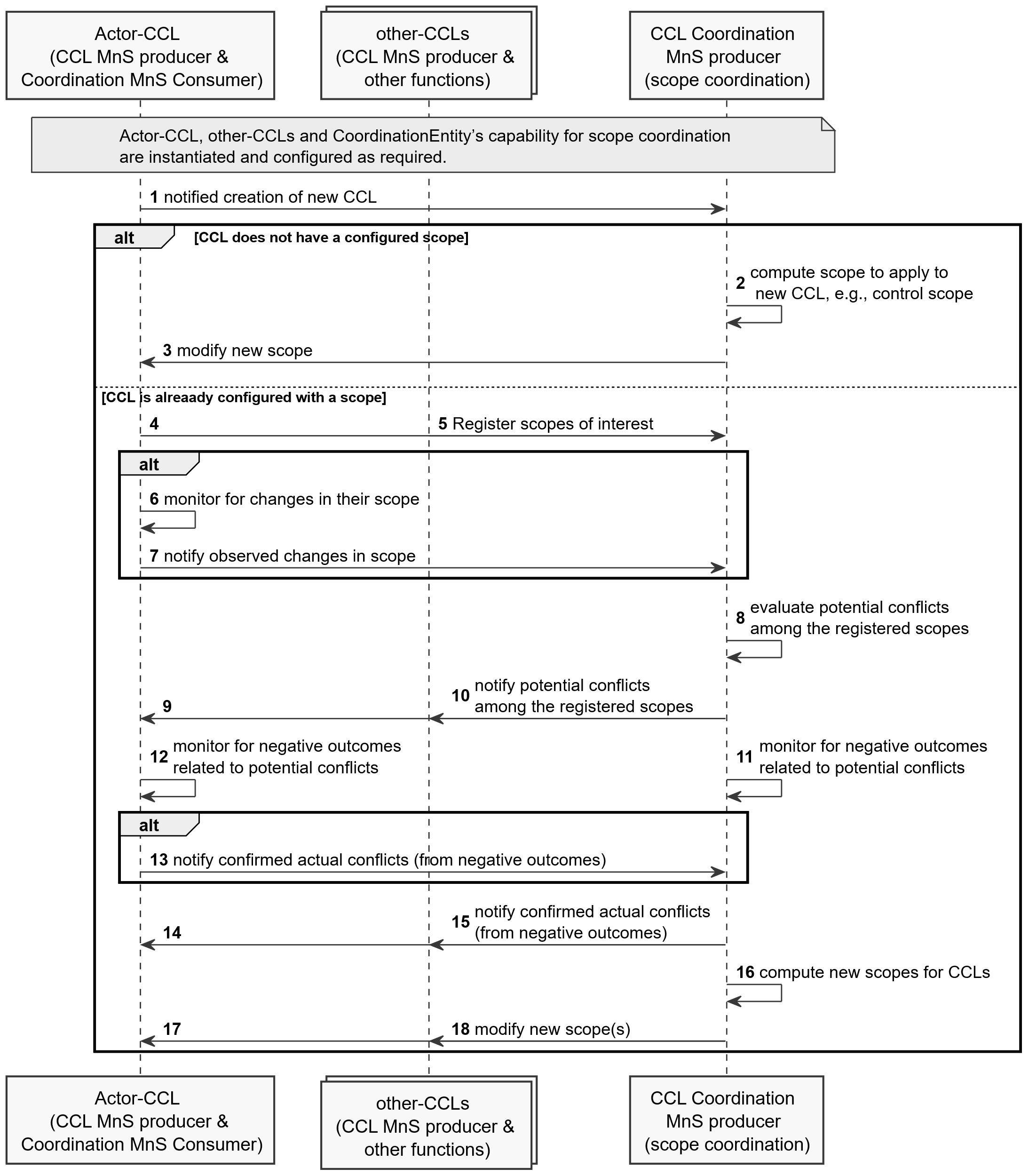


Figure 7.5-1: CCL-impact assessment and actual metric-value conflicts resolution

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.

\* \* \* Third Change \* \* \* \*

# 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.A-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 a 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 affetced 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.A CCL Scope conflicts avoidance, detection and resolution (Figure 7.A-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.A-1 CCL-Scope conflicts avoidance, detection and resolution**