**3GPP TSG-SA5 Meeting #162 *S5-253336***

Gothenburg, Sweden, 25 – 29 August 2025

**Source: Nokia**

**Title: Pseudo-CR on CCL Action conflict 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 Action conflict Coordinationas was agreed in the CCL study in TR28. 867

**Proposed Changes**

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

# 6 Model

## 6.3 Class definitions

### 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 |
| coordinationCapability | M | T | T | F | T |
| cCLActionCoordinationCapability | M | T | T | F | T |
| coordinatedCCLsScopes | M | T | T | F | T |
| cCLActionConflictsHandling | M | T | T | F | T |
| desiredCCLActions | 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.A CCLActionCoordinationCapability <<datatype>>

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

Table 6.3.A.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.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.

### 6.3.12 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.B ActionPlan

#### 6.3.B.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.B.2 Attributes

Table 6.3.11.2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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.B.3 Attribute constraints

None.

#### 6.3.B.4 Notifications

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

### 6.3.13 ActionConflict <<datatype>>

#### 6.3.13.1 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.

#### 6.3.13.2 Attributes

Table 6.3.13.2-1

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

#### 6.3.13.3 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 |
| 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 specific type of CCL conflict coordination capacity | type: CCLScopeCoordinationCapability  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| cCLActionCoordinationCapability | It indicates a specific type of CCL conflict coordination functionality of the ConflictManagementAndCoordinationEntity | type: CCLActionConflictsHandling  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 |
| detectedActionConflicts | 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  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.  Note: The Allowed values are FFS | 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 |
| actionPlanID | It identifies an actionPlan generated by a CCL | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| cCLID | It identifies the DN of a CCL that has generated an actionPlan | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| actions | It indicates the CM changes proposed a CCL | type: PlannedConfigurationDescriptor  multiplicity: 1 ..\*  isOrdered: False  isUnique: True defaultValue: None  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  multiplicity: \*  isOrdered: False  isUnique: False  defaultValue: None  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.  allowedValues: [0, 100] | Type: pair<string,integer>  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  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.  allowedValues: [0, 100] | Type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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>  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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.  allowedValues: [0, 100] | Type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  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  multiplicity: \*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |

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

# 7 Procedures

## 7.C 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.C.

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.

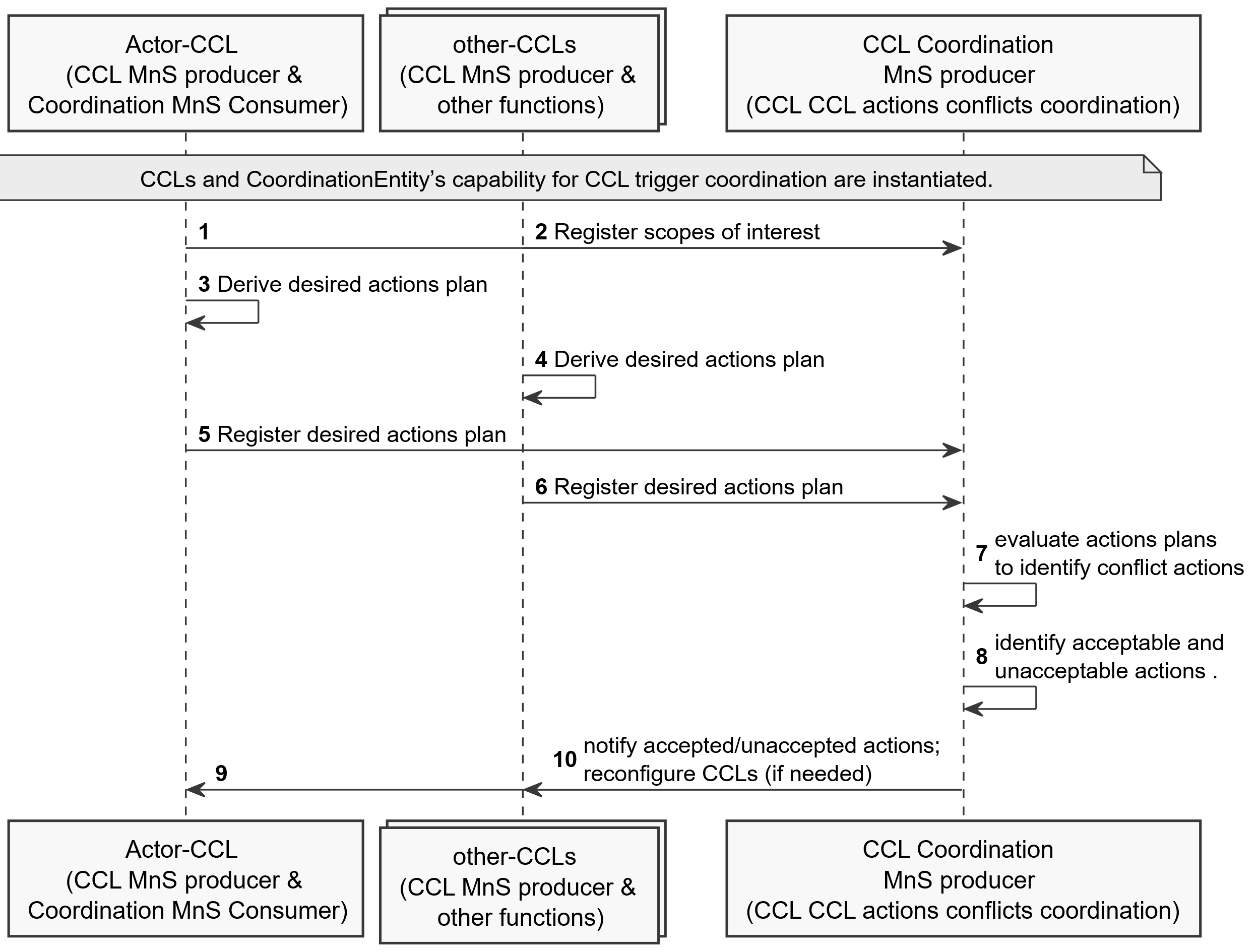


Figure 7.C-1: CCL-impact assessment concurrent actions conflicts resolution

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.D CCL non-concurrent actions conflicts avoidance, detection and resolution

### 7.D.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.D.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.

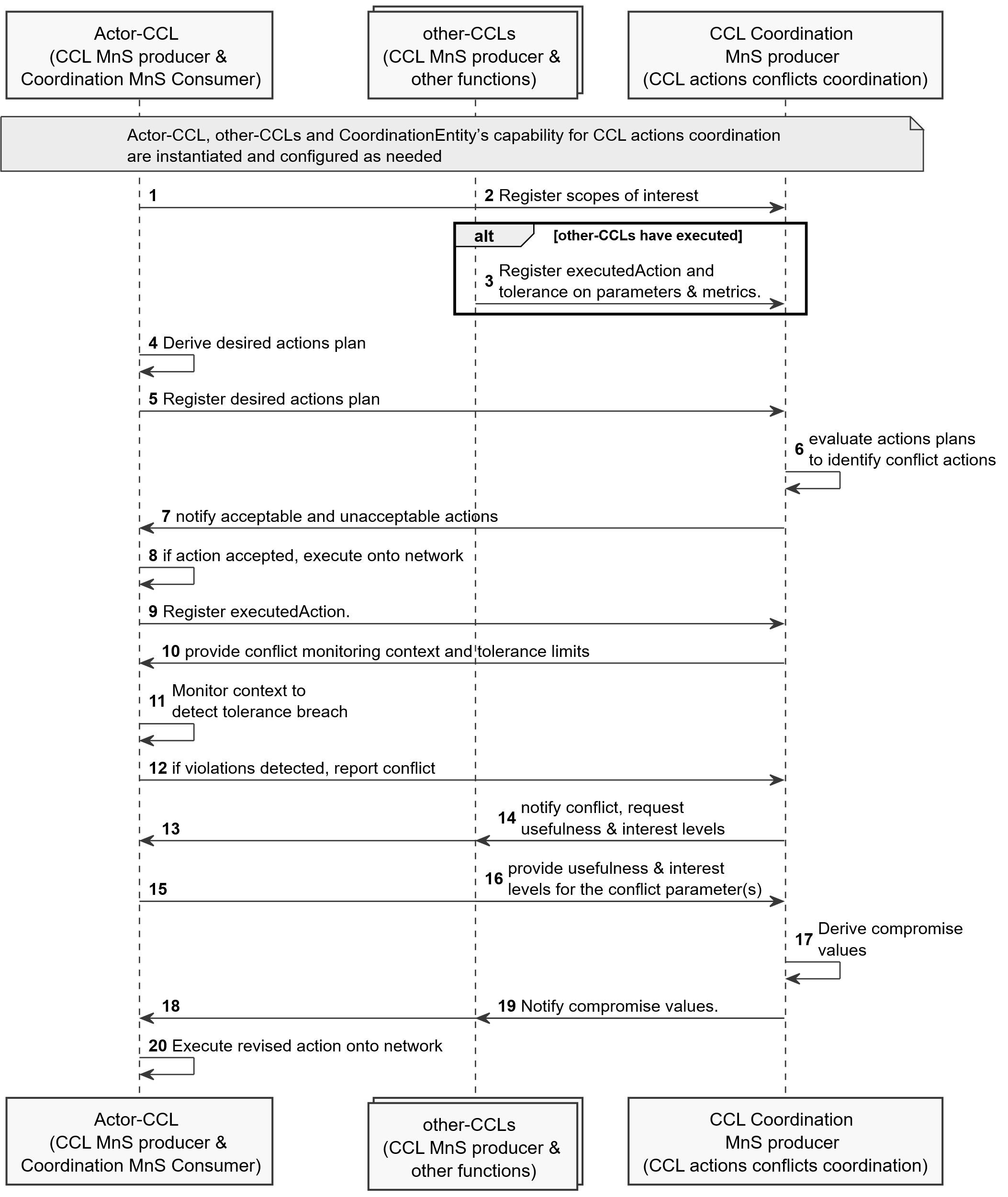


Figure 7.D-1: CCL-impact assessment and actual metric-value conflicts resolution

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

\* \* \* 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.C CCL concurrent actions conflicts avoidance, detection and resolution (Figure 7.C-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.C-1 CCL coordination to avoid, detect and resolve CCL- concurrent actions conflicts**

## B.D CCL non-concurrent actions conflicts CCL coordination to avoid, detect and resolve (Figure 7.D-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.D-1 CCL coordination to avoid, detect and resolve CCL non-concurrent actions conflicts**