

Technical Report

O-RAN Work Group 3 (Near-Real-time RAN Intelligent Controller and E2 Interface)

Conflict Mitigation

Copyright © 2024 by the O-RAN ALLIANCE e.V.

The copying or incorporation into any other work of part or all of the material available in this document in any form without the prior written permission of O-RAN ALLIANCE e.V. is prohibited, save that you may print or download extracts of the material of this document for your personal use, or copy the material of this document for the purpose of sending to individual third parties for their information provided that you acknowledge O-RAN ALLIANCE as the source of the material and that you inform the third party that these conditions apply to them and that they must comply with them.

O-RAN ALLIANCE e.V., Buschkauler Weg 27, 53347 Alfter, Germany

Register of Associations, Bonn VR 11238, VAT ID DE321720189

Contents

List of figures 4

List of tables 4

Foreword 5

Modal verbs terminology 5

1 Scope 6

2 References 6

2.1 Informative references 6

3 Definition of terms, symbols and abbreviations 6

3.1 Terms 6

3.2 Symbols 6

3.3 Abbreviations 6

4 Background and Concepts 7

4.1 Background 7

4.2 Concepts 7

4.2.1 Types of Conflicts 7

4.2.2 Framework for Conflict Mitigation 8

4.3 Working Principles 8

5 Conflict Detection 9

5.1 Key Issues 9

5.1.1 Key Issue #1 Detection of E2-related Potential Conflicts 9

5.1.2 Key Issue #2 Detection of E2-related Direct Conflicts 9

5.1.3 Key Issue #3 Detection of E2-related Indirect and/or Implicit Conflicts 10

5.2 Solutions 11

5.2.1 Solution #1 E2AP level direct conflict detection by Near-RT RIC platform 11

5.2.2 Solution #2 E2SM level direct conflict detection by Near-RT RIC platform 12

5.2.3 Solution #3 E2SM level direct conflict detection by xApps 13

5.2.4 Solution #4 Post-Action Detection of E2-related Indirect and/or Implicit Conflicts 18

5.2.5 Solution #5 Pre-Action Detection of E2-related Indirect Conflicts with user configured parameter dependency information 26

5.2.6 Solution #6 Pre-Action Detection of Potential E2-related Implicit Conflicts with user configured parameter dependency information 28

5.2.7 Solution #7 Pre-Action Detection of E2-related Implicit Conflicts with with user configured parameter dependency information 30

5.3 Solutions-to-Key Issues Mapping 32

6 Conflict Resolution 32

6.1 Key Issues 32

6.1.1 Key Issue #1 Resolution of E2-related Conflicts 32

6.2 Solutions 33

6.2.1 Solution #1 Resolution of Direct Parameter Conflicts between xApps at E2SM level 33

6.3 Solutions-to-Key Issues Mapping 39

7 Conflict Avoidance 39

7.1 Key Issues 39

7.1.1 Key Issue #1 Avoidance of E2-related Conflicts 39

7.2 Solutions 41

7.2.1 Solution #1 xApp based E2SM level service conflict avoidance 41

7.3 Solutions-to-Key Issues Mapping 47

8 Conclusions 47

8.1 Conflict Detection 47

8.1.1 Key issues and potential solutions 47

8.1.2 Solution requirements 47

8.2 Conflict Resolution 48

8.2.1 Key Issues and potential solutions 48

8.2.2 Solution requirements 48

8.3 Conflict Avoidance 48

8.3.1 Key Issues and potential solutions 48

8.3.2 Solution requirements 48

Annex A: Title of annex 50

Annex Bibliography 51

Annex: Change history/Change request (history) 52

# List of figures

[Figure 5.2-1: RICARCH Figure 9.3.3.1-1: xApp initiated E2 guidance request/response procedure 15](#_Toc167466983)

[Figure 5.2-2: RICARCH Figure 9.3.3.5-1 E2 Guidance Modification Procedure 16](#_Toc167466984)

[Figure 5.2-3: Implementation Option 1 (Platform Handles E2SM) 21](#_Toc167466985)

[Figure 5.2-4: Implementation Option 2 (Helper xApp Handles E2SM) 24](#_Toc167466986)

[Figure 6.2-1: Implementation Option 1 (Platform Handles E2SM) 29](#_Toc167466987)

[Figure 6.2-2: Implementation Option 2 (Helper xApp Handles E2SM) 31](#_Toc167466988)

[Figure 7.2-1: RICARCH Figure 9.3.3.1-1: xApp initiated E2 guidance request/response procedure 38](#_Toc167466989)

[Figure 7.2-2: RICARCH Figure 9.3.3.5-1 E2 Guidance Modification Procedure 40](#_Toc167466990)

# List of tables

[Table 5.3-1: Solutions to Key Issue Mapping 7](#_Toc151110841)

[Table 6.3-1: Solutions to Key Issue Mapping 8](#_Toc151110842)

[Table 7.3-1 Solutions to Key Issue Mapping 9](#_Toc151110843)

# Foreword

This Technical Specification (TS) has been produced by WG3 of the O-RAN Alliance.

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

version xx.yy.zz

where:

xx: the first digit-group is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc. (the initial approved document will have xx=01). Always 2 digits with leading zero if needed.

yy: the second digit-group is incremented when editorial only changes have been incorporated in the document. Always 2 digits with leading zero if needed.

zz: the third digit-group included only in working versions of the document indicating incremental changes during the editing process. External versions never include the third digit-group. Always 2 digits with leading zero if needed.

# Modal verbs terminology

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

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

# 1 Scope

The present document provides a technical report on the Conflict Mitigation functions in the Near-RT RIC.

# 2 References

## 2.1 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non‑specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] O-RAN.WG3.RICARCH-v05.00.00: “O-RAN, Near-Real-time RAN Intelligent Controller, Near-RT RIC Architecture”.

[i.2] ORAN Working Group 3, Near-Real-time RAN Intelligent Controller, E2 Service Model, RAN Control (E2SM-RC).

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

## 3.2 Symbols

## 3.3 Abbreviations

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

API Application Programming Interface

CCO Coverage and Capacity Optimization

ConMit Conflict Mitigation

E2SM E2 Service Model

KPI Key Performance Indicator

MRO Mobility Robustness Optimization

Near-RT RIC Near-Real-time RAN Intelligent Controller

RAN Radio Access Network

RICARCH RAN Intelligent Controller Architecture

SMO Service Management and Orchestration

# 4 Background and Concepts

## 4.1 Background

The principle that the Near-RT RIC includes support for Conflict Mitigation has been part of RICARCH [i.1] since the first version was published in 2021. Conflict Mitigation in Near-RT RIC is a necessary functionality because xApps objectives may be chosen/configured such that they result in conflicting actions. Since then, however, the only change has been the addition of "E2 Guidance" procedure into clause 9.3 [i.1] for E2 Subscription and E2 Control procedures and the split of the v01.00 text into a brief statement in clause 6.2.3 and additional details in Annex A.

This state is sufficient if the Near-RT RIC platform hosts all aspects of conflict mitigation and hence but is insufficient, if, and when, deeper analysis reveals new requirements related to conflict mitigation that involve either inter-action with xApps and/or the SMO domain.

The purpose of the TR is therefore an analysis of the different types of conflicts defined in clause 4.2.1 in terms of methods for Conflict Detection, Resolution and Avoidance with the objective of identifying when and if new RIC APIs, new O1 use cases, and/or extensions to E2SMs are required to support Conflict Mitigation. This identification shall result in "key issues" and their " potential solutions" for Conflict Detection, Resolution and Avoidance.

## 4.2 Concepts

### 4.2.1 Types of Conflicts

Conflicts between xApps can be broadly divided into three types.

1. Direct Conflicts: The conflicts can be observed directly. Some cases are described as below:

* Two or more xApps request different settings for the very same configuration of one or more control parameters.
* The new control or policy request from an xApp may conflict with the running configuration resulting from a previous request of another or the same xApp.
* Two xApps may not be provided services by a RAN Function simultaneously, e.g., due to lack of sufficieint resources. In this sense, the two xApps shall directly compete for the same RAN resource.

1. Indirect Conflicts: The conflicts cannot be observed directly, nevertheless, some dependence among the parameters and resources that the xApps target can be observed. For instance, different xApps target different configuration parameters to optimize the same metric according to their respective objectives. Even though this will not result in conflicting parameter settings, it may have uncontrollable or inadvertent system impacts. One example of such indirect conflicts can occur when the changes required by one xApp create a system impact which is equivalent to a parameter change targeted by another xApp. For example, antenna tilts and measurement offsets are different control points, but they both impact the handover boundary.
2. Implicit Conflicts: The conflicts cannot be observed directly, even the dependence between xApps is not obvious. For instance, different xApps may optimize different metrics and (re-)configure different parameters. Nonetheless, optimizing one metric may have implicit, unwanted, and maybe adversary side effects on one of the metrics optimized by another xApp. E.g., protecting throughput metrics for GBR users may degrade non-GBR metrics or even cell throughput.

### 4.2.2 Framework for Conflict Mitigation

A comprehensive Conflict Mitigation framework consists of three procedures:

1. Conflict Detection: Detection of potential and actual direct, indirect, and implicit conflicts.
2. Conflict Resolution: Resolution of ongoing conflicts, possibly beyond simple xApp enablement or disablement.
3. Conflict Avoidance: Providing guidance and information to xApps in order to avoid future conflicts.

The above procedures should be enabled to make use of Near RT RIC platform functionalities and they may require services provided by xApps. These services and functionalities may include AI/ML methods and optimization techniques etc. For this reason, the above procedures may be seen as “use-cases”. However, these use-cases do not deal with network performance KPIs directly, such as the use-cases considered in UCTG WG1. In contrast, these procedures enable xApps to execute their business logic safely in an environment in which there could be many multi-vendor xApps with different business logic and objectives.

It is clear that the above procedures are potentially more “intelligent” than other native functionalities of Near RT RIC platform such as data base, AI/ML training, xApp subscription management etc. In particular, Conflict Mitigation is likely to need to decode/understand E2SM level data such as network KPIs and control parameters.

One design option would be to introduce support for specialized "conflict mitigation related xApps" to perform these roles. This approach opens up a significant opportunity for companies to develop intelligent solutions, in form of xApps, that provide conflict mitigation related value added services to Conflict Mitigation in the platform.

The above text discusses only discusses a few possibilities that may be considered. The precise "key Issues” arising in the Conflict Mitigation framework and their solutions shall be discussed in detail in Chapters 5, 6, and 7.

## 4.3 Working Principles

In light of the RIC ARCH specification [i.1] clause 6.2.3 and the WI description, the current scope of this TR is restricted to mitigation of conflicts in xApp actions over the E2 interface. The motivations behind xApp actions (e.g., O1 configuration or A1 policies etc.) are not covered currently by the TR in context of conflict mitigation. However, this work may serve as the basis for and complimentary to further future work covering A1 and O1 aspects as well.

# 5 Conflict Detection

## 5.1 Key Issues

### 5.1.1 Key Issue #1 Detection of E2-related Potential Conflicts

#### 5.1.1.1 Description

Potential conflicts (i.e., future probable conflicts) between xApp actions may be detected *pre-action* by the Near-RT RIC platform, i.e., before RIC Control Request or RIC Subscription Request messages are sent to the E2 Node. At least for direct conflicts, this conflict mitigation functionality shall allow for pre-action xApp conflict resolution and conflict avoidance. Information obtained from this procedure may be used during confirmation/detection of actual conflicts discussed in later clauses.

NOTE: This key issue is restricted to only pre-action detection of “potential” conflicts, i.e., these conflicts may demand further investigation, but they may not result in “actual” conflicts discussed in later clauses.

#### 5.1.1.2 Impacted Use-Cases

All use cases impacted by, at least, E2-related direct conflicts. See clause 5.1.2.2.

#### 5.1.1.3 Considerations

The following considerations should be kept in mind when providing solutions to this key issue:

1. Near-RT RIC platform’s conflict mitigation (ConMit) functionality may be used. The platform may obtain information from xApp Subscription Management to perform this task.
2. The detection of potential conflicts by the ConMit functionality may be restricted to using only E2AP level information.
3. The xApp control actions do not need to be simultaneous for them to cause conflicts. For example, if an xApp’s control target is a particular cell, then it may “potentially” cause a conflict with an existing running xApp subscription whose control target is the same cell.
4. Because xApps are likely to repeat their behavior, previously detected potential conflicts may be considered for future E2 Guidance [i.1].

#### 5.1.1.4 Open issues

void.

### 5.1.2 Key Issue #2 Detection of E2-related Direct Conflicts

#### 5.1.2.1 Description

Direct conflicts occur due to overlapping RAN Parameters and/or overlapping RAN resources between xApps, e.g., when two xApps request to configure the same RAN parameter(s). See clause 4.2.1 for description of conflict types.

#### 5.1.2.2 Impacted Use-Cases

Several use-cases may share the same control parameter, or more generally, access the same RAN resources. For example, handover parameters may be configured by xApps for Mobility Robustness Optimization (MRO), Traffic Steering and/or Load-Balancing.

#### 5.1.2.3 Considerations

The following considerations should be kept in mind when providing solutions to this key issue:

1. Detection of direct conflicts may use E2AP and/or E2SM specific information.
2. Detection of direct conflicts may happen after further investigation of detected potential conflicts.
3. For example, even if two or more xApps target the same RAN control parameter(s), which may indicate a potential conflict, their specific control or policy actions may ultimately decide whether the resulting configurations are conflicting. A typical example of a direct conflict is when the first xApp suggests increasing the value of a RAN control parameter while the second xApp suggests decreasing it.
4. Conflicts may be detected pre-action (i.e., before RIC Control Request or RIC Subscription Request messages have been sent to the E2 Node) or post-action (i.e., after RIC Control Request or RIC Subscription Request messages have been sent to the E2 Node.
5. xApps actions may be conflicting directly even if they are not simultaneous.
6. Direct conflicts may not just involve RAN control parameters. Two xApps may request the same RAN resources or the sum of RAN resources required by xApps may exceed the capacity of the RAN node.
7. If the two xApps target different KPIs, then KPI improvement/degradation or potential KPI improvement/degradation and their relative importance may also be considered at this stage. For example, if actions of one xApp do not degrade KPIs of other xApps, then this may not be considered as a conflict.

#### 5.1.2.4 Open issues

The following issues are open:

1. If E2SM-specific information is used to detect conflicts, it's unclear whether E2SM inspection may be performed by the platform or by the conflicting xApps through direct or indirect communication.
2. If the detection is performed by the platform using E2SM-specific information, the platform shall need to “understand/decode” this information.
3. Privacy Issue: The platform may not have access to E2SM information specific to an xApp. For example, the platform may not have access to details of xApp’s E2SM control/policies.

### 5.1.3 Key Issue #3 Detection of E2-related Indirect and/or Implicit Conflicts

#### 5.1.3.1 Description

During indirect and/or implicit conflicts, xApps target different RAN control parameters (i.e., they do not conflict directly) but there may be some known or unknown coupling/dependence between them (see clause 4.2.1 in Chapter 4). This dependence may be in terms of their KPIs (which may be overlapping or different) or the impacts of their actions on the overall RAN state, i.e., actions of one xApp may cause the RAN state to change in such a way that previous actions of another xApp are no longer optimal for this new RAN state. Unless such conflicts are resolved/managed, such dependencies can cause uncontrollable system oscillations and performance degradations.

#### 5.1.3.2 Impacted Use-Cases

It is likely that many use cases are impacted by indirect and/or implicit conflicts including the already specified use cases. An example is the change in handover boundary (by e.g., MRO) which shall have an impact on xApps implementing Energy Savings, Traffic Steering (TS), Load Balancing, and Coverage and Capacity (CCO) etc. functionality.

#### 5.1.3.3 Considerations

The following considerations should be kept in mind when providing solutions to this key issue:

1. Detection may use E2AP and/or E2SM-specific information.
2. Detection of indirect and/or implicit conflicts may happen after further investigation of detected potential conflicts.
3. Conflicts may be detected pre-action (i.e., before RIC Control Request or RIC Subscription Request messages have been sent to the E2 Node) or post-action (i.e., after RIC Control Request or RIC Subscription Request messages have been sent to the E2 Node.
4. The same solution may work for both indirect and implicit conflicts.
5. Only “harmful” conflicts may be detected. This may be justified by the fact that many RAN KPIs are coupled with each other, and achieving an entirely conflict-free system may be impossible. In such cases, we can consider managing conflicts rather than removing them completely.
6. The detection of indirect and/or implicit conflicts between xApps may be significantly challenging and it may require state-of-art AI/ML methods.

#### 5.1.3.4 Open issues

Same as discussed in clause 5.1.2.4.

## 5.2 Solutions

### 5.2.1 Solution #1 E2AP level direct conflict detection by Near-RT RIC platform

#### 5.2.1.1 Introduction

Key issue #1 for Conflict Detection (clause 5.1.1) describes the need for potential *pre-action* conflict detection by the Near-RT RIC platform for the specific case where the ConMit functionality in the Near-RT RIC platform is restricted to using only E2AP level information.

This solution assumes that only E2AP level information {E2 Node Id, RAN Function ID, RIC Action Type} may be used by the platform during the RICARCH [i.1] defined xApp initiated E2 Subscription procedure ([i.1] clause 9.3.2.1), E2 Control procedure ([i.1] clause 9.3.2.4) and E2 Guidance request/response procedure ([i.1] clause 9.3.3.1). any additional information contained within E2SM specific encoded IEs would be transparent to the Near-RT RIC Platform.

#### 5.2.1.2 Functional Description

The Conflict Mitigation functionality in the Near-RT RIC platform ([i.1] clause 6.2.3) is assumed to handle E2AP level information from a requesting xApp and may compare this with similar information obtained from other xApps from previous E2 Subscription, E2 Control and/or E2 Guidance requests.

An “E2AP level” potential conflict would be detected when and if the xApp requests the same RIC Action Type (particularly for POLICY) from E2 Subscription Request or CONTROL from E2 Control Request, for the same RAN Function ID on the same E2 Node ID as was previously used by another xApp.

#### 5.2.1.3 Procedures

For the E2AP Level conflict detection process, RICARCH [i.1] clause 9.3.3.1 (E2 guidance request/response procedure) step 3 “Near-RT RIC platform processes request” and clause 9.3.3.5 (E2 Guidance modification procedure) step 1 “Near-RT RIC platform identifies affected xApps and performs conflict mitigation processing” covers the scope of this solution.

#### 5.2.1.4 Evaluation

This solution is, by itself, sub-optimal as E2AP level potential conflict detection would often generate false positive detections whenever multiple xApps attempt to use the same RIC Action Type (i.e. POLICY) but with different target services.

Examples include:

- For a RAN function defined using E2SM-RC [i.2] when the different xApps are requesting POLICY services using different RIC Style Types. In this case there would not be a direct conflict but this would require analysis at the E2SM level and not just the E2AP level.

### 5.2.2 Solution #2 E2SM level direct conflict detection by Near-RT RIC platform

#### 5.2.2.1 Introduction

Key issue #1 for Conflict Detection (clause 5.1.1) describes the need for potential *pre-action* conflict detection by the Near-RT RIC platform for the specific case where the ConMit functionality in the Near-RT RIC Platform is able to use both E2AP and E2SM level information.

Key issue #2 for Conflict Detection (clause 5.1.2) describes the possibility of “E2SM inspection may be performed by the platform”.

This solution assumes that after E2AP level information {E2 Node Id, RAN Function ID, RIC Action Type} analysis the platform may then also perform E2SM level analysis during the RICARCH [i.1] defined xApp initiated E2 Subscription procedure ([i.1] clause 9.3.2.1), E2 Control procedure ([i.1] clause 9.3.2.4) and E2 Guidance request/response procedure ([i.1] clause 9.3.3.1).

#### 5.2.2.2 Functional Description

Solution #1 describes how the Near-RT RIC platform ([i.1] clause 6.2.3) would use E2AP level information from a requesting xApp and compare this with similar information obtained from other xApps from previous E2 Subscription, E2 Control and/or E2 Guidance requests. The result would be a “E2AP level” potential conflict detection.

In this solution the Conflict Mitigation functionality in the Near-RT RIC platform may then inspect E2SM level information from both the requesting xApp and the previous requests selected using E2AP level detection to determine if a direct conflict is likely to exist.

Examples include:

- For a RAN function defined using E2SM-RC [i.2] when the different xApps are requesting POLICY services using the same RIC Style Type (i.e. Style 1 for Radio Bearer Control), same RIC Action ID (i.e. ID 2 for QoS flow mapping configuration). In this case there would be a significant chance that the request is in direct conflict with a previous request from another xApp and that a Conflict Avoidance mechanism would be required.

#### 5.2.2.3 Procedures

For the E2AP level conflict detection process, RICARCH [i.1] clause 9.3.3.1 (E2 guidance request/response procedure) step 3 “Near-RT RIC platform processes request” and clause 9.3.3.5 (E2 Guidance modification procedure) step 1 “Near-RT RIC platform identifies affected xApps and performs conflict mitigation processing” covers the scope of this solution.

In this case the Near-RT RIC platform would process both E2AP and E2SM level information.

#### 5.2.2.4 Evaluation

This solution offers a sound solution for both E2AP and E2SM level direct conflict detection with a low chance of false positive detections.

The solution does however suffer from the need to align Near-RT RIC platform support of each E2SM with the versions used by each xApp which may prove to be difficult.

### 5.2.3 Solution #3 E2SM level direct conflict detection by xApps

#### 5.2.3.1 Introduction

Key issue #1 for Conflict Detection (clause 5.1.1) describes the need for potential *pre-action* conflict detection by the Near-RT RIC platform for the specific case where the ConMit functionality in the Near-RT RIC Platform is restricted to using only E2AP level information.

Key issue #2 for Conflict Detection (clause 5.1.2) describes the possibility of “E2SM inspection may be performed by the platform or by the conflicting xApps through direct or indirect communication”.

This solution assumes that after E2AP level information {E2 Node Id, RAN Function ID, RIC Action Type} analysis the platform may request that one or more xApps perform E2SM level analysis during the RICARCH [i.1] defined xApp initiated E2 Subscription procedure ([i.1] clause 9.3.2.1), E2 Control procedure ([i.1] clause 9.3.2.4) and E2 Guidance request/response procedure ([i.1] clause 9.3.3.1).

#### 5.2.3.2 Functional Description

Solution #1 for Conflict Detection describes how the Near-RT RIC platform ([i.1] clause 6.2.3) would use E2AP level information from a requesting xApp and compare this with similar information obtained from other xApps from previous E2 Subscription, E2 Control and/or E2 Guidance requests. The result would be a “E2AP level” potential conflict detection.

In this solution the Conflict Mitigation functionality in the Near-RT RIC platform may then request both the requesting xApp and/or one or more of the previous xApp selected using E2AP level detection to then inspect the E2SM level information from the requesting xApp and the previous requests selected using E2AP level detection to determine if a direct conflict is likely to exist.

Examples include:

- For a RAN function defined using E2SM-RC [i.2] when the different xApps are requesting POLICY services using the same RIC Style Type (i.e. Style 1 for Radio Bearer Control), same RIC Action ID (i.e. ID 2 for QoS flow mapping configuration). In this case there would be a significant chance that the request is in direct conflict with a previous request from another xApp and that a Conflict Avoidance mechanism would be required.

#### 5.2.3.3 Procedures

For the E2AP level conflict detection process, RICARCH [i.1] clause 9.3.3.1 (E2 guidance request/response procedure) step 3 “Near-RT RIC platform processes request” and clause 9.3.3.5 (E2 Guidance modification procedure) step 1 “Near-RT RIC platform identifies affected xApps and performs conflict mitigation processing” only covers the E2AP level analysis described in this solution. These procedures would need to be modified to so that the Near-RT RIC platform may seek E2SM level analysis from one or more xApps.

**RICARCH 9.3.3 E2 Guidance related procedures**

**RICARCH 9.3.3.1 E2 Guidance request/response procedure**

The purpose of the xApp initiated E2 Guidance request/response API procedure in the Near-RT RIC is to allow authorized xApp to obtain guidance from the Near-RT RIC platform (with the conflict mitigation functionality) prior to initiating an action.

Guidance from Near-RT RIC platform may include:

- Indication on whether or not the xApp proposed E2 Related API message or series of messages may result in a conflict with E2 related API messages from other xApps;

- Recommendations on how the proposed E2 Related message or series of messages should be modified to avoid conflict;

- Modification of previous guidance to other xApps and/or Near-RT RIC platform internal processes.

This procedure is based on the following assumptions:

- xApp may use E2 Related API to obtain guidance from Near-RT RIC platform to resolve potential conflicts prior to initiating a RIC function procedure;

- xApp may use Near-RT RIC platform response in a subsequent procedure (i.e., for a RIC Functional Procedure).

This procedure is initiated by an xApp using **E2 Related API: E2 Guidance** request. The following outcomes are considered:

- Near-RT RIC Platform provides guidance to requesting xApp using E2 Related API: E2 Guidance response;

- Near-RT RIC Platform provides modified guidance to another xApp and/or its internal processes.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | xApp initiation of Conflict Mitigation guidance. |
| Actors and Roles | * xApp: Originator of Conflict Mitigation guidance request; * Near-RT RIC platform, with the following functionalities: * Database; * Conflict mitigation. |
| Assumptions | * Near-RT RIC platform has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution; * Near-RT RIC platform may initiate guidance towards other xApp as an optional addition response to Guidance Request. |
| Pre-conditions | * xApp has been authorized to request guidance from Near-RT RIC platform for conflict mitigation; * xApp has been assigned xApp ID. |
| Begins when | xApp determines need to request guidance from Near-RT RIC platform for conflict mitigation. |
| Step 1 (M) | xApp sends **E2 Related API E2 Guidance** Request to Near-RT RIC platform. |
| Step 2 (M) | Near-RT RIC platform collects related information. |
| Step 3A (M) | (MODIFIED) Near-RT RIC platform performs conflict mitigation detection |
| Step 3b (O) | (NEW) Near-RT RIC platform requests identified xApps to perform conflict mitigation detection |
| Step 3c (O) | (NEW) Near-RT RIC platform performs conflict mitigation processing. |
| Step 4-5 (O) | Near-RT RIC platform may signal conflict and/or provide guidance to internal processes and/or other xApps (see clause 9.3.3.5). |
| Step 6 (M) | Near-RT RIC platform sends **E2 Related API E2 Guidance** response to xApp. |
| Ends with | xApp continues processing using guidance response. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

box “Near-RT RIC”

Collections “Other xApp” as xApp

participant “xApp” as xApp2

Participant plt as "Near-RT RIC platform"

end box

xApp2 -> plt: 1: (E2 related API) E2 Guidance (Request)

plt -> plt: 2: Collect related information

plt -> plt: 3a: (MODIFIED)Conflict mitigation detection

Alt If platform has identified a potential conflict

plt<-->xApp2: 3b. (NEW)Conflict mitigation detection

plt<-->xApp: 3b. (NEW)Conflict mitigation detection

end

plt -> plt: 3c. (NEW)Conflict mitigation processing

plt --> plt: 4: Guidance to internal processes

xApp <-- plt: 5: (E2 related API) E2 Guidance (Modification)

plt -> xApp2: 6: (E2 related API) E2 Guidance (Response)

@enduml

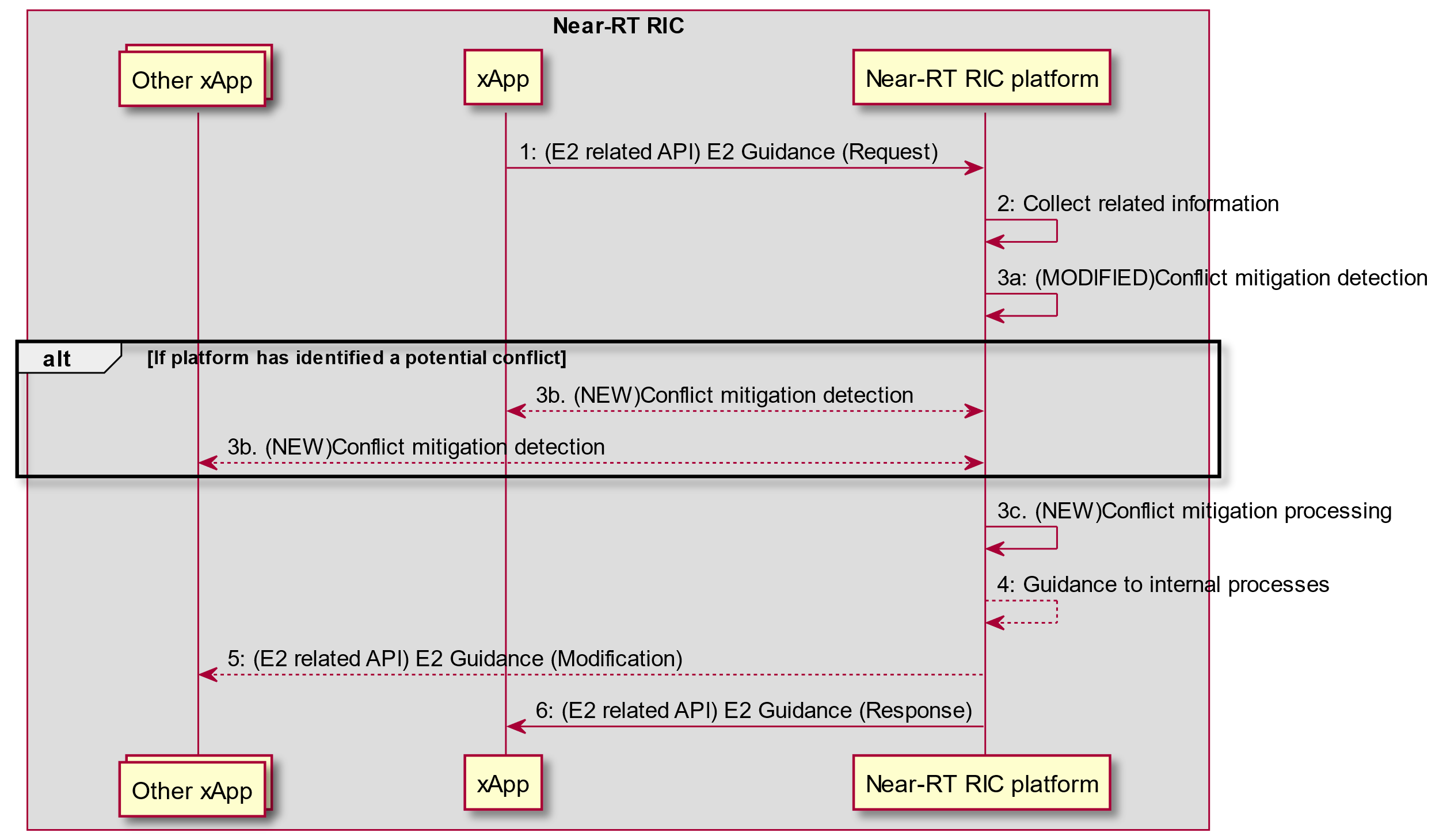


Figure 5.2-1: RICARCH Figure 9.3.3.1-1: xApp initiated E2 guidance request/response procedure

**RICARCH 9.3.3.5 E2 Guidance modification procedure**

The purpose of the E2 Guidance modification procedure is to allow Near-RT RIC platform to signal a modified guidance to the xApp, which can be triggered by the Near-RT RIC platform’s internal process (e.g., xApp subscription management), or by a message from the other xApps or external interfaces.

The guidance may also be used for Near-RT RIC platform internal purposes.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | Near-RT RIC platform indicates guidance to an xApp without request from the xApp. |
| Actors and Roles | * xApp * Near-RT RIC platform, with the following functionalities: * Database * Conflict mitigation * External entity (e.g., E2 Node) |
| Assumptions | * Near-RT RIC platform is capable of collecting related information, and generating proper guidance for an xApp. |
| Pre-conditions | * Message from xApp, message from external interfaces, or Near-RT RIC platform’s internal process triggers conflict mitigation in Near-RT RIC platform. |
| Step 1a (M) | (MODIFIED) Near-RT RIC platform identifies potentially affected xApps and performs conflict mitigation processing. |
| Step 1b (O) | (NEW) Near-RT RIC platform requests identified xApps to perform conflict mitigation detection |
| Step 1c (O) | (NEW) Near-RT RIC platform performs conflict mitigation processing. |
| Step 2 (O) | Near-RT RIC platform may provide guidance to the affected xApp using E2 Guidance modification message. |
| Step 3 (O) | Near-RT RIC platform may also use the guidance for internal purposes. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Box "Near-RT RIC"

Participant xApp

Collections “Other xApp(s)” as OtherxApp

Participant “Near-RT RIC platform” as plt

endbox

Collections “External entity (e.g. Non-RT RIC)” as Otherentity

Alt message from other xApp(s) triggers conflict mitigation

OtherxApp -> plt: Message that may introduce conflict

Else message from external entity triggers conflict mitigation

Otherentity -> plt: Message that may introduce conflict

Else internal processing triggers conflict mitigation

plt -> plt: internal process that may introduce conflict

End

plt -> plt: 1a.(MODIFIED)Conflict mitigation detection

Alt If platform has identified a potential conflict

plt<-->xApp: 1b. (NEW)Conflict mitigation detection

plt<-->OtherxApp: 1b. (NEW)Conflict mitigation detection

end

plt -> plt: 1c. (NEW)Conflict mitigation processing

plt --> xApp: 2: E2 Guidance (Modification)

plt --> plt: 3: Guidance for internal purposes

@enduml

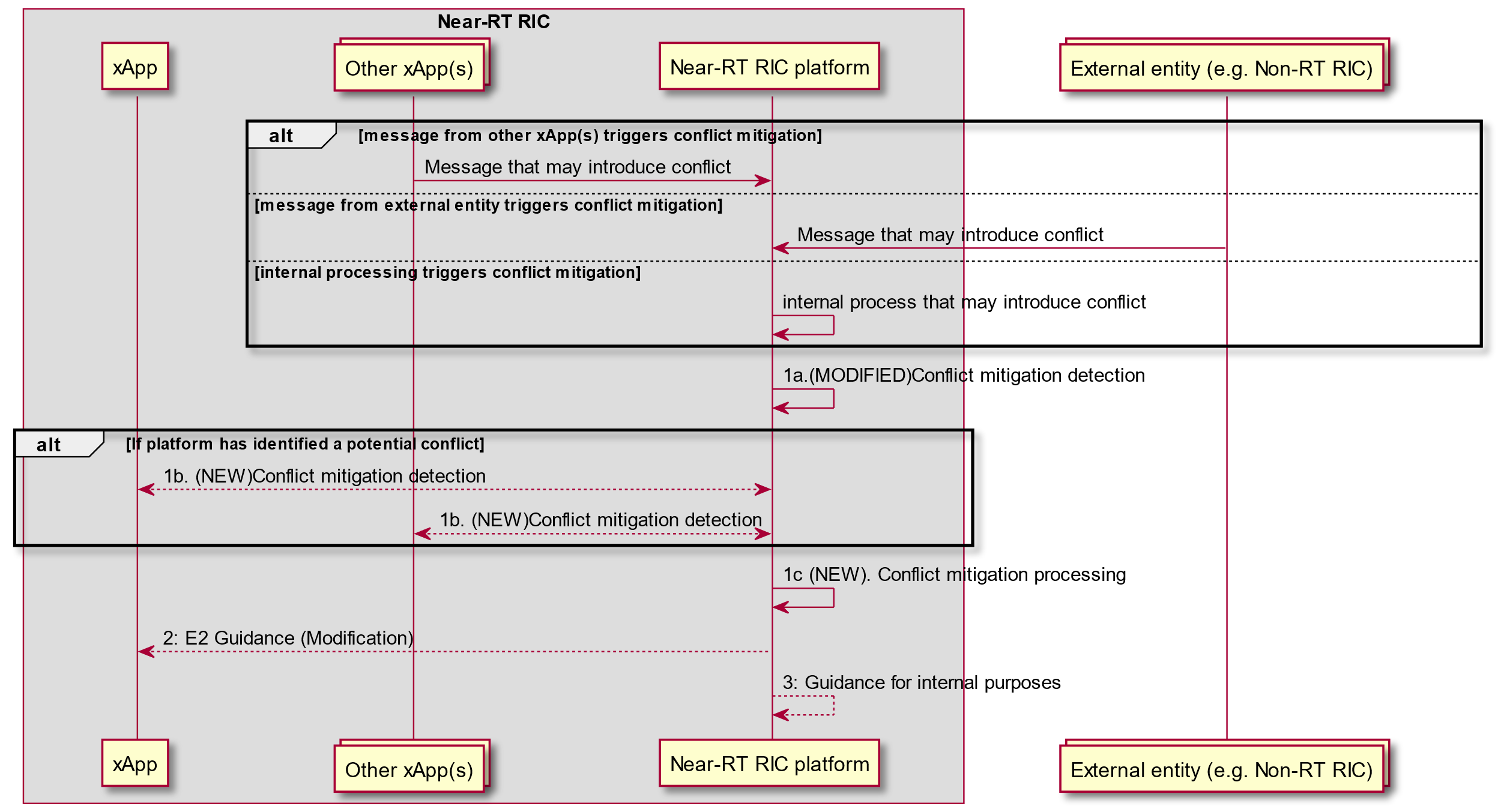


Figure 5.2-2: RICARCH Figure 9.3.3.5-1 E2 Guidance Modification Procedure

#### 5.2.3.4 Evaluation

This solution offers a sound solution for both E2AP and E2SM level direct conflict detection with a low chance of false positive detections.

The solution avoids the need to align Near-RT RIC platform support of each E2SM with the versions used by each xApp. It does however rely on xApps supporting the appropriate E2SM versions.

### 5.2.4 Solution #4 Post-Action Detection of E2-related Indirect and/or Implicit Conflicts

#### 5.2.4.1 Introduction

Key issue #1 for Conflict Detection (clause 5.1.1) describes the need for potential *pre-action* conflict detection by the Near-RT RIC platform for the specific case where the ConMit functionality in the Near-RT RIC platform may use both E2AP and E2SM level information.

Key issue#3 for Conflict detection (clause 5.1.3) describes the possibility that detection of indirect and/or implicit conflicts may happen after further investigation of detected potential conflicts. However, an E2AP level conflict may not be confirmed at E2SM level *pre-action*. This may happen especially in case of indirect or implicit conflicts (see clause 4.2.1 and Key issue #3 in clause 5.1.3). Such conflicts are difficult to detect because in general xApp objectives based on some RAN KPIs are often “coupled” and complete decoupling may be impossible to obtain. The platform, therefore, may decide to allow the xApps to perform their respective policy or control actions on the E2 Node for as long as certain KPIs are not affected beyond an unacceptable level *post-action*. This means that, in such cases, the further investigation mentioned above (after detection of a potential conflict) must wait until after the affects of xApps’ actions are observable in the E2 Node. See clause 5.1.1.3 for considerations regarding indirect/implicit conflicts.

The proposed solution (detailed in clauses 5.2.4.2 and 5.2.4.3) involves monitoring and detection of indirect and/or implicit conflicts post-action (as considered in clause 5.1.1.3) after obtaining some conflict relevant information from the xApps and observing RAN performance metrics relevant to the xApps’ objectives. The analysis happens in a limited context which is constructed from the information obtained from xApps’ subscription(s) (e.g., E2 Node, RAN Function, timing of associated RIC actions etc.) and the specific information provided by the xApps.

NOTE: Since the platform may not be provisioned to handle E2SM information, the solution proposes two alternatives for E2SM processing. In implementation option 1, the ConMit functionality in the platform handles E2SM processing. In implementation option 2, a ConMit helper xApp handles E2SM processing and subscribes to the E2 Node to collect required information.

#### 5.2.4.2 Functional Description

The solution assumes the following prerequisites for the required conflict context:

* A potential conflict between xApps has been previously detected pre-action involving an E2 Node. For example, two xApps have taken policy or control related actions at the same E2 Node.
* The potentially conflicting xApps have been identified.
* A direct conflict has not been confirmed at the E2SM level pre-action. This analysis/inspection, however, may result in E2SM knowledge specific to potentially conflicting xApps, e.g., any RAN parameters contained in xApp’s subscription.

Overall Description: Post-action Conflict monitoring may be triggered by a request from an xApp, which is aware of a potential post-action conflict, and it has detected abnormalities/oscillations in it’s KPI(s). The solution proposes that the identified xApps provide the platform a set of RAN KPIs to be monitored. An xApp may provide the platform optional additional information such as, latest monitored values of these KPIs, the minimum acceptable values of these KPIs, and relevant RAN control parameters (NOTE: relevant control parameters may already be known to the platform during the prerequisite steps). The set of KPIs (per xApp) represents those E2SM defined KPIs in the E2 Node that the xApps consider to be important in terms of their own performance goals. If the provided information is insufficient to detect conflicts (e.g., if latest KPI values are not available), then ConMit must obtain the required data from the RAN E2 Node. After obtaining KPI data and other information, the ConMit service can monitor for any interdependence between xApp actions post-action using all the available information.

As an example, consider 2 xApps that request the platform to monitor two KPIs and they may provide some additional optional information as mentioned above. The solution can be extended to any number of xApps. KPI1 is identified by xApp1 and KPI2 is identified by xApp2. By monitoring the relative changes in the two KPIs, the E2AP and E2SM level subscription information (obtained from previous conflict mitigation processing mentioned in the assumptions above), and the additional optional information provided by the xApps, the platform may detect a certain “harmful interdependence” between the two xApps. The platform may use advance AI/ML methods or some other algorithms to perform this task. For example, if KPI1 degrades beyond a certain performance minimum in correlation with KPI2, this indicates a harmful coupling between the two xApps, i.e., a post-action indirect/implicit conflict has been detected. Another possibility is that, if KPI1 and KPI2 are the same, actions of the two xApps may cause oscillations as they change different RAN parameter values according to different objectives (see clause 4.2). The platform may use side-information to further enhance this detection (such as any faults reported by the E2 Node) to isolate the degradation cause as: *a (possibly abnormal) degradation/oscillation caused by a conflict between xApps’ E2 related actions*. The exact solution for this analysis is left for implementation.

NOTE: The list of potentially affected KPIs, which the platfom obtains from the xApps, is the minimum information required for effective indirect conflict detection. However, even if the platform does not obtain added information from the xApps describing the unacceptable level of degradation or related RAN parameters, the platform may still detect abnormalities and dependence in KPIs.

As such, the solution consists of the following steps:

* Initialization: Trigger to monitor and detect indirect and/or implicit conflicts between certain xApps post-action. This part may be labelled as conflict mitigation related pre-processing as in E2 Gudiance API description in [i.1]. The platform may decide on its own to monitor for conflicts. Alternatively, the trigger may come from an xApp as mentioned above. At this point, it is assumed that, due to the earlier potential conflict detection, the platform is already aware of potentially conflicting xApps (i.e., which xApps are acting on the RAN Node in question). If there are no such other xApps, the platform may infom the requesting xApps of the same immediately.
* E2 Guidance: A list of E2SM defined KPIs provided by xApps to be monitored for post-action conflicts along with optional additional information that may support the platform to detect unacceptable/abnormal degradation and inter-dependence in xApp actions.
* Conflict Context: The ConMit functionality or a helper ConMit xApp constructs the conflict context containing identification of the E2 Node, the RAN Function, and KPIs to be monitored etc. The E2AP information is assumed to be available at this point because of the earlier potential conflict detection.
* E2SM reporting & Monitoring: If required, the ConMit functionality or a helper ConMit xApp subscribes to the E2 Node for data collection in the limited conflict context. Post-action KPI monitoring is performed by the platform either directly or by a ConMit helper xApp.
* Data analysis: Processing for conflict detection performed by either the platform itself or by the ConMit helper xApp. This processing is left for implementation.
* E2 Guidance: Reporting of the detected conflict (or absence thereof) to the xApps by the platform contains the affected KPIs and, optionally, potentially conflicting RAN parameters of this xApp etc. NOTE: This information may then be used for subsequent conflict resolution and avoidance.

#### 5.2.4.3 Procedures

##### 5.2.4.3.1 Implementation 1 (Platform handles E2SM):

|  |  |
| --- | --- |
| **Use Case Step** | **Description of the Step** |
| Goal | ConMit functionality wants to resolve indirect and/or implicit conflicts at E2SM level among xApps |
| Actors and Roles | ConMit functionality  xApps  E2 Node |
| Assumptions | There are at least 2 xApps, 1 ConMit functionality, and 1 E2 Node available.  The ConMit functionality understand E2SM information and it is provisioned to handle such information. |
| Pre-conditions | ConMit functionality is authorized to resolve conflicts among xApps.  An E2AP level potential conflict is detected but has not been confirmed previously at E2SM level. xApps have been identified and informed about a potential conflict post-action. The necessary information regarding the E2 Node and the RAN Function is available. Additional optional side-information may also be available such as which RAN parameters may be involved in the conflict.  ConMit may request xApps for information which is required to detect the conflict post-action. |
| Begins when | Conflict detection trigger: Either an xApp requests the platform to detect a conflict with atleast one other xApp post action, or the platfrom’s ConMit functionality triggers the process. |
| Step 0 (O) | a) An xApp reports a potential performance related conflict to the ConMit functionality.  b) The platform informs the xApp if there is a possible conflict that it can investigate. |
| Step 1 (M) | The ConMit requests xApps for the conflict relevant information |
| Step 2 (M) | 1. The xApps prepare the conflict relevant information which contains the potentially affected KPIs and additional optional information such as degradation tolerance and RAN control parameters used in xApp’s RIC control or policy actions. 2. The xApps send conflict relevant information to ConMit functionality and request conflict detection based on this information. |
| Step 3 (M) | The ConMit functionality responds with an Acknowlegment of conflict detection request based on the provided information. |
| Step 4 (M) | The ConMit functionality constructs the conflict context |
| Step 5 (M) | Data analysis is performed by the ConMit functionality, and it is determined if the information is sufficient for conflict determination |
| Step 6 (O) | If step 5 results in a conflict determination, conflict report is sent to the xApps and the platform records the detected conflict for future use in conflict avoidance and resolution |
| Step 7 (M) | If data collection from RAN is required, the ConMit functionality subscribes to the E2 Node for information reporting in the conflict context |
| Step 8 (M) | The E2 Node does information reporting |
| Step 9 (M) | Data analysis is performed by the ConMit functionality |
| Step 10 (O) | Conflict report is sent to the xApps and ConMit records the detected conflict for future use in conflict avoidance and resolution |
| Ends when | A conflict or lack thereof is determined |
| Exceptions | N/A |
| Post conditions | N/A |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 4

skinparam lifelineStrategy solid

skinparam defaultFontSize 12

participant xApp1 as xApp1

participant xApp2 as xApp2

Participant plt as "Near-RT RIC platform"

participant e2node as “E2 Node”

plt -> plt: (Pre-action) Potential Conflict Detection

Alt (Pre-action) E2SM level Potential conflict detection

xApp1<->plt: Conflict Detection

xApp2<->plt: Conflict Detection

end

Alt (Post-action) If xApp detects oscillation due to a potential conflict

xApp1-->plt: 0a: (E2 Related API) E2 Guidance ((NEW) Conflict Check (Request))

plt-->xApp1: 0b: (E2 Related API) E2 Guidance ((NEW) Conflict Check (Response))

end

plt -> plt: (Post-action) Conflict Detection (Trigger)

plt -> xApp1: 1: (E2 related API) E2 Guidance (Modification, potential conflict information)

plt -> xApp2: 1: (E2 related API) E2 Guidance (Modification, potential conflict information)

xApp2 -> xApp2: 2a: (Modified) Prepare the conflict relevant information

xApp1 -> xApp1: 2a: (Modified) Prepare the conflict relevant information

xApp1 -> plt: 2b: (E2 related API) E2 Guidance (Request, conflict relevant information)

xApp2 -> plt: 2b: (E2 related API) E2 Guidance (Request, conflict relevant information)

plt -> xApp1: 3a: (E2 related API) E2 Guidance (Reponse, Acknowledge)

plt -> xApp2: 3a: (E2 related API) E2 Guidance (Response, Acknowledge)

plt -> plt: 4: (NEW) Compute conflict context

Alt If sufficient Information for detection

plt -> plt: 5: (Modified) Conflict Mitigation (Data analysis)

plt --> xApp1: 6: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> xApp2: 6: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> plt: Record conflict

else If data collection from RAN required

plt <-> e2node: 7: (E2) RIC Subscription (Report)

Alt While Subscription Active

e2node -> plt: 8: (E2) RIC Indication (Report)

plt -> plt: 9: (Modified) Conflict Mitigation (Data analysis)

Alt Detection Complete

plt --> xApp1: 10: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> xApp2: 10: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> plt: Record conflict

end

end

end

@enduml

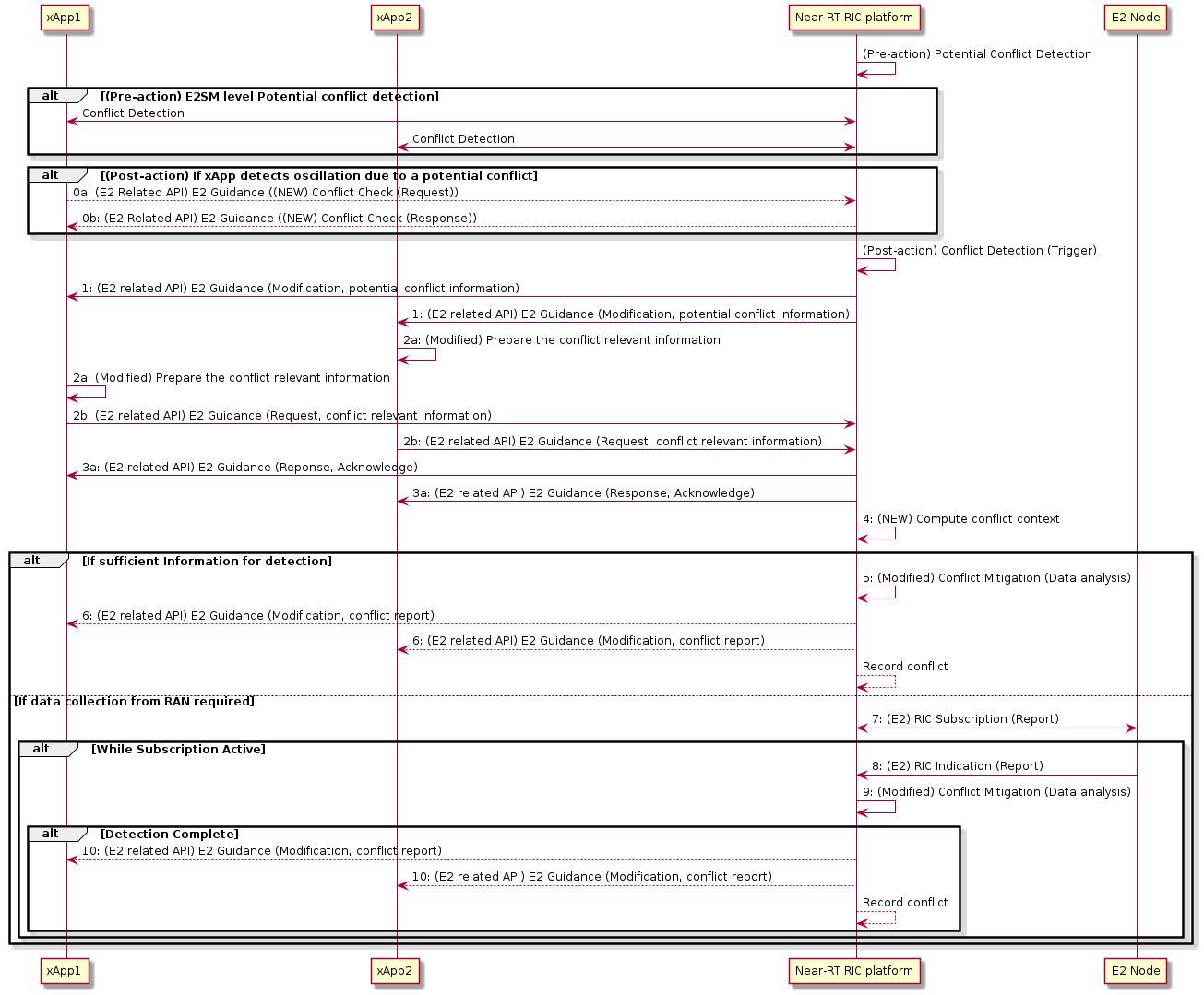


Figure 5.2-3: Implementation Option 1 (Platform Handles E2SM)

##### 5.2.4.3.2 Implementation 2 (Platform does not handle E2SM):

|  |  |
| --- | --- |
| **Use Case Step** | **Description of the Step** |
| Goal | ConMit functionality wants to resolve indirect and/or implicit conflicts at E2SM level among xApps |
| Actors and Roles | ConMit functionality  xApps  ConMit helper xApp |
| Assumptions | There are at least 2 xApps, 1 ConMit platform functionality, and 1 ConMit helper xApp.  The ConMit functionality does not understand E2SM information and/or it is not provisioned to handle such information. |
| Pre-conditions | ConMit functionality is authorized to resolve conflicts among xApps.  An E2AP level potential conflict is detected but has not been confirmed previously at E2SM level. xApps have been identified and informed about a potential conflict post-action. The necessary information regarding the E2 Node and the RAN Function is available. Additional optional side-information may also be available.  ConMit may request xApps for information which is required to detect the conflict post-action. |
| Begins when | Conflict detection trigger: Either an xApp requests the platform to detect a conflict with atleast one other xApp post action, or the platfrom’s ConMit functionality triggers the process. |
| Step 0 (O) | a) An xApp reports a potential performance related conflict to the ConMit functionality.  b) The platform informs the xApp if there is a possible conflict that it can investigate. |
| Step 1 (M) | The ConMit requests xApps for the conflict relevant information |
| Step 2 (M) | 1. The xApps prepare the conflict relevant information which contains the potentially affected KPIs and additional optional information such as degradation tolerance and RAN control parameters used in xApp’s RIC control or policy actions. 2. The xApps send conflict relevant information to ConMit functionality and request conflict detection based on this information. |
| Step 3 (M) | 1. The ConMit functionality responds with an Acknowlegment of conflict detection request based on the provided information. 2. The ConMit functionality, after obtaining information from the xApps, forwards this information to the helper xApp, provides associated E2AP level information, and requests a post-action conflict check |
| Step 4 (M) | The ConMit helper xApp contructs the conflict context. |
| Step 5 (M) | Data analysis is performed by the ConMit functionality, and it is determined if the information is sufficient for conflict determination |
| Step 6 (M) | If step 5 results in some determination, a conflict report is sent to the ConMit functionality by ConMit helper xApp. |
| Step 7 (0) | If step 5 results in some determination, conflict report is sent to the xApps and the platform records the detected conflict for future use in conflict avoidance and resolution. |
| Step 8 (M) | If data collection from RAN is required, ConMit helper xApp subscribes to the E2 Node for information reporting in the conflict context. |
| Step 9 (M) | The E2 Node does information reporting. |
| Step 10 (M) | Data analysis is performed by the ConMit helper xApp. |
| Step 11 (M) | Conflict report is sent to the ConMit functionality by ConMit helper xApp. |
| Step 12 (O) | Conflict report is sent to the xApps and ConMit records the detected conflict for future use in conflict avoidance and resolution |
| Ends when | A conflict or lack thereof is determined |
| Exceptions | N/A |
| Post conditions | N/A |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 4

skinparam lifelineStrategy solid

skinparam defaultFontSize 12

participant xApp1 as xApp1

participant xApp2 as xApp2

Participant plt as "Near-RT RIC platform"

Participant xApph as “Helper xApp”

participant e2node as “E2 Node”

plt -> plt: (Pre-action) Potential Conflict Detection

Alt (Pre-action) E2SM level Potential Conflict Detection

xApp1<->plt: Conflict Detection

xApp2<->plt: Conflict Detection

end

Alt (Post-action) If xApp detects oscillation due to a potential conflict

xApp1-->plt: 0a: (E2 Related API) E2 Guidance((NEW) Conflict Check (Request))

plt-->xApp1: 0b: (E2 Related API) E2 Guidance ((NEW) Conflict Check (Response))

end

plt -> plt: (Post-action) Conflict Detection (Trigger)

plt -> xApp1: 1: (E2 related API) E2 Guidance (Modification, potential conflict information)

plt -> xApp2: 1: (E2 related API) E2 Guidance (Modification, potential conflict information)

xApp2 -> xApp2: 2a: (Modified) Prepare the conflict relevant information

xApp1 -> xApp1: 2a: (Modified) Prepare the conflict relevant information

xApp1 -> plt: 2b: (E2 related API) E2 Guidance (Request, conflict relevant information)

xApp2 -> plt: 2b: (E2 related API) E2 Guidance (Request, conflict relevant information)

plt -> xApp1: 3a: (E2 related API) E2 Guidance (Reponse, Acknowledge)

plt -> xApp2: 3a: (E2 related API) E2 Guidance (Response, Acknowledge)

plt ->xApph: 3b: (NEW) conflict check (E2AP and conflict relevant informations)

xApph -> xApph: 4: (NEW) Compute conflict context

Alt If sufficient Information for detection

xApph -> xApph: 5: (Modified) Conflict Mitigation (Data analysis)

xApph -> plt: 6: (NEW) conflict check (conflict report)

plt --> xApp1: 7: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> xApp2: 7: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> plt: Record conflict

else If data collection from RAN required

plt <-> xApph: 8: (E2 related API) E2 Subscription (Report)

plt <-> e2node: 8: (E2) RIC Subscription (Report)

Alt While Subscription Active

e2node -> plt: 9: (E2) RIC Indication (Report)

plt -> xApph: 9: (E2 related API) E2 Indication (Report)

xApph -> xApph: 10: (NEW) Conflict Mitigation (Data analysis)

Alt Detection Complete

xApph -> plt: 11: (NEW) conflict check (conflict report)

plt --> xApp1: 12: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> xApp2: 12: (E2 related API) E2 Guidance (Modification, conflict report)

plt --> plt: Record conflict

end

end

end

@enduml

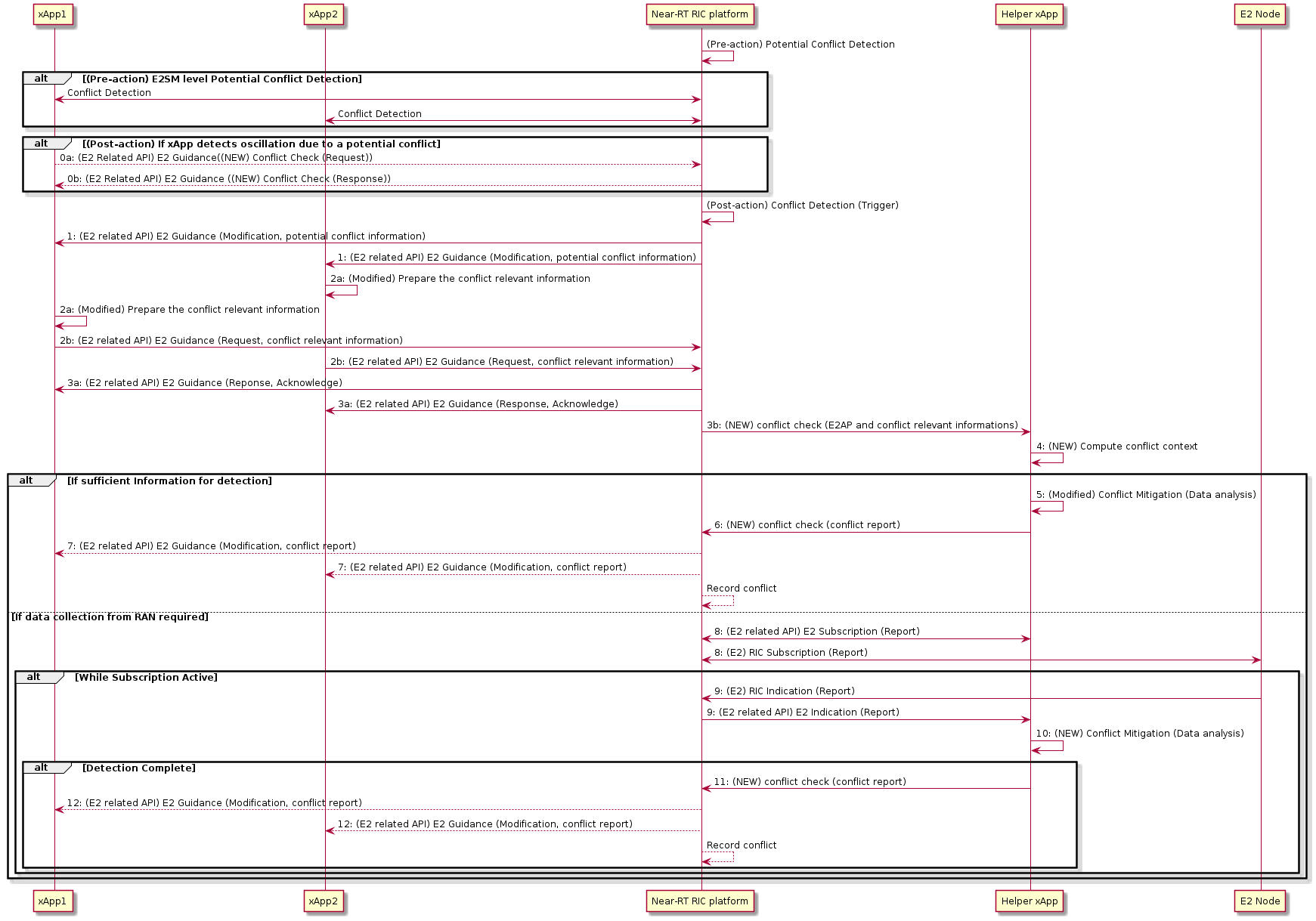


Figure 5.2-4: Implementation Option 2 (Helper xApp Handles E2SM)

#### 5.2.4.4 Evaluation

The solution addresses the considerations in clause 5.1.1.3 in the following ways:

* The solution requires both E2AP and E2SM level information.
* The solution assumes that the platform and xApps are aware of a previously detected potential conflict which was not confirmed (pre-action) using E2AP and E2SM information. So further post-action investigation is required to detect any indirect/implicit conflicts.
* The solution proposes to detect indirect/implicit conflicts post-action. It is assumed that these conflicts are either not detected pre-action or they only arise post-action for certain RAN states.
* Only “harmful” conflicts may be detected. This may be justified by the fact that many RAN KPIs are coupled with each other, and achieving an entirely conflict-free system may be impossible. In such cases, we can consider managing conflicts rather than removing them completely.
* Same solution works for both indirect and/or implicit conflicts.

The solution addresses the open issues in clause 5.1.1.4 in the following ways:

* The solution does not require direct xApp communication for conflict detection. The E2SM inspection is done by the platform or a helper xApp.
* If the platform is not provisioned to handle E2SM information, then a helper xApp is deployed for this purpose. The xApps are not required to expose their business-logic. For example, the xApps may not expose changes made to the RAN parameters. The KPIs they expose/provide for conflict monitoring are standardized KPIs.

The solution requires tools for intelligent analysis of E2AP and E2SM specific data (e.g., the required correlation analysis or AI/ML methods) to be either part of the Near-RT RIC platform functionality or be provided as part of the helper xApp.

### 5.2.5 Solution #5 Pre-Action Detection of E2-related Indirect Conflicts with user configured parameter dependency information

#### 5.2.5.1 Introduction

Key issue#3 for Conflict detection (clause 5.1.3) describes the possibility that indirect conflicts may be detected pre-action, i.e., before RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST messages have been sent to the E2 Node.

NOTE: As mentioned in clause 5.1.3.1, when xApps target different RAN control parameters, there can be conflicting impacts of their actions on the overall RAN state. This solution considers such scenarios as indirect conflicts, which do not need any KPIs to be monitored to detect the conflict.

The proposed solution (detailed in clauses 5.2.5.2 and 5.2.5.3) focuses on detecting the indirect conflicts pre-action (i.e. before RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST messages have been sent to the E2 nodes) based on the parameter dependency information provided by the user as an input to the ConMit functionality or a helper ConMit xApp.

This solution assumes that after E2AP level information {E2 Node Id, RAN Function ID, RIC Action Type} analysis the ConMit Functionality or ConMit helper xApp also performs E2SM level analysis during the RICARCH [i.1] defined xApp initiated E2 Subscription procedure ([i.1] clause 9.3.2.1), E2 Control procedure ([i.1] clause 9.3.2.4) and E2 Guidance request/response procedure ([i.1] clause 9.3.3.1).

NOTE: Since the platform may not be provisioned to handle E2SM information, the solution proposes two alternatives for E2SM processing. In implementation option 1, the ConMit functionality in the platform handles E2SM processing. In implementation option 2, a ConMit helper xApp handles E2SM processing and subscribes to the E2 Node to collect required information.

#### 5.2.5.2 Functional Description

The solution assumes the following prerequisites:

* Parameter dependency information is provided to the ConMit functionality or ConMit helper xApp as an input.
* This information can be in the form of a linear lookup table or a dependency graph or any other preferred format, which captures the relationship between the RAN parameters in RIC CONTROL/POLICY service and their probable impact on a set of KPIs (see clause 4.2.1). This information can be continuously improved as and when new conflicts are discovered.
* A human user can prepare this dependency information. Potentially the dependency information can be prepared and continuously improved by an AI/ML system.

Overall Description: Before the xApps are onboarded and put in operation, a user can analyze and predict the inter-dependencies among the RAN parameters changed by the xApps. The user provided parameter dependency information maps the RIC CONTROL/POLICY parameters to a common system impact. If the ConMit functionality or a helper ConMit xApp has inter-parameter dependencies and predicted system impacts for certain parameters, then it can detect indirect conflict after investigating E2SM and E2AP information from the xApp procedures, whenever these parameters are involved. As more conflicts between parameters are learned/discovered, the user can update the knowledge base.

ConMit functionality or a helper ConMit xApp uses this mapping to declare that RIC CONTROL/POLICY parameters changed by two or more xApps are in indirect conflict. Since the detection of potential conflict takes place before the RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST is applied to the E2 nodes, this is a pre-action detection. The solution can be extended to detecting any number of indirect co-existing conflicts across two or more xApps. For example, let us assume two xApps changing two different RAN parameters. xApp1 and xApp2 change RIC CONTROL/POLICY parameters C1 and C2 respectively. C1 and C2 are different parameters, hence there is no direct conflict. However, changes in C1 and C2 individually may lead to a common system impact resulting in indirect conflict. The user provided parameter dependency information maps the relationship between C1 and C2 to a common system impact. ConMit functionality or a helper ConMit xApp uses this mapping to declare that RIC CONTROL/POLICY parameters changed by xApp1 and xApp2 are in indirect conflict.

As such, the solution consists of the following steps:

* Before the xApps are put in operation, a user configures the parameter dependency information as input into the ConMit functionality or helper ConMit xApp. This dependency information is sent to the Near-RT RIC e.g. via O1 interface.
* xApps may send the changed parameters with E2AP and E2SM level details as part of E2 CONTROL REQUEST or E2 SUBSCRIPTION REQUEST or E2 GUIDANCE REQUEST to Near-RT RIC platform. These outputs do not need to be simultaneous to be considered as potential conflict.
* ConMit functionality or helper ConMit xApp checks the RAN parameters received from the xApps against the parameter dependency information. If the changed parameters by different xApps map to a common KPI impact, then the ConMit functionality or helper ConMit xApp declares it as a potential implicit conflict.
* The detected conflict (or absence thereof) is reported to the xApps by the Near-RT RIC platform. This report contains information on theimpacted system functions, optionally, potential conflicting RAN parameters of the conflicting xApps, etc.
* In addition to the pre-action conflict detection based on the user provided parameter dependency information, the ConMit functionality or helper ConMit xApp can further leverage post-action detection where certain KPIs can be monitored after the changes from xApps are applied to the E2 nodes, to further detect unacceptable/abnormal degradation and inter-dependence in xApp actions that are not detected in the pre-action conflict detection stage.

#### 5.2.5.3 Procedures

Two or more xApps send the E2 CONTROL REQUEST or E2 SUBSCRIPTION REQUEST or E2 GUIDANCE REQUEST to the Near-RT RIC platform. The ConMit functionality or the helper xApp decodes both the E2AP and E2SM level details in these messages. With the use of the user-provided parameter dependency information, if the ConMit functionality or the helper xApp map the changed parameters to a common system impact, then an indirect conflict is detected. This conflict detection report with relevant information is sent to the involved xApps.

#### 5.2.5.4 Evaluation

### 5.2.6 Solution #6 Pre-Action Detection of Potential E2-related Implicit Conflicts with user configured parameter dependency information

#### 5.2.6.1 Introduction

Key issue #1 for Conflict Detection (clause 5.1.1) describes the need for potential *pre-action* conflict detection by the Near-RT RIC platform for the specific case where the ConMit functionality in the Near-RT RIC platform may use both E2AP and E2SM level information.

NOTE: As mentioned in clause 5.1.3.1, when xApps target different RAN control parameters, there may be conflicting impacts of their actions in terms of KPIs. This solution considers such scenarios as implicit conflicts.

The proposed solution (detailed in clauses 5.2.6.2 and 5.2.6.3) focuses on detecting the potential implicit conflicts pre-action (i.e. before RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST messages have been sent to the E2 nodes) based on the parameter dependency information provided by the user as an input to the ConMit functionality or a helper ConMit xApp.

This solution assumes that after E2AP level information {E2 Node Id, RAN Function ID, RIC Action Type} analysis, the ConMit Functinoality or ConMit helper xApp also performs E2SM level analysis during the RICARCH [i.1] defined xApp initiated E2 Subscription procedure ([i.1] clause 9.3.2.1), E2 Control procedure ([i.1] clause 9.3.2.4) and E2 Guidance request/response procedure ([i.1] clause 9.3.3.1).

NOTE: Since the platform may not be provisioned to handle E2SM information, the solution proposes two alternatives for E2SM processing. In implementation option 1, the ConMit functionality in the platform handles E2SM processing. In implementation option 2, a ConMit helper xApp handles E2SM processing and subscribes to the E2 Node to collect required information.

#### 5.2.6.2 Functional Description

The solution assumes the following prerequisites:

- Parameter dependency information is provided to the ConMit functionality or ConMit helper xApp as an input.

- This information can be in the form of a linear lookup table or a dependency graph or any other preferred format, which captures the relationship between the RAN parameters in RIC CONTROL/POLICY service and their probable impact on a set of KPIs (see clause 4.2.1). This information can be continuously improved as and when new conflicts are discovered.

- A human user can prepare this dependency information. Potentially the dependency information can be prepared and continuously improved by an AI/ML system.

Overall Description: Before xApps are onboarded and put in operation, system/user can analyze and predict the inter-dependencies among the RAN parameters that may cause implicit KPI conflicts. The system/user provided parameter dependency information maps the RIC CONTROL/POLICY parameters to possible KPI impacts. If the ConMit functionality or a helper ConMit xApp has inter-parameter dependencies and possible KPI impacts for certain parameters, then it can detect potential implicit conflict after investigating E2SM and E2AP information from the xApp procedures, whenever these parameters are involved. As more conflicts between parameters are learned/discovered, the system/user can update the knowledge base.

ConMit functionality or a helper ConMit xApp uses this mapping to declare that RIC CONTROL/POLICY parameters changed by two or more xApps are in potential implicit conflict if the changes done by xApps map to common KPI impact. Since the detection of potential conflict takes place before the RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST is applied to the E2 nodes, this is a pre-action detection. The solution can be extended to detecting any number of indirect co-existing conflicts across two or more xApps.

For example, let us assume an xApp, xApp1, changes RIC CONTROL/POLICY RAN parameter C1. The parameter dependency information is provided to ConMit functionality or a helper ConMit xApp by mapping C1 to KPI1 & KPI2 meaning changes in C1 can possibly impact KPI1 & KPI2. Then, another xApp, xApp 2, change another RAN parameter C2 which is marked having impact to KPI1 and KPI2 as well. The ConMit functionality or a helper ConMit xApp uses this mapping to declares that RIC CONTROL/POLICY parameters changed by xApp1 and xApp2 are in potential implicit conflict. The potential pre-action conflict is declared, and post-action conflict detection will be triggered by either the Near-RT RIC framework or the involved xApps. This solution can be extended by mapping multiple RIC CONTROL/POLICY parameters to multiple sets of KPIs with possible weightage.

As such, the solution consists of the following steps:

- Before the xApps are put in operation, a user configures the parameter dependency information as input into the ConMit functionality or helper ConMit xApp. This dependency information is sent to the Near-RT RIC e.g. via O1 interface.

- xApps may send the changed parameters with E2AP and E2SM level details as part of E2 CONTROL REQUEST or E2 SUBSCRIPTION REQUEST or E2 GUIDANCE REQUEST to Near-RT RIC platform. These outputs do not need to be simultaneous to be considered as potential conflict.

- ConMit functionality or helper ConMit xApp checks the RAN parameters received from the xApps against the parameter dependency information. If the changed parameters by different xApps map to a common KPI impact, then the ConMit functionality or helper ConMit xApp declares it as a potential implicit conflict.

- The detected conflict (or absence thereof) is reported to the xApps by the Near-RT RIC platform. This report contains information on theimpacted system functions, optionally, potential conflicting RAN parameters of the conflicting xApps, etc.

- In addition to the pre-action conflict detection based on the user provided parameter dependency information, the ConMit functionality or helper ConMit xApp can further leverage post-action detection where certain KPIs can be monitored after the changes from xApps are applied to the E2 nodes, to further detect unacceptable/abnormal degradation and inter-dependence in xApp actions that are not detected in the pre-action conflict detection stage.

#### 5.2.6.3 Procedures

Two or more xApps send the E2 CONTROL REQUEST or E2 SUBSCRIPTION REQUEST or E2 GUIDANCE REQUEST to the Near-RT RIC platform. The ConMit functionality or the helper xApp decodes both the E2AP and E2SM level details in these messages. With the use of the user-provided parameter dependency information, if the ConMit functionality or the helper xApp map the changed parameters to a common KPI impact, then a potential implicit conflict is detected. This conflict detection report with relevant information is sent to the involved xApps.

#### 5.2.6.4 Evaluation

### 5.2.7 Solution #7 Pre-Action Detection of E2-related Implicit Conflicts with with user configured parameter dependency information

#### 5.2.7.1 Introduction

Key issue#3 for Conflict detection (clause 5.1.3) describes the possibility that implicit conflicts may be detected in pre-action also, i.e., before RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST messages have been sent to the E2 Node.

NOTE: As mentioned in clause 5.1.3.1, when xApps target different RAN control parameters, there may be conflicting impacts of their actions in terms of KPIs. This solution considers such scenarios as implicit conflict

The proposed solution (detailed in clauses 5.2.7.2 and 5.2.7.3) focuses on detecting the implicit conflicts pre-action including both potential conficts and the actual conficts. The solution is based on the paramter dependency information provided as an input to the the ConMit functionality or a helper ConMit xApp which help to detect potential conflict pre-action. If a potential conflict is detected, ConMit functionality or a helper ConMit xApp invoke another algorithm (e.g. AI/ML or digital twin based) to predict the impacts on KPIs by changes done in the RAN parameters based on learning from historical data. This prediction can be performed by ConMit functionality or a helper ConMit xApp itself or by another service in Near-RT RIC or another xApp. Both the potential conflict detection and actual conflict detection thourgh KPI impact prediction are done in pre-action stage before the RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST messages have been sent to the E2 Node.

NOTE: Since the platform may not be provisioned to handle E2SM information, the solution proposes two alternatives for E2SM processing. In implementation option 1, the ConMit functionality in the platform handles E2SM processing. In implementation option 2, a ConMit helper xApp handles E2SM processing and subscribes to the E2 Node to collect required information.

#### 5.2.7.2 Functional Description

The solution assumes the following prerequisites:

- Parameter dependency information is provided to the ConMit functionality or ConMit helper xApp as an input.

- This information can be in the form of a linear lookup table or a dependency graph or any other preferred format, which captures the relationship between the RAN parameters in RIC CONTROL/POLICY service and their predicted impact on a set of KPIs (see clause 4.2.1) This information can be continuously improved as and when new conflicts are discovered.

- A human user can prepare this dependency information. Potentially the dependency information can be prepared and continuously improved by an AI/ML system.

Overall Description: Before certain xApps are onboarded and put in operation, system/user can analyze and predict the inter-dependencies among the RAN parameters that may cause implicit KPI conflicts. The system/user provided parameter dependency information maps the RIC CONTROL/POLICY parameters to a common KPI impact. If the ConMit functionality or a helper ConMit xApp has inter-parameter dependencies and predicted KPI impacts for certain parameters, then it can detect potential implicit conflict after investigating E2SM and E2AP information from the xApp procedures, whenever these parameters are involved. As more conflicts between parameters are learned/discovered, the system/user can update the knowledge base.

The ConMit functionality or a helper ConMit xApp uses this mapping to declare that RIC CONTROL/POLICY parameters change by the xApps are in potential implicit conflict. After this pre-action potential implicit conflict detection, either the Near-RT RIC framework or the involved xApps invokes another algorithm to predict the impacts on KPIs by the changes done.

For example, let us assume the prediction algorithm predicts that the change in RIC CONTROL/POLICY parameter C1 from xApp1 will degrade KPI1 and improve KPI2, however the changes from xApp2 on parameter C2 will not affect KPI1 but degrade KPI2. The ConMit functionality or a helper ConMit xApp immediately declare it as implicit conflict. Since the detection of potential conflict takes place before the RIC CONTROL REQUEST or RIC SUBSCRIPTION REQUEST is applied to the E2 nodes, this is a pre-action detection. The solution can be extended to any number of xApps.

As such, the solution consists of the following steps:

- Before the xApps are put in operation, a user configures the parameter dependency information as input into the ConMit functionality or helper ConMit xApp. This dependency information is sent to the Near-RT RIC e.g. via O1 interface.

- xApps may send the changed parameters with E2AP and E2SM level details as part of E2 CONTROL REQUEST or E2 SUBSCRIPTION REQUEST or E2 GUIDANCE REQUEST to Near-RT RIC platform. These outputs do not need to be simultaneous to be considered as potential conflict.

- ConMit functionality or helper ConMit xApp checks the RAN parameters received from the xApps against the parameter dependency information. If the changed parameters by different xApps map to the common KPI impacts, then the ConMit functionality or helper ConMit xApp declares it as potential implicit conflict.

- Another algorithm or AIML model is invoked bythe ConMit functionality or helper ConMit xApp to predict the exact KPI impacts by the changed parameters.

- If a KPI impact conflict is detected through the prediction, the ConMit functionality or helper ConMit xApp report the detected conflict (or absence thereof) to the xApps. This report contains information on theimpacted system functions, optionally, potential conflicting RAN parameters of the conflicting xApps, etc.

NOTE: This information may then be used for subsequent conflict resolution and avoidance.

#### 5.2.7.3 Procedures

Two or more xApps send the E2 CONTROL REQUEST or E2 SUBSCRIPTION REQUEST or E2 GUIDANCE REQUEST to the Near-RT RIC platform. The ConMit functionality or the helper xApp decodes both the E2AP and E2SM level details in these messages. With the use of the user-provided parameter dependency information, if the ConMit functionality or the helper xApp map the changed parameters to a common KPI impact, then a potential implicit conflict is detected. Furthermore, with the use of another algorithm, if the ConMit functionality and the helper xApp predict the exact KPI impact, then actual implicit conflict is detected. This conflict detection report with relevant information is sent to the involved xApps.

#### 5.2.7.4 Evaluation

## 5.3 Solutions-to-Key Issues Mapping

Table 5.3-: Solutions to Key Issue Mapping

|  | Key Issue | | |
| --- | --- | --- | --- |
| **Solution** | #1 | #2 | #3 |
| #1 | yes |  |  |
| #2 |  | yes |  |
| #3 |  | yes |  |
| #4 |  |  | yes |
| #5 |  |  | yes |
| #6 | yes |  |  |
| #7 |  |  | yes |

# 6 Conflict Resolution

## 6.1 Key Issues

### 6.1.1 Key Issue #1 Resolution of E2-related Conflicts

#### 6.1.1.1 Description

This key issue addresses the problem of E2-related conflict resolution between xApps under the assumption that direct, indirect, or implicit conflicts have been detected and the conflicting xApps have been identified.

The resolutions may comprise the following:

1. xApps may reach a “resolution agreement” themselves directly or indirectly facilitated by conflict mitigation (ConMit) functionality in the platform.
2. ConMit may compute E2AP or E2SM level agreement for the xApps. This agreement is later enforced by ConMit.

#### 6.1.1.2 Impacted Use-Cases

void

#### 6.1.1.3 Considerations

Solutions for E2-related conflict resolution may consider the following:

1. Resolution may take place both pre-action and post-action depending on when the conflict is detected.
2. A general solution may work for all conflict types or different solutions may be provided for different conflict types. For example, a solution may address direct conflicts but not indirect and/or implicit conflicts.
3. E2AP level agreements: For example, xApps may agree not to use certain RIC service types (control, policy etc.) for the same E2 Node and/or RAN Function.
4. E2SM level agreements: For example, xApps may agree not to access certain RAN parameter(s) or a similar agreement may be reached in case of RAN resource conflicts.
5. E2AP level guidance agreement: xApps may agree to access an E2 Node and/or RAN Function only after guidance by the platform.
6. E2SM level guidance agreement: xApps may agree to access certain RAN parameters or RAN resources only after guidance by the platform.
7. E2SM level negotiation: xApps may negotiate and reach an agreement on certian RAN Parameters, RAN KPIs, and/or RAN Resources. This may allow them to “manage” their conflicts.
8. Agreements may be recorded/saved for future conflict avoidance and/or during a new conflict resolution.

#### 6.1.1.4 Open issues

Following open issues are relevant:

1. ConMit may facilitate inter-xApp resolution, but it may not be able to resolve conflicts by itself at E2SM level without xApp-specific E2SM information.
2. Furthremore, xApps can, in principle, resolve their conflicts but this additional functionality must be part of their design.
3. Can ConMit force conflict resolution and an agreement, e.g., by occasionally denying E2 Subscription or E2 Control requests by xApps?
4. How to deal with failure of conflict resolution process between xApps?
5. Privacy: How to protect xApps’ business logic during resolution?

## 6.2 Solutions

### 6.2.1 Solution #1 Resolution of Direct Parameter Conflicts between xApps at E2SM level

#### 6.2.1.1 Introduction

A direct parameter conflict may occur between two or more xApps if they request access to the same set of RAN parameters in the same E2 Node. Some consideration for solutions to this type of conflicts are mentioned in clause 6.1.2.3.

As mentioned in clause 6.1.2.3, one possible resolution may be to prioritize one xApp’s E2 Subscription among all others so that only the xApp that has the priority accesses the RAN function/service and changes the parameter. Note that, such a resolution requires an E2SM level agreement between the xApps. However, such resolutions may not always be possible, e.g., if the objectives of an xApp that is not able to make RAN parameter changes are compromised beyond an acceptable level. Also, this type of approach may lead to sub-optimal performance of the network.

Another approach may be for xApps to directly negotiate and reach an E2SM level agreement over the values of a certain RAN parameter via inter-xApp communication. However, in a multi-vendor environment, when different xApps may be produced by different vendors, xApps may not reach such an agreement. As mentioned in clause 6.1.2.4, such direct inter-xApp functionality may not be supported for all xApps. Furthermore, for a large number of xApps, this may not be technically feasible within the Near-RT RIC control time-granularity.

So, a solution is necessary to resolve direct parameter conflicts among xApps at E2SM level without requiring direct inter-xApp communication for negotiation. The proposed solution is “indirect” in the sense that the conflict mitigation functionality in the platform proposes a compromised/common value of the RAN parameter to the xApps (e.g., within an E2 Guidance Modification message [i.1]). The proposed solution, which can then be configured in the E2 Node, represents a value of the RAN parameter serving the common interests of the xApps. Such common value may be calculated in various ways as shown in the examples in clause 6.2.1.2.

NOTE: The common value may be computed by the platform itself if it is capable of handling E2SM information. However, to address open issues 1 and 4 in key issue#1 clause 6.1.1.4, an option is to avoid processing E2SM specific information in the platform ConMit functionality directly. Instead, a “helper” ConMit xApp may be used for this purpose. The conflicting xApps send E2SM information to the platform which forwards the same to the helper xApp over the Near-RT RIC internal APIs. It is assumed that the helper xApp understands all E2SM specific information, but conflicting xApps’ business logic is not exposed to the helper xApp. After calculation of the common parameter, the helper xApp sends the results back to the platform which forwards them to the xApps. The same helper xApp may also be used to configure the common RAN parameter in the RAN on behalf of the xApps. Whether such an xApp and the required APIs are technically feasible is an important consideration. The same applies to privacy and security issues related to E2SM information exchanges.

#### 6.2.1.2 Functional Description

The solution assumes the following:

1. A direct RAN parameter conflict has already been detected pre-action or post-action involving a particular RAN parameter and the conflicting xApps have already been identified.
2. An E2AP level resolution/agreement is not appropriate.
3. An E2SM level resolution has been triggered either by xApp(s) or by conflict mitigation inside the platform.
4. Near-RT RIC may configure the RAN parameter in the E2 Node after successful resolution.

The xApps may communicate to the platform their preference regarding the RAN parameter. This preference is used to compute a common value for that parameter taking interests of all the xApps into account. Below we explain the solution with an example of a conflict between two xApps. The idea can be extended to any number of xApps. Below we provide two examples showing how the common value may be computed.

##### 6.2.1.2.1 Example 1: Common Value Calculation Method 1

In this method, the xApps send their preferred values of the parameter and the common value is calculated by averaging those values. For example, let us assume that xApp1 and xApp2 want to access and change the same RAN parameter *p* in the same E2 node. xApp1 wants to set the value at *p1* while xApp2 wants to set the value at *p2*. After both send this value, the common value is calculated as the average *p = pc = (p1 + p2)/2.* This solution is simple and fast, but not optimal.

##### 6.2.1.2.2 Example 2: Preferred Common Value Calculation Method 2

In this method, instead of sending values of the parameter, the xApps send utility functions. A utility function maps the “goodness” of a parameter value to a predefined scale, e.g., [0:1] scale. So, when different xApps send their utility functions, the common resolved value of the parameter is the one for which the product of the utility functions is maximized. This is called *Nash Social Welfare Solution*. Note that the utility functions do not expose the business-logic of the xApps but rather their “interests”.

#### 6.2.1.3 Procedures

##### 6.2.1.3.1 Implementation 1 (Platform handles E2SM):

|  |  |
| --- | --- |
| **Use Case Step** | **Description of the Step** |
| Goal | ConMit functionality wants to resolve direct parameter conflicts at E2SM level among xApps |
| Actors and Roles | ConMit functionality  xApps |
| Assumptions | There are at least 2 xApps and 1 ConMit functionality available. There is at least 1 conflicting RAN parameter.  The ConMit functionality understands E2SM specific information, or it uses a helper xApp to process such an information. |
| Pre-conditions | ConMit functionality is authorized to resolve conflicts among xApps.  ConMit may request xApps for information which is necessary to resolve the conflict. |
| Begins when | Conflict detection: A direct RAN parameter has been identified and the xApps have been identified to take part in conflict resolution at the E2SM level. |
| Step 1 (M) | The ConMit requests xApps to send their preference over the parameter. |
| Step 2 (M) | 1. The xApps computer their preferences 2. The xApps send preferences to ConMit functionality. |
| Step 3 (M) | 1. The ConMit functionality calculates the final common value of the parameter. 2. The ConMit functionality sends the computed value to the xApps. |
| Ends when | ConMit records the agreement |
| Exceptions | N/A |
| Post conditions | N/A |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

box “Near-RT RIC”

participant xApp1 as xApp1

participant xApp2 as xApp2

Participant plt as "Near-RT RIC platform"

end box

participant e2node as “E2 Node”

plt -> plt: Conflict Detection

plt -> plt: Conflict Resolution Pre-Processing

plt -> xApp1: 1: (E2 related API) E2 Guidance (Modification, parameter name)

plt -> xApp2: 1: (E2 related API) E2 Guidance (Modification, parameter name)

xApp2 -> xApp2: 2a: (Modified) Process the preference

xApp1 -> xApp1: 2a: (Modified) Process the preference

xApp2 -> plt: 2b: (E2 related API) E2 Guidance (Request, preference)

xApp2 -> plt: 2b: (E2 related API) E2 Guidance (Request, preference)

plt -> plt: 3a: (Modified) Compute common value

plt -> xApp1: 3b: (E2 related API) E2 Guidance (Response, common value)

plt -> xApp2: 3b: (E2 related API) E2 Guidance (Response, common value)

plt --> plt: Record agreement

@enduml

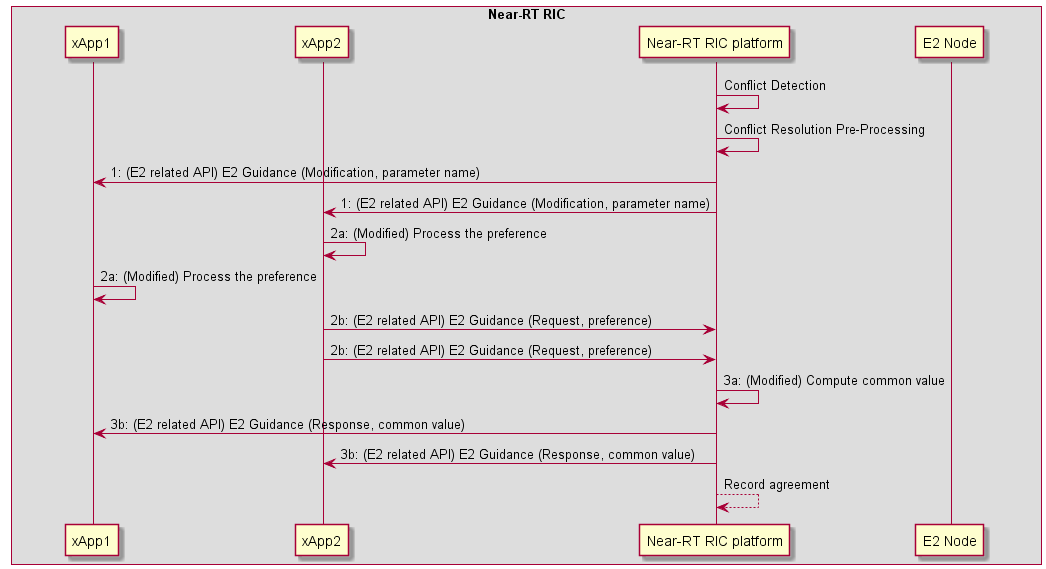


Figure 6.2-1: Implementation Option 1 (Platform Handles E2SM)

##### 6.2.1.3.2 Implementation 2 (Helper xApp handles E2SM):

|  |  |
| --- | --- |
| **Use Case Step** | **Description of the Step** |
| Goal | ConMit functionality wants to resolve direct parameter conflicts at E2SM level among xApps |
| Actors and Roles | ConMit functionality  xApps  helper xApp |
| Assumptions | There are at least 2 xApps, 1 ConMit functionality, and 1 helper xApp available. There is at least 1 conflicting RAN parameter.  The ConMit functionality does not understand/handle E2SM specific information. It uses a helper xApp to process E2SM. |
| Pre-conditions | ConMit functionality is authorized to resolve conflicts among xApps.  The helper xApp is authorized to process E2SM information and compute common value (agreement) of the RAN parameter.  ConMit may request xApps for information which is necessary to resolve the conflict. |
| Begins when | Conflict detection and resolution pre-processing: A direct RAN parameter has been identified and the xApps have been identified to take part in conflict resolution at the E2SM level. |
| Step 1 (M) | The ConMit functionality requests xApps to send their preference regarding the parameter. |
| Step 2 (M) | 1. The xApps compute their preference (either parameter value or the utility function) 2. The xApps send these values to ConMit functionality. |
| Step 3 (M) | 1. The ConMit functionality forwards the preference to the helper xApp 2. The helper xApp computes the common value (i.e., the agreement) 3. The helper xApp sends ConMit the common value |
| Step 4 (M) | ConMit forwards the common value to the xApps |
| Ends when | ConMit records the agreement |
| Post conditions | N/A |

@startuml

skin rose

skinparam ParticipantPadding 30

skinparam BoxPadding 10

skinparam lifelineStrategy solid

box “Near-RT RIC”

participant xApp1 as xApp1

participant xApp2 as xApp2

Participant plt as "Near-RT RIC platform"

Participant xAppH as “Helper xApp”

end box

participant e2node as “E2 Node”

plt <-> xAppH: (NEW) Conflict Detection (Identify RAN parameter)

plt -> plt: (NEW) Conflict Resolution Pre-processing (Identify xApps)

plt -> xApp1: 1: (E2 related API) E2 Guidance (Modification, {parameter name})

plt -> xApp2: 1: (E2 related API) E2 Guidance (Modification, {parameter name})

xApp2 -> xApp2: 2a: (Modified) Process the preference

xApp1 -> xApp1: 2a: (Modified) Process the preference

xApp2 -> plt: 2b: (E2 related API) E2 Guidance (Request, {preference})

xApp2 -> plt: 2b: (E2 related API) E2 Guidance (Request, {preference})

plt -> xAppH: 3a: (NEW) (Near-RT RIC API) Forward {preference}

xAppH -> xAppH: 3b: (NEW) Compute common value

xAppH -> plt: 3c: (NEW) (Near-RT RIC API) {common value}

plt -> xApp1: 4: (E2 related API) E2 Guidance (Response, {common value})

plt -> xApp2: 4: (E2 related API) E2 Guidance (Response, {common value})

plt --> plt: Record agreement

@enduml

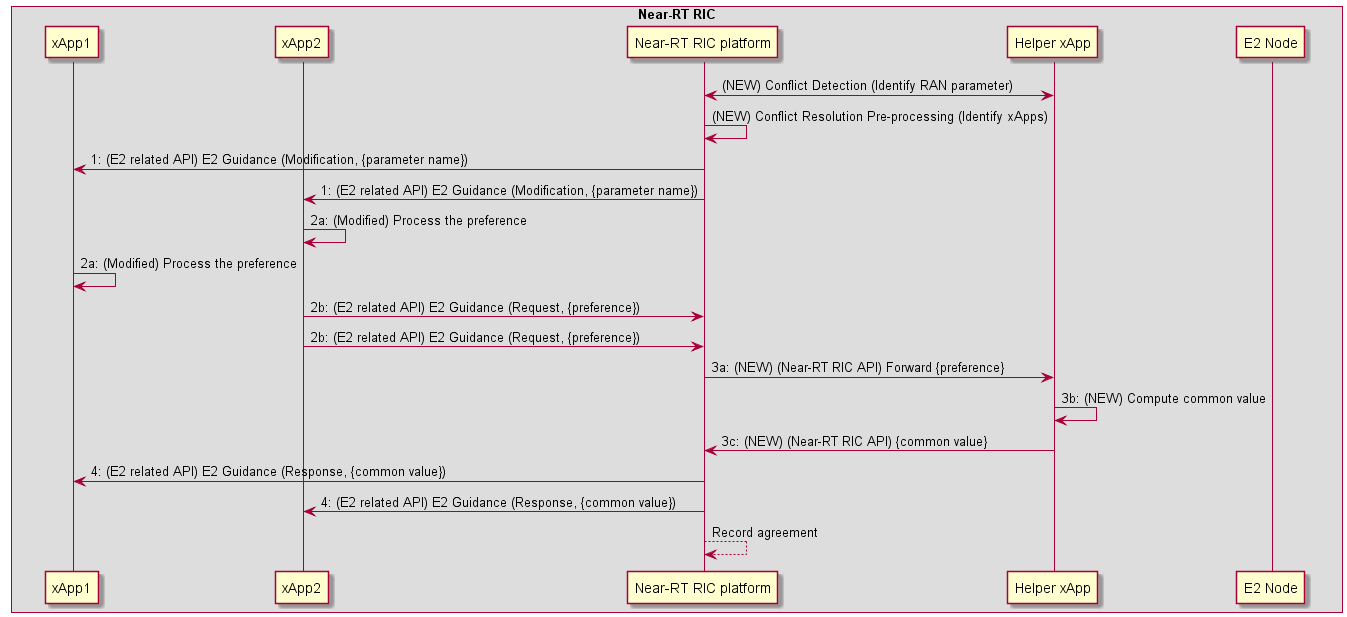


Figure 6.2-2: Implementation Option 2 (Helper xApp Handles E2SM)

#### 6.2.1.4 Evaluation

The solution requires an enabling detection solution that can identify the RAN parameters and the conflicting xApps. The solution also requires an enabling conflict avoidance solution, i.e., a solution addressing the eventual configuration of the common parameter *p* in the E2 Node and maintenance/enforcement of the resolution agreement (including failure cases). These mechanisms may be enabled by solutions in clauses 5.2 and 7.2 of the TR. To address open issues 1 and 4 in key issue#1 clause 6.1.1.4, general solutions may be added in conflict avoidance (clause 7.2) key issues.

The solution addresses the following consideration and open issues in clause 6.1.1:

1. Consideration 1 (in clause 6.1.1.3) is addressed. The resolution may take place both post-action (to avoid further conflict) and pre-action (to avoid future conflict).
2. Consideration 7 (in clause 6.1.1.3) is addressed. xApps manage their conflicts by agreeing to a common value.
3. Open issue 1 and 4 (in clause 6.1.1.4) are partly addressed by implementation 2, although the technical feasibility of the helper xApp may be studied further. Note that, the utility functions in Method 2 do not expose the business-logic of the xApps but rather only their “interests” on a [0 1] scale.
4. Example Method 2 is good to address the combined interests of the xApps because it ensures that all xApps are treated equally after considering their preferences. However, the common value may also be computed by taking into account other factors (in addition to xApp preferences) such as overall network performance.

## 6.3 Solutions-to-Key Issues Mapping

Table 6.3-: Solutions to Key Issue Mapping

|  | Key Issue | | |
| --- | --- | --- | --- |
| **Solution** | #1 | #Y | #Z |
| #1 | yes | - | - |
| #Y | - | - | - |
| #Z | - | - | - |

# 7 Conflict Avoidance

## 7.1 Key Issues

### 7.1.1 Key Issue #1 Avoidance of E2-related Conflicts

#### 7.1.1.1 Description

This key issue addresses the problem of E2-related conflict avoidance between xApps. Conflict avoidance takes place *pre-action*, i.e., before RIC Control Request or RIC Suscription Request messages have been sent to the E2 Node. Note that, conflict detection and resolution methods/solutions may be seen as part of the overall conflict avoidance goal.

Currently basic conflict avoidance functionality is enabled by a set of “E2 Guidance” APIs, while conflict detection and resolution may involve methods and processing required to complete this functionality. In more detail, the Near-RT RIC ARCH specification [i.1] already provides the following functionalities:

* Conflict avoidance may be enabled by the ConMit functionality in the platform as an “E2 Guidance” service. This service consists of the following:
* E2 Guidance Request: An xApp may request guidance from the ConMit functionality to avoid potential E2-related conflicts.
* E2 Guidance Response: The Near-RT RIC platform may provide E2 Guidance towards an xApp as a response to an E2 Guidance Request. The platform performs “conflict mitigation” related processing before providing Guidance.
* E2 Guidance Modification: An E2 Guidance request by an xApp may trigger E2 Guidance from the platform towards other potentially conflicting xApps.
* E2 Guidance Modification: A message from an xApp (e.g., E2 Subscription/Control Request), a message from an external interface, or a message from Near-RT RIC platform’s internal processes may trigger E2 Guidance towards xApps.

This key issue addresses further (possibly general) requirements and improvements for E2 Guidance mechanism in addition to solutions for conflict detection and resolution in clauses 5.2 and 6.2, respectively. Following are examples of aspects that may be addressed:

1. Improvements in requirements for E2 Guidance. What may the API messages contain, and do we need to extend these APIs?
2. How to avoid excessive and unnecessary use of E2 Guidance by xApps?
3. How to handle failure cases of conflict resolution?
4. How to ensure xApp security and privacy during E2 Guidance procedures?
5. The use of E2 Guidance for avoiding previously detected and resolved conflicts. How such agreements should be maintained or enforced for conflict avoidance?
6. What kind of external or internal events and messages may lead to triggering of E2 Guidance (or conflict avoidance) in Near-RT RIC?

#### 7.1.1.2 Impacted Use-Cases

Void

#### 7.1.1.3 Considerations

Solutions for conflict avoidance may consider the following:

1. A solution may enhance the above-mentioned “E2 Guidance” service but may not provide any solutions or methods for conflict detection and/or resolution tasks. Enhancements may support or refer to solutions for detection and resolution in clauses 5.2 and 6.2.
2. An xApp may use E2 Guidance (i.e., trigger conflict avoidance) to request if the platform detects a potential conflict. This request may contain E2AP and/or E2SM specific information.
3. ConMit functionality may trigger E2 Guidance (e.g., by issuing E2 Guidance Modification) if it detects a potential conflict between xApps. This guidance may be at E2AP and/or E2SM level.
4. Resolved Conflicts: Conflict avoidance may consider previously detected and resolved conflicts.
5. xApps may consider existing conflict resolution “agreements” before requesting guidance from the ConMit functionality in the platform.
6. Upon receiving a guidance request, ConMit functionality may consider previously detected and/or resolved conflicts between xApps before providing guidance to xApps for conflict avoidance.
7. New Conflicts: A guidance request by an xApp may trigger conflict detection and/or resolution by the ConMit functionality as part of the above-mentioned “conflict mitigation” processing.

#### 7.1.1.4 Open issues

Following open issues are relevant:

1. ConMit functionality may facilitate conflict avoidance, but it may not be able to provide required guidance at E2SM level without xApp-specific E2SM information.
2. Furthremore, xApps can, in principle, avoid conflicts but this functionality may need to be part of their design.
3. May the platform enforce an existing conflict resolution agreement, e.g., by occasionally denying E2 Subscription or E2 Control requests by xApps?
4. May conflict avoidance involve inter-xApp communication?
5. Privacy: How to protect xApps’ business logic during conflict avoidance?

## 7.2 Solutions

### 7.2.1 Solution #1 xApp based E2SM level service conflict avoidance

#### 7.2.1.1 Introduction

Solution #3 for Conflict Detection (clause 5.2.3) provides a mechanism based on Near-RT RIC platform detecting potential direct conflicts between 2 or more xApps using E2AP level information and then using E2 Guidance RICAPI to request identified xApp to perform E2SM level analysis.

Key Issue #1 for Conflict Resolution (clause 6.1.1) describes the possibility that “xApps may reach a “resolution agreement” themselves directly or indirectly facilitated by conflict mitigation (ConMit) functionality in the platform”.

Key Issue #1 for Conflict Avoidance (clause 7.1.1) describes the role of avoidance as “*pre-action*, i.e., before RIC Control Request or RIC Suscription Request messages have been sent to the E2 Node” and assumes the possibility that “ConMit functionality may facilitate conflict avoidance, but it may not be able to provide required guidance at E2SM level without xApp-specific E2SM information”, “May conflict avoidance involve inter-xApp communication?” and “How to protect xApps’ business logic during conflict avoidance?”.

This solution assumes that a list of xApps has been identified that are in potential direct conflict with each other and that they may use either inter-xApp message exchange or using the Conflict Mitigation functionality in the platform to route messages between xApps to negotiate a mutually acceptable partition of the E2 Node declared supported E2SM level services. The objective for this negotiation is to agree on which xApps may use which E2SM level service without risk of future direct conflicts.

At an E2SM level, direct conflicts are essentially due to 2 or more xApps attempting to use exactly the same E2SM specific RIC service.

For example:

- A POLICY service using E2SM-RC on a given E2 node. An E2AP level potential direct conflict would exist in this case if different xApps are requesting subscriptions with the same E2AP level information (same E2 Node, same RAN Function, same RIC Action Type) but with different Styles, Actions and/or RAN Parameters. If the xApps could reach an agreement on which E2SM level service are to be used by which xApp then E2SM level conflicts would be avoided.

#### 7.2.1.2 Functional Description

Solution #1 for Conflict Detection (clause 5.2.1) describes how the Near-RT RIC platform ([i.1] clause 6.2.3) would use E2AP level information from a requesting xApp and compare this with similar information obtained from other xApps from previous E2 Subscription, E2 Control and/or E2 Guidance requests. The result would be a “E2AP level” potential conflict detection.

Solution #2 and #3 for Conflict Detection describes how E2SM level information may be used to reduce the list of identified xApps in potential conflict using the Near-RT RIC platform (clause 5.2.2) and xApps (clause 5.2.3) respectively.

In this solution the Conflict Mitigation functionality in the Near-RT RIC platform may request the set of identified xApps to initiate a Conflict Avoidance negotiation. The proposed avoidance process would focus on negotiation between xApps to determine a mutually exclusive subset of the E2 Node supported RAN Function services.

Once the negotiation process terminates the xApps would respond to the Conflict Mitigation functionality in the platform to indicate success or failure. The platform may save the oucome of the negotiation and use it for future reference to Conflict Mitigation.

The platform may also issue a unique “Negotiation Key” to xApps which may then be used in future requests (E2 Guidance, E2 Subscription, E2 Control) to avoid un-necessary repetition of successive conflict detection and conflict avoidance events.

Finally, the platform may also request an xApp that was party to a previous negotiation to validate whether or not another xApp is acting in compliance with the agreement.

#### 7.2.1.3 Procedures

For the E2SM Level conflict avoidance process, RICARCH [i.1] clause 9.3.3.1 (E2 guidance request/response procedure) step 3 “Near-RT RIC platform processes request” and clause 9.3.3.5 (E2 Guidance modification procedure) step 1 “Near-RT RIC platform identifies affected xApps and performs conflict mitigation processing” may be used to perform both conflict detection and conflict avoidance. These procedures would need to be modified so that the Near-RT RIC platform may seek E2SM level analysis and avoidance negotiation from 2 or more xApps.

The following extensions are proposed to enable E2SM level conflict mitigation processing in xApps:

- E2 Guidance request/response and E2 Guidance Modification procedures extended:

- Near-RT RIC platform performs E2AP level potential conflict detection using solution proposed in clause 5.2.1 to identify one or more xApps that may be in conflict with requesting xApp

- Identified xApps requested to negotiate to find a suitable solution to avoid conflicts (i.e. to agree on a mutually exclusive subset of E2SM level defined RIC services)

- All xApps respond to Near-RT RIC platform with outcome of negotiation

- xApps may obtain a unique “Negotiation Key” that may be used in subsequent E2 related procedures

- E2 Related procedures extended to support xApps using assigned “Negotiation Key” in request messages to claim successful negotiation has already been performed

#### RICARCH 9.3.3.1 E2 Guidance request/response procedure

The purpose of the xApp initiated E2 Guidance request/response API procedure in the Near-RT RIC is to allow authorized xApps to obtain guidance from the Near-RT RIC platform (with the conflict mitigation functionality) prior to initiating an action.

Guidance from Near-RT RIC platform may include:

- Indication on whether or not the xApp proposed E2 Related API message or series of messages may result in a conflict with E2 related API messages from other xApps;

- Recommendations on how the proposed E2 Related message or series of messages should be modified to avoid conflict;

- Modification of previous guidance to other xApps and/or Near-RT RIC platform internal processes.

This procedure is based on the following assumptions:

- xApp may use E2 Related API to obtain guidance from Near-RT RIC platform to resolve potential conflicts prior to initiating a RIC function procedure;

- (NEW)Near-RT RIC platform may initiate a conflict mitigation avoidance negotiation process between two or more xApps prior to responding to xApp;

- xApp may use Near-RT RIC platform response in a subsequent procedure (i.e., for a RIC Functional Procedure).

This procedure is initiated by an xApp using **E2 Related API: E2 Guidance** request. The following outcomes are considered:

- Near-RT RIC Platform provides guidance to requesting xApp using E2 Related API: E2 Guidance response;

- Near-RT RIC Platform provides modified guidance to another xApp and/or its internal processes.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | xApp initiation of Conflict Mitigation guidance. |
| Actors and Roles | * xApp: Originator of Conflict Mitigation guidance request; * (NEW)Other xApp(s): One or more other xApps identified by Near-RT RIC platform as being in potential conflict with originator xApp * Near-RT RIC platform, with the following functionalities: * Database; * Conflict mitigation. |
| Assumptions | * Near-RT RIC platform has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution; * Near-RT RIC platform may initiate guidance towards other xApp as an optional addition response to Guidance Request. |
| Pre-conditions | * xApp has been authorized to request guidance from Near-RT RIC platform for conflict mitigation; * xApp has been assigned xApp ID. |
| Begins when | xApp determines need to request guidance from Near-RT RIC platform for conflict mitigation. |
| Step 1 (M) | xApp sends **E2 Related API E2 Guidance** Request to Near-RT RIC platform. |
| Step 2 (M) | Near-RT RIC platform collects related information. |
| Step 3A (M) | (MODIFIED) Near-RT RIC platform performs conflict mitigation detection |
|  | [ALT] Remaining step 3 events are executed if Near-RT RIC platform detects a potential conflict |
| Step 3B (O) | (NEW) Near-RT RIC platform requests identified xApps to perform conflict mitigation avoidance |
| Step 3C (O) | (NEW) Identified xApps uses either direct or indirect messaging to negotiate to obtain a mutually acceptable conflict avoidance agreement |
| Step 3D (O) | (NEW) Identified xApps respond to Near-RT RIC platform with outcome of negotiation |
| Step 3E (O) | (NEW) Near-RT RIC platform validates negotiated agreement and stores outcome. |
| Step 4-5 (O) | Near-RT RIC platform may signal conflict and/or provide guidance to internal processes and/or other xApps (see clause 9.3.3.5). (NEW)Message may contain a “negotiation key” that may be used in future requests from xApp |
| Step 6 (M) | Near-RT RIC platform sends **E2 Related API E2 Guidance** response to xApp. (NEW)Message may contain a “negotiation key” that may be used in future requests from xApp |
| Ends with | xApp continues processing using guidance response. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

box “Near-RT RIC”

Collections “Other xApp” as xApp

participant “xApp” as xApp2

Participant plt as "Near-RT RIC platform"

end box

xApp2 -> plt: 1: (E2 related API) E2 Guidance (Request)

plt -> plt: 2: Collect related information

plt -> plt: 3a: (MODIFIED)Conflict mitigation detection

Alt If platform has identified a potential conflict

plt-->xApp2: 3b. (NEW)Conflict mitigation avoidance request

plt-->xApp: 3b. (NEW)Conflict mitigation avoidance request

xApp<-->xApp2: 3c (NEW)Conflict avoidance negotiation

xApp2-->plt: 3d. (NEW)Conflict mitigation avoidance response

xApp-->plt: 3d. (NEW)Conflict mitigation avoidance response

plt -> plt: 3e. (NEW)Conflict mitigation processing

end

plt --> plt: 4: Guidance to internal processes

xApp <-- plt: 5: (E2 related API) E2 Guidance (Modification)

plt -> xApp2: 6: (E2 related API) E2 Guidance (Response)

@enduml

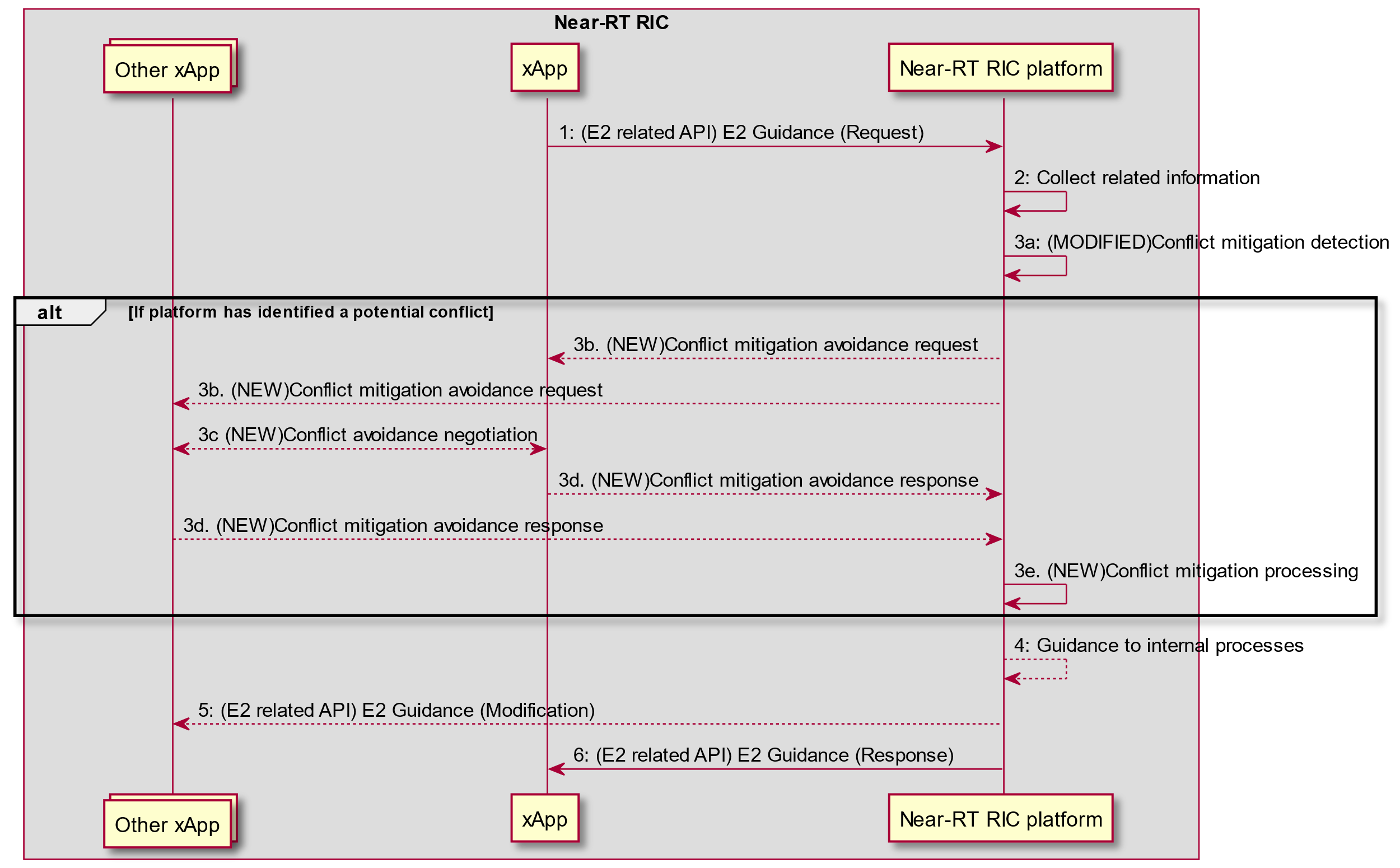


Figure 7.2-1: RICARCH Figure 9.3.3.1-1: xApp initiated E2 guidance request/response procedure

#### RICARCH 9.3.3.5 E2 Guidance modification procedure

The purpose of the E2 Guidance modification procedure is to allow Near-RT RIC platform to signal a modified guidance to the xApp, which can be triggered by the Near-RT RIC platform’s internal process (e.g., xApp subscription management), or by a message from the other xApps or external interfaces.

The guidance may also be used for Near-RT RIC platform internal purposes.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | Near-RT RIC platform indicates guidance to an xApp without request from the xApp. |
| Actors and Roles | * xApp * Near-RT RIC platform, with the following functionalities: * Database * Conflict mitigation * External entity (e.g., E2 Node) |
| Assumptions | * Near-RT RIC platform is capable of collecting related information, and generating proper guidance for an xApp. |
| Pre-conditions | * Message from xApp, message from external interfaces, or Near-RT RIC platform’s internal process triggers conflict mitigation in Near-RT RIC platform. |
| Step 1A (M) | (MODIFIED) Near-RT RIC platform identifies performs conflict mitigation detection.  This step may be avoided for the case of a message from xApp that contains an indication that the request is compliant with a previous negotiated agreement.  Near-RT RIC platform may request an xApp that was previously party to a negotiated agreement to perform an audit of another xApp’s request. |
|  | [ALT] Remaining step 1 events are executed if Near-RT RIC platform detects a potential conflict |
| Step 1B (O) | (NEW) Near-RT RIC platform requests identified xApps to perform conflict mitigation avoidance |
| Step 1C (O) | (NEW) Identified xApps uses either direct or indirect messaging to negotiate to obtain a mutually acceptable conflict avoidance agreement |
| Step 1D (O) | (NEW) Identified xApps respond to Near-RT RIC platform with outcome of negotiation |
| Step 1E (O) | (NEW) Near-RT RIC platform validates negotiated agreement and stores outcome. |
| Step 2 (O) | Near-RT RIC platform may provide guidance to the affected xApps using E2 Guidance modification message. (NEW)Message may contain a “negotiation key” that may be used in future requests from xApp |
| Step 3 (O) | Near-RT RIC platform may also use the guidance for internal purposes. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Box "Near-RT RIC"

Participant xApp

Collections “Other xApp(s)” as xApp2

Participant “Near-RT RIC platform” as plt

endbox

Collections “External entity (e.g. Non-RT RIC)” as Otherentity

Alt message from other xApp(s) triggers conflict mitigation

xApp2 -> plt: Message that may introduce conflict

Alt (NEW)message claims a previous negotiated agreement

plt <--> xApp: (NEW)Conflict mitigation may initiate avoidance audit procedure

end

Else message from external entity triggers conflict mitigation

Otherentity -> plt: Message that may introduce conflict

Else internal processing triggers conflict mitigation

plt -> plt: internal process that may introduce conflict

End

plt -> plt: 1a.(MODIFIED)Conflict mitigation detection

Alt If platform has identified a potential conflict

plt-->xApp2: 1b. (NEW)Conflict mitigation avoidance request

plt-->xApp: 1b. (NEW)Conflict mitigation avoidance request

xApp<-->xApp2: 1c (NEW)Conflict avoidance negotiation

xApp2-->plt: 1d. (NEW)Conflict mitigation avoidance response

xApp-->plt: 1d. (NEW)Conflict mitigation avoidance response

plt -> plt: 1e. (NEW)Conflict mitigation processing

end

plt --> xApp: 2a: E2 Guidance (Modification)

plt --> xApp2: 2b: E2 Guidance (Modification)

plt --> plt: 3: Guidance for internal purposes

@enduml

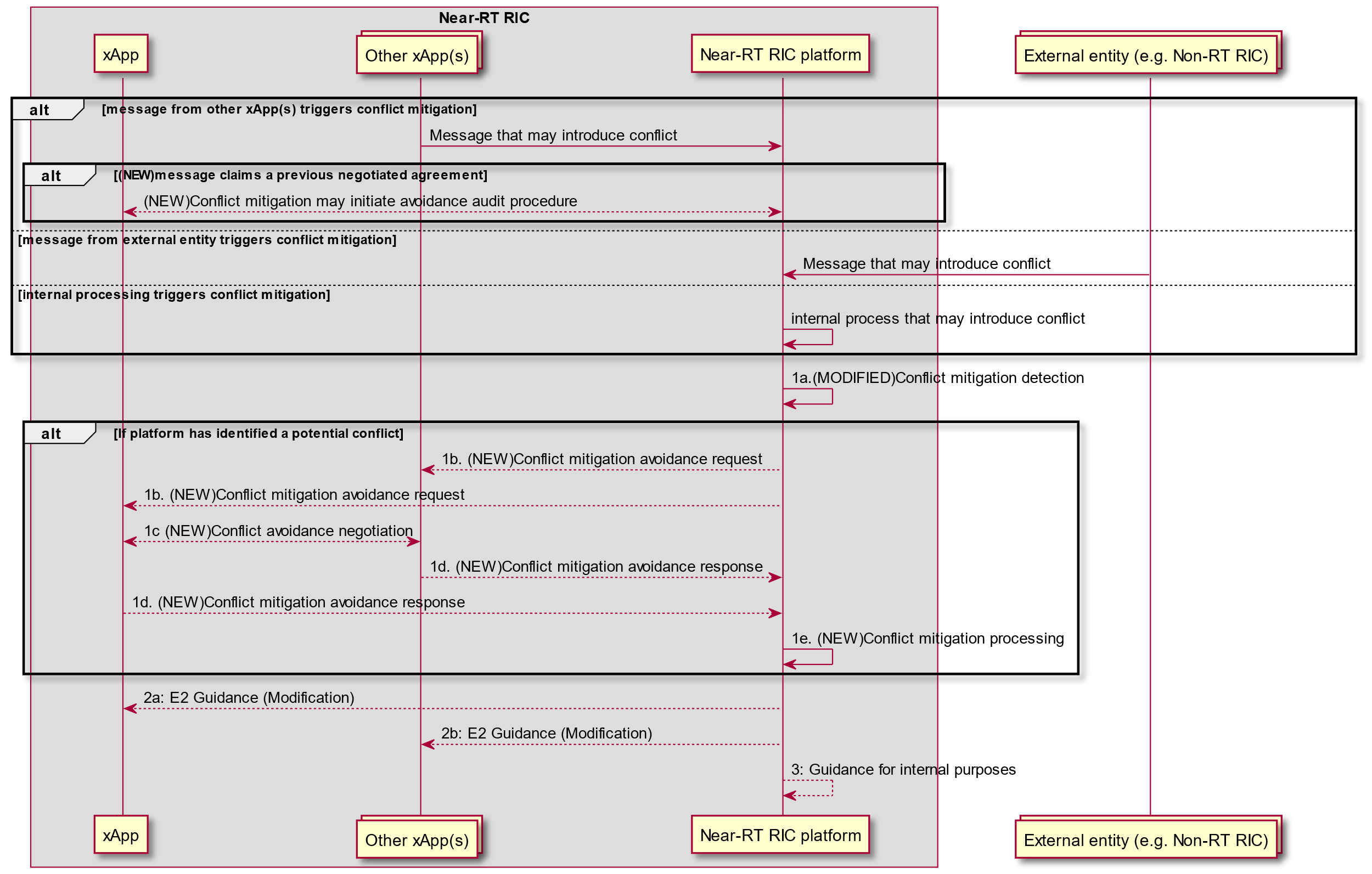


Figure 7.2-2: RICARCH Figure 9.3.3.5-1 E2 Guidance Modification Procedure

#### 7.2.1.4 Evaluation

This solution offers a sound solution for both E2AP and E2SM level direct conflict avoidance, allows xApp to negotiate without revealing business logic, supports a solution to avoid repetition of the conflict avoidance process and supports a solution to audit claims from xApps to be compliant with previous negotiated agreements.

The solution avoids the need to align Near-RT RIC platform support of each E2SM with the versions used by each xApp. It does however rely on xApps supporting the appropriate E2SM versions.

## 7.3 Solutions-to-Key Issues Mapping

Table 7.3- Solutions to Key Issue Mapping

|  | Key Issue | | |
| --- | --- | --- | --- |
| **Solution** | #1 | #Y | #Z |
| #1 | yes | - | - |
| #Y | - | - | - |
| #Z | - | - | - |

# 8 Conclusions

## 8.1 Conflict Detection

### 8.1.1 Key issues and potential solutions

All key issues are addressed by the proposed solutions as shown in clause 5.3. However, the current Near-RT RIC ARCH specification [i.1] and Near-RT RIC APIs only partially support the solutions.

### 8.1.2 Solution requirements

The following prelimanary requirements can be derived:

1. Solution #1 and #2 described in clauses 5.2.1 and 5.2.2, respectively, are both supported by existing signalling. However, Solution #2 further assumes that the Near-RT RIC platform may inspect E2SM information contained in E2 Subscription Request, E2 Control Request, and E2 Guidance Request/Response messages initiated by xApps.
2. Solution #3 requires that, when the ConMit functionality inspects the E2AP information contained in the above-mentioned messages by an xApp, it may suspect a potential conflict with another xApp. The ConMit functionality may then request xApps to perform the E2SM level analysis, thereby avoioding the need for the platform to do the same.
3. Solution #4 requires that the platform may seek further information from the xApps after identifying a potential conflict. Alternatively, Solution #4 may be triggered by the xApp which detects a potential conflict with another xApp.
4. Solution #4 requires that either the platform or a special helper xApp may collect data from the RAN for conflict discovery/detection.
5. Solution #4 requires that the platform or a special helper xApp has ability/functionality to perform AI/ML based data analysis to detect potential indirect/implicit conflicts between xApps.
6. Solutions #5, #6, and #7 assume that a certain "dependency information" containing information about couplings between RAN parameters, resources, and KPIs is generated. Given this information, the ConMit functionality can then enhance its conflict mitigation processing. The dependency information can be provided to the ConMit using, e.g., the O1 interface.
7. Solution #6 assumes that along with the dependency information, ConMit or a helper xApp may further use AI/ML methods to predict impacts of the conflicting RAN parameters on the KPIs and stability of the network.

## 8.2 Conflict Resolution

### 8.2.1 Key Issues and potential solutions

All key issues are addressed by the proposed solutions as shown in clause 6.3. However, the current near-RT RIC ARCH specification [i.1] and Near-RT RIC APIs only partially support the proposed solutions.

### 8.2.2 Solution requirements

The following prelimanary requirements can be derived:

1. Solution #1 assumes that, before resolution of direct parameter conflicts is triggered, conflicts at E2SM level (parameter level) have already been discovered and attemps at conflict avoidance by e.g., using Solution #1 in clause 7.2.1, have not been successful.
2. Solution #1 described in clauses 6.2.1 requires that ConMit requests the xApps for their preference about the RAN parameter(s) in conflict and computes a "common value" on behalf of the xApps. This solution also assumes an intelligent/computing mechanism inside the ConMit or a helper xApp to compute such a common value by taking into xApp preferences.

## 8.3 Conflict Avoidance

### 8.3.1 Key Issues and potential solutions

All key issues are addressed by the proposed solutions as shown in clause 7.3. However, the current near-RT RIC ARCH specification [i.1] and Near-RT RIC APIs only partially support the solutions.

### 8.3.2 Solution requirements

The following prelimanary requirements can be derived:

1. Solution #1 assumes that a list of xApps in conflict with each other has been generated. Solutions in clause 5 may be used for this purpose depending on the type of conflict.
2. Solution #1 works on the basis of further conflict avoidance by orthogonalizing E2SM level services usage within the given E2 Node by the xApps, assuming that such a partition or separation is possible and agreeable to the involved xApps.
3. Solution #1 requires either inter-xApp message exchange or using the ConMit functionality in the platform to route messages between xApps to negotiate a mutually acceptable partition of the E2 Node declared supported E2SM level services.

NOTE: Solution #1 may serve as a pre-requisite for conflict resolution solutions in clause 6. In more detail, the failure of an avoidance agreement between xApps means that conflicts between xApps cannot be avoided, and an intervention by the ConMit functionality is required. Solution #1 in clause 6.2.1, e.g., resolves the direct conflict in that the platform computes a common value for the conflicting RAN parameter itself.

# Annex A: Title of annex

# Annex Bibliography

# Annex: Change history/Change request (history)

|  |  |  |
| --- | --- | --- |
| Date | Revision | Description |
| 2023.11.20 | 00.00.00 | Dokument Skeleton |
| 2023.11.27 | 00.00.01v1 | Dokument Skeletion Cleanup |
| 2023.11.30 | 00.00.01v2 | Addition of subsections 4.x and 5.1.x.2, 6.1.x.2, and 7.1.x.2 |
| 2023.12.12 | 00.00.01 | Approved Baseline |
| 2024.01.15 | 00.00.02 | Inclusion of NOK CR 0001 after agreement in WG3# 210 |
| 2024.02.12 | 00.00.03 | Inclusion of NOK CRs 0002 and 0003 after agreement in WG3#2014 |
| 2024.03.19 | 00.00.04 | Inclusion of NOK CRs 0004, 0005, 0006, 0007, and 0008 after agreement |
| 2024.05.27 | 00.00.05 | Inclusion of NOK CRs 0009 and 0010 |
| 2024.07.02 | 00.00.06 | Inclusion of SYM CRs 0007, 0008, and 0009 |
| 2024.07.11 | 00.00.07 | Inclusion of NOK CR 0011 |
| 2024.07.23 | 01.00.00 | Version 1 after editorial fixes and approval vote |