 O-RAN.WG3.RICARCH-R003-v04.00

Technical Specification

O-RAN Working Group 3 (Near-Real-time RAN Intelligent

Controller and E2 Interface Workgroup)

Near-RT RIC Architecture

Copyright © 2023 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 specification 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 specification for your personal use, or copy the material of this specification 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

Foreword 4

Modal verbs terminology 4

1 Scope 5

2 References 5

2.1 Normative references 5

2.2 Informative references 6

3 Definition of terms, symbols and abbreviations 6

3.1 Terms 6

3.2 Symbols 7

3.3 Abbreviations 7

4 General Principles 8

5 Near-RT RIC Architecture 8

5.1 Requirements 8

5.1.1 Platform Requirements 8

5.1.2 xApp Requirements 9

5.1.3 Near-RT RIC API Requirements 9

5.2 Overall Architecture Description 10

6 Near-RT RIC Functions Description 11

6.1 General 11

6.2 Platform Functions 12

6.2.1 Database and SDL 12

6.2.2 xApp Subscription Management 13

6.2.3 Conflict Mitigation 13

6.2.4 Messaging Infrastructure 14

6.2.5 Security 14

6.2.6 Management Function 14

6.2.7 Interface Termination 15

6.2.8 API Enablement 16

6.2.9 AI/ML Support 16

6.2.10 xApp Repository Function 17

6.3 xApps 17

7 Near-RT RIC APIs 17

7.1 Overall Description 17

7.1.1 Introduction 17

7.1.2 Near-RT RIC API approaches 18

7.1.3 Network API approach 20

7.1.4 SDK approach 21

7.1.5 Near-RT RIC API support options 21

7.2 A1 related APIs 23

7.3 E2 related APIs 24

7.4 Management APIs 24

7.5 Void 25

7.6 SDL APIs 25

7.7 Enablement APIs 25

8 External Interfaces of Near-RT RIC 25

8.1 E2 Interface 25

8.2 A1 Interface 26

8.3 O1 Interface 26

8.4 Y1 Interface 26

9 Near-RT RIC API Procedures 26

9.1 Disclaimer 26

9.2 A1 Related API Procedures 26

9.2.1 Introduction 26

9.2.2 A1 Policy procedures 27

9.2.3 A1-EI related procedures 36

9.3 E2 Related API Procedures 47

9.3.1 Introduction 47

9.3.2 RIC Functional Procedures 48

9.3.3 E2 Guidance related procedures 63

9.4 Management API Procedures 69

9.4.1 xApp Registration procedure 69

9.4.2 xApp Deregistration procedure 70

9.4.3 Void 70

9.4.4 Create MOI 71

9.4.5 Modify MOI attributes 71

9.4.6 Delete MOI 72

9.4.7 Read MOI attributes 73

9.4.8 Notify MOI changes 74

9.4.9 Subscription Control 75

9.4.10 Fault Notification 76

9.4.11 Fault Supervision Control 76

9.4.12 Performance Data File Reporting 77

9.4.13 Report Streamed Data 78

9.4.14 Measurement Job Control 79

9.5 SDL API Procedures 79

9.5.1 SDL Client Registration procedure 79

9.5.2 SDL Client Deregistration procedure 80

9.5.3 Fetch Data procedure 80

9.5.4 Subscribe/Notify procedure 81

9.5.5 Store Data procedure 82

9.5.6 Modify Data procedure 82

9.5.7 Subscribe/Push procedure 83

9.5.8 Use cases of SDL APIs 84

9.5.9 SDL API data structures 86

9.6 API Enablement Procedures 88

9.6.1 Near-RT RIC API Discovery procedure 88

9.6.2 Procedures related to API Event Subscription and Notification 89

Revision history 93

History 93

# Foreword

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

# 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 contents of the present document are 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.

The present document specifies the overall Near-RT-RIC (Near-real-time RAN Intelligent Controller) architecture and functionalities, including interaction between hosted applications and common functions in the Near-RT RIC platform.

# 2 References

## 2.1 Normative references

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

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 necessary for the application of the present document.

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

[2] O-RAN-WG3.E2GAP, “O-RAN Working Group 3, Near-Real-time RAN Intelligent Controller, E2 General Aspects and Principles”.

[3] O-RAN-WG3.E2AP, “O-RAN Working Group 3, Near-Real-time RAN Intelligent Controller, E2 Application Protocol (E2AP)”.

[4] O-RAN-WG10.OAM Architecture, “O-RAN Operations and Maintenance Architecture”.

[5] O-RAN-WG10.O1-Interface, “O-RAN Operations and Maintenance Interface Specification”.

[6] 3GPP TS 33.401: “3GPP System Architecture Evolution (SAE); Security architecture”.

[7] 3GPP TS 33.501: “Security architecture and procedures for 5G System”.

[8] O-RAN-WG2.A1.GA&P, “O-RAN Working Group 2, A1 interface: General Aspects and Principles”.

[9] O-RAN-WG2.A1AP, “O-RAN Working Group 2, A1 Interface: Application Protocol”.

[10] O-RAN-WG1.O-RAN Architecture, “O-RAN Working Group 1, O-RAN Architecture Description”.

[11] 3GPP TS 36.401: "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Architecture Description".

[12] 3GPP TS 38.300: “NR; NR and NG-RAN Overall Description; Stage 2”.

[13] 3GPP TS 37.324: “Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Service Data Adaptation Protocol (SDAP) specification”.

[14] 3GPP TS 38.331: “NR; Radio Resource Control (RRC); Protocol specification”.

[15] 3GPP TS 38.323: “NR; Packet Data Convergence Protocol (PDCP) specification”.

[16] 3GPP TS 38.322: “NR; Radio Link Control (RLC) protocol specification”.

[17] 3GPP TS 38.321: “NR; Medium Access Control (MAC) protocol specification”.

[18] 3GPP TS 38.201: “NR; Physical layer; General description”.

[19] O-RAN-WG2.O-RAN Architecture, “O-RAN Working Group 2, AI/ML Workflow description and Requirements”.

[20] O-RAN.WG6.ORCH-USE-CASES, "Orchestration Use Cases and Requirements for O-RAN Virtualized RAN".

[21] 3GPP TS 28.622: “Generic Network Resource Model (NRM)”.

## 2.2 Informative references

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication, 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.

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

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

**Near-RT RIC**:O-RAN Near-Real-Time RAN Intelligent Controller: A logical function that enables near-real-time control and optimization of RAN elements and resources via fine-grained data collection and actions over E2 interface.

**Non-RT RIC**:O-RAN Non-Real-Time RAN Intelligent Controller: A logical function within SMO that drives the content carried across the A1 interface. It is comprised of the Non-RT RIC Framework and the Non-RT RIC Applications. Please refer to [8] for more information.

**O-CU-CP**: O-RAN Central Unit – Control Plane: a logical node hosting RRC [14] and the control plane part of PDCP protocol [15].

**O-CU-UP**: O-RAN Central Unit – User Plane: a logical node hosting the user plane part of PDCP protocol [15] and SDAP protocol [13].

**O-DU**: O-RAN Distributed Unit: a logical node hosting RLC [16]/MAC [17]/High-PHY [18] layers based on a lower layer functional split.

**O-RU**: O-RAN Radio Unit: a logical node hosting Low-PHY layer and RF processing based on a lower layer functional split.  This is similar to 3GPP’s “TRP” or “RRH” but more specific in including the Low-PHY layer (FFT/iFFT, PRACH extraction).

**O-eNB**:An eNB [10] or ng-eNB [11] that supports E2 interface.

**O1**: An interface between SMO and O-RAN managed elements, for operation and management, by which FCAPS management, PNF (Physical Network Function) Software management, File management shall be achieved.

**SMO**: A Service Management and Orchestration system as described in [4].

**A1**: An interface between Non-RT RIC and Near-RT RIC to enable policy-driven guidance of Near-RT RIC applications/functions, and support AI/ML workflow. Please refer to [8] for more information.

**E2**: An interface connecting Near-RT RIC and one or more O-CU-CPs, one or more O-CU-UPs, or one or more O-DUs.

**E2 Node**: A logical node terminating E2 interface. In this version of the specification, O-RAN nodes terminating E2 interface are:

- for NR access: O-CU-CP, O-CU-UP, O-DU or any combination as defined in [4];

- for E-UTRA access: O-eNB.

**xApp:** An application designed to run on Near-RT RIC. It may consist of one or more microservices and at the point of on-boarding it identifies which data it consumes and which data it provides. It is independent of Near-RT RIC and may be provided by any third party. The E2 interface enables a direct association between xApp and RAN functionalities.

**O-Cloud:** O-Cloud is a cloud computing platform comprising a collection of physical infrastructure nodes that meet O-RAN requirements to host the relevant O-RAN functions (such as Near-RT RIC, O-CU-CP, O-CU-UP, and O-DU), the supporting software components (such as Operating System, Virtual Machine Monitor, Container Runtime, etc.) and the appropriate management and orchestration functions.

**Y1:** An interface between Near-RT RIC and Y1 consumers [10]. The interface enables RAN analytics information exposure from Near-RT RIC.

## 3.2 Symbols

No symbols are defined in the present document.

## 3.3 Abbreviations

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

API Application Programming Interface

FM Fault Management

LCM Life-Cycle Management

ML Machine Learning

Non-RT RIC Non-real-time RAN Intelligent Controller:

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

O-CU-CP O-RAN Central Unit – Control Plane

O-CU-UP O-RAN Central Unit – User Plane

O-DU O-RAN Distributed Unit

O-RU O-RAN Radio Unit

PM Performance Management

R-NIB Radio-Network Information Base

SCTP Stream Control Transmission Protocol

SDL Shared Data Layer

SMO Service Management and Orchestration

UE-NIB UE-Network Information Base

# 4 General Principles

The architecture described herein follows the architecture principles specified in [2], and the following principle(s):

* The Near-RT RIC architecture and internal interfaces shall be open to support 3rd party xApps.

# 5 Near-RT RIC Architecture

## 5.1 Requirements

The architecture described herein fulfills the Near-RT RIC requirements specified in [2] and the following requirement(s):

* Near-RT RIC shall consist of multiple xApps and a set of platform functions that are commonly used to support the specific functions hosted by xApps.

### 5.1.1 Platform Requirements

* Near-RT RIC shall provide a database that stores an up-to-date RAN information, history of time-varying network state, as well as configurations related to E2 Nodes, Cells, Bearers, Flows, UEs, etc., and the mapping between them. This information shall be provided as a service to any xApp that requests it;
* Near-RT RIC shall provide AI/ML tools that support for data pipelining, training, and performance monitoring;
* Near-RT RIC shall provide a messaging infrastructure;
* Near-RT RIC shall provide logging, tracing and metrics collected from Near-RT RIC platform and xApps toward SMO;
* Near-RT RIC shall provide security functions;
* Near-RT RIC shall support resolution of potential conflicts or overlaps of controls from xApps toward an E2 node;
* Near-RT RIC shall communicate with xApp(s) via Near-RT RIC APIs;
* Near-RT RIC shall register the Near-RT RIC APIs it produces;
* Near-RT RIC shall be capable of discovering the Near-RT RIC APIs it consumes;
* Near-RT RIC shall provide means to resolve compatibility clashes between xApps and the Near-RT RIC services they access;
* Near-RT RIC shall support subscription merging from multiple xApps to avoid unnecessary network load;
* Near-RT RIC shall provide an O1 interface;
* Near-RT RIC shall be able to route A1 policy management messages to the registered xApps based on A1 policy type and operator policies;
* Near-RT RIC shall control access of A1-EI types for xApps based on operator policies.

### 5.1.2 xApp Requirements

* xApp may enhance the RRM capabilities of Near-RT RIC;
* xApp may be associated with zero, one or more E2SMs;
* xApp shall use Near-RT RIC APIs to make use of the Information Elements (IEs) of E2SMs that are associated with it;
* xApp that is associated with a given E2SM shall be able to interface with any E2 Node that supports that E2SM without any intermediary xApp;
* xApp shall be able to receive event-triggered information on RAN information and time-varying network state.
* xApp shall provide collected logging, tracing and metrics information to Near-RT RIC;
* xApp shall provide a descriptor that includes the following basic information of the xApp:
* Configuration: It includes a data dictionary for configuration data, i.e., meta data such as a YANG definition or a list of configuration parameters and their semantics. It may also include an initial configuration of xApp;
* Control: It includes the types of data that an xApp consumes and generates, in order to perform control capabilities (e.g., xApp URL, parameters, input/output type);
* Metrics: It includes a list of metrics (e.g., metric name, type, unit and semantics) provided by the xApp.
* The xApp descriptor shall also provide the necessary data to enable management and orchestration of the xApp, aligned with [4];
* xApps shall communicate with Near-RT RIC platform via Near-RT RIC APIs;
* xApp shall register the Near-RT RIC APIs it produces;
* xApp shall be capable of discovering the Near-RT RIC APIs they consume.

### 5.1.3 Near-RT RIC API Requirements

* Near-RT RIC shall provide APIs enabling the hosting of 3rd party xApps and xApps from the Near-RT RIC platform vendor;
* Near-RT RIC APIs shall not adversely impact low-latency and high throughput operations of Near-RT RIC. Specifically, the Near-RT RIC APIs shall support the Near-RT RIC control loop of execution time from 10 milliseconds to 1 second;
* Near-RT RIC shall provide APIs decoupled from specific implementation solutions, including a Shared Data Layer (SDL) that works as an overlay for underlying databases and enables simplified data access;
* Near-RT RIC shall provide an API repository/registry for the services provided by the Near-RT RIC platform and/or xApps;
* Near-RT RIC shall provide means for xApps to discover the published APIs based on the xApps’ needs;
* Near-RT RIC shall provide means to restrict xApps from discovering some published APIs based on configured policies;
* Near-RT RIC shall provide APIs enabling all xApps to directly use the information elements of E2SMs with which they are associated;
* Near-RT RIC shall provide APIs aiming to simplify the development of xApps and enable rapid innovation;
* Near-RT RIC shall provide Near-RT RIC APIs supporting xApp development in multiple programming languages (e.g. C, C++, Python, Go);
* Near-RT RIC APIs shall support xApp subscription management based on operators’ policies. An xApp may be restricted to interface with only a subset of E2 Nodes by such policies. Near-RT RIC shall be responsible for routing messages between this xApp and the subset of E2 Nodes;
* The xApp Management API shall provide the capability to configure the xApp;
* The xApp Management API shall provide the capability to change the administrative state of the xApp;
* The xApp Management API shall provide the capability to read the current configuration of the xApp;
* The xApp Management API shall provide the capability for the xApp to notify the Management Function of any change in the xApp configuration;
* The xApp Management API shall provide the capability for the xApp to notify the Management Function of an associated MOI Creation;
* The xApp Management API shall provide the capability for the xApp to notify the Management Function of an associated MOI Deletion;
* The xApp Management API shall provide the capability for the xApp to notify the Management Function of a fault detection;
* The xApp Management API shall provide the capability for the Consumer (SMO) to fetch the list of Active Faults on the xApp;
* The xApp Management API shall provide the capability for the xApp to notify the Management Function that there is a file available;
* The xApp Management API shall provide the capability for the Management Function to configure the xApp to send streaming data;
* The xApp Management API shall provide the capability for the Management Function to configure the xApp to collect specific PM data;

NOTE: Communication between xApps is FFS.

## 5.2 Overall Architecture Description

The overall O-RAN architecture specified in [10] describes the location and interfaces of Near-RT RIC, as well as possible deployment options.

The RRM functional allocation between Near-RT RIC and E2 Node is described in [2].

# 6 Near-RT RIC Functions Description

## 6.1 General

Near-RT RIC hosts the following functions:

* Database, and related SDL (Shared Data Layer) services, which allows reading and writing of RAN/UE information and other information required to support specific use cases;
* xApp subscription management, which merges subscriptions from different xApps and provides unified data distribution to xApps;
* Conflict mitigation, which resolves potentially overlapping or conflicting requests from multiple xApps;
* Messaging infrastructure, which enables message interaction amongst Near-RT RIC internal functions;
* Security, which provides the security scheme for xApps;
* Management Function:
* Fault management, configuration management, and performance management as a service producer to SMO;
* Logging, tracing and metrics collection, which capture, monitor and collect the status of Near-RT RIC internals and can be transferred to external system for further evaluation.
* Interface Termination:
* E2 termination, which terminates E2 interface from an E2 Node;
* A1 termination, which terminates A1 interface from Non-RT RIC;
* O1 termination, which terminates O1 interface from SMO;
* Y1 termination, which terminates Y1 interface from Y1 consumer.
* Functions hosted by xApps, which allow services to be executed at Near-RT RIC and outcomes to be sent to E2 Nodes via E2 interface;
* API Enablement function supporting capabilities related to Near-RT RIC API operations (API repository/registry, authentication, discovery, generic event subscription, etc.);
* AI/ML support:
* Data pipelining, Training, and Performance monitoring for xApps.
* xApp Repository Function:
* Selection of xApps for A1 message routing based on A1 policy types and operator policies;
* Access control of A1-EI types for xApps based on operator policies.

NOTE: The LCM of xApp is performed by SMO and O-Cloud as specified in [4] and [20].

This is summarized in the figure below.

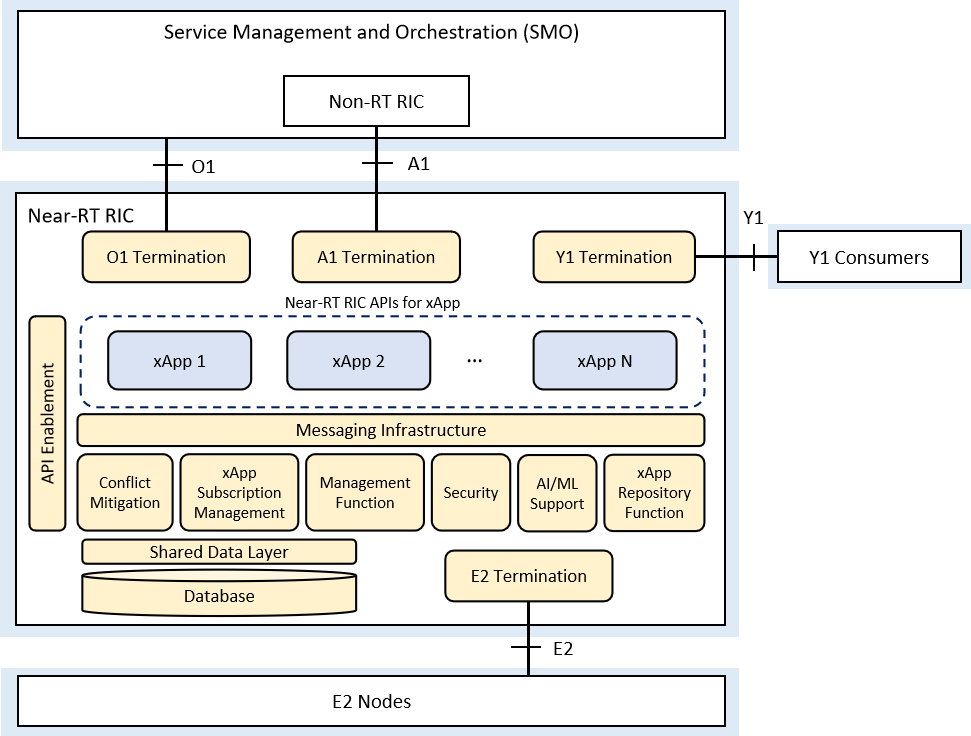


Figure 6.1-1: Near-RT RIC Internal Architecture

## 6.2 Platform Functions

### 6.2.1 Database and SDL

#### 6.2.1.1 UE-NIB

The Near-RT RIC and some xApps may generate or access UE related information to be stored in the UE-NIB database:

* UE-NIB maintains a list of UEs and associated data;
* UE-NIB maintains tracking and correlation of the UE identities associated with the connected E2 Nodes.

#### 6.2.1.2 R-NIB

The Near-RT RIC and some xApps may generate or access network related information to be stored in the R-NIB database:

* The R-NIB stores the configurations and near real-time information relating to connected E2 Nodes and the mappings between them.

#### 6.2.1.3 SDL

The SDL (Shared Data Layer) is used by xApps to subscribe to database notification services and to read, write and modify information stored on the database. UE-NIB, R-NIB and other use case specific information may be exposed using the SDL services.

### 6.2.2 xApp Subscription Management

* xApp subscription management manages subscriptions from xApps to E2 Nodes;
* xApp subscription management enforces authorization of policies controlling xApp access to messages;
* xApp subscription management enables merging of identical subscriptions from different xApps into a single subscription toward an E2 Node.

### 6.2.3 Conflict Mitigation

In the context of Near-RT RIC, Conflict Mitigation is about addressing conflicting interactions between different xApps. An application will typically change one or more parameters with the objective of optimizing a specific metric. Conflict Mitigation is necessary because xApps objectives may be chosen/configured such that they result in conflicting actions.

The control target of the radio resource management can be a cell, a UE or a bearer, etc. The control contents of the radio resource management can cover access control, bearer control, handover control, QoS control, resource assignment and so on. The control time span indicates the valid control duration which is expected by the control request. The conflicts of control can be illustrated as below.

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

* Two or more xApps request different settings for the very same configuration of one or more parameters of a Control Target. Conflict mitigation processes the requests and decides on a resolution.
* The new request from an xApp may conflict with the running configuration resulting from a previous request of another or the same xApp.
* The total requested resources from different xApps may exceed the limitation of the RAN system, e.g. the sum of resources required by the two different xApps may be far beyond the resource limitation of the RAN system.

2) Indirect Conflicts: The conflicts cannot be observed directly, nevertheless, some dependence among the parameters and resources that the xApps target can be observed. Conflict Mitigation may anticipate the possible conflicts and take actions to mitigate them. For instance, different xApps target different configuration parameters to optimize the same metric according to the respective objective. 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. E.g., antenna tilts and measurement offsets are different control points, but they both impact the handover boundary.

3) Implicit Conflicts: The conflicts cannot be observed directly, even the dependence between xApps are 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.

For mitigating these conflicts, different approaches exist:

1) Direct conflicts typically can be mitigated by pre-action coordination, i.e., the xApps or a Conflict Mitigation component needs to make the final determination on whether any specific change is made, or in which order the changes are applied.

2) Indirect conflicts can be resolved by post-action verification. Here, the actions are executed and the effects on the target metric are observed. Based on the observations, the system has to decide on potential corrections, e.g., rolling back one of the xApp actions.

3) Implicit conflicts are the most difficult to mitigate since these dependencies are difficult or impossible to observe and therefore hard to model in any mitigation scheme. In some cases, it may be possible to design around such conflicts by ensuring that use cases (xApps) target different parameters, thus falling back to approach 2), but preferably, a generic approach to managing such conflicts is established.

The individual xApp goals are defined by A1 policies, but it is also important to define utility metrics that incorporate the relative importance of each of the metrics targeted by the xApps as well as the importance of the optimization (use case). A Conflict Mitigation function may also use ML approaches, e.g., Reinforcement Learning, to a-priori assess, for each proposed change, the likely probability of degrading a metric versus the potential improvement.

### 6.2.4 Messaging Infrastructure

Messaging infrastructure provides low-latency message delivery service between Near-RT RIC internal endpoints.

* It supports registration/discovery/deletion of endpoints:
* Registration: Endpoints register themselves to the messaging infrastructure;
* Discovery: Endpoints are discovered by the messaging infrastructure initially and registered to the messaging infrastructure;
* Deletion: Endpoints are deleted once they are not used anymore.
* It provides the following APIs:
* An API for sending messages to the messaging infrastructure;
* An API for receiving messages from the messaging infrastructure.
* It supports multiple messaging modes, e.g. point-to-point mode (e.g. message exchange among endpoints), publish/subscribe mode (e.g. real-time data dispatching from E2 termination to multiple subscriber xApps);
* It provides message routing, namely according to the message routing information, messages can be dispatched to different endpoints;
* It supports message robustness to avoid data loss during a messaging infrastructure outage/restart or to release resources from the messaging infrastructure once a message is outdated.

### 6.2.5 Security

The security function given in this section only applies to Near-RT RIC. One of the targets is to prevent malicious xApps from abusing radio network information (e.g. exporting to unauthorized external systems) and/or control capabilities over RAN functions. The security requirements for the 3GPP LTE eNB is defined in [6] and for the 5G NR gNB in [7].

NOTE: The description of security functions is not included in the release.

### 6.2.6 Management Function

#### 6.2.6.1 Void

#### 6.2.6.2 OAM Management of Near-RT RIC

OAM management consists of fault, configuration, accounting, performance, file, security and other management plane services. OAM management follows O1 related management aspects defined in [4].

To support OAM management services, Near-RT RIC provides the following capabilities:

* Fault Management: The Near-RT RIC provides Near-RT RIC platform Fault Supervision MnS over the O1 interface as defined in [4];
* Configuration management: The Near-RT RIC provides Near-RT RIC platform Provisioning MnS over the O1 interface as defined in [4];
* Logging: logging is to capture information needed to operate, troubleshoot and report on the performance of the Near-RT RIC platform and its constituent components. Log records may be viewed and consumed directly by users and systems, indexed and loaded into a data storage, and used to compute metrics and generate reports. Near-RT RIC components log events according to a common logging format. Different logs can be generated (e.g., audit log, metrics log, error log and debug log).
* Tracing: tracing mechanisms are needed to monitor the transactions or a workflow. An example subscription workflow can be broken into two traces namely, a subscription request trace followed by a response trace. Individual traces can be analysed to understand timing latencies as the workflow traverses a particular Near-RT RIC component.
* Metrics collection: metrics for performance and fault management specific to each xApp logic and other internal functions are collected and published for authorized consumer (e.g., SMO). A metrics collection mechanism is needed to collect and report metrics.

### 6.2.7 Interface Termination

#### 6.2.7.1 E2 Termination

* E2 Termination terminates SCTP connection from each E2 Node;
* E2 Termination routes messages from xApps through the SCTP connection to an E2 Node;
* E2 Termination decodes the payload of an incoming ASN.1 message enough to determine message type;
* E2 Termination handles incoming E2 messages related to E2 connectivity;
* E2 Termination receives and respond to the E2 Setup Request from an E2 Node;
* E2 Termination notifies xApps of the list of RAN functions supported by an E2 Node based on information derived from the E2 Setup and RIC Service Update procedures [3];
* E2 Termination notifies the newly connected E2 Node of the list of accepted functions.

#### 6.2.7.2 A1 Termination

A1 Termination provides a generic API by means of which Near-RT RIC can receive and send messages via A1 interface [8]. These include, e.g., A1 policies and enrichment information received from Non-RT RIC, or A1 policy feedback sent towards Non-RT RIC.

#### 6.2.7.3 O1 Termination

An implementation of O1 Termination at Near-RT RIC depends on the deployment options described in [4], i.e. when Near-RT RIC is modelled as a stand-alone Managed Element.

O1 Termination communicates with SMO via O1 interface and exposes O1-related management services [5] from Near-RT RIC. For the O1 management services (MnS) shown below, Near-RT RIC is the MnS producer and SMO is the MnS consumer:

* O1 Termination exposes provisioning management services from Near-RT RIC to O1 provisioning management service consumer;
* O1 Termination supports translation of O1 management services to Near-RT RIC internal APIs;
* O1 Termination exposes FM services to report faults and events from Near-RT RIC to O1 FM service consumer;
* O1 Termination exposes PM services to report bulk and real-time PM data from Near-RT RIC to O1 PM service consumer;
* O1 Termination exposes file management services to download ML files, software files, etc. and upload log/trace files to/from file MnS consumer;
* O1 Termination exposes communication surveillance services to O1 communication surveillance service consumer.

#### 6.2.7.4 Y1 Termination

Y1 Termination communicates with Y1 consumers via Y1 interface and exposes RAN analytics information service(s) from Near-RT RIC.

### 6.2.8 API Enablement

In the context of Near-RT RIC, the Near-RT RIC APIs can be categorized based on the interaction with the Near-RT RIC platform and can be related to E2-related services, A1-related services, Management Function, and Database services.

API Enablement provides support for registration, discovery and consumption of Near-RT RIC APIs within the Near-RT RIC scope. In particular, the API enablement services include:

* Repository / Registry services for the Near-RT RIC APIs;
* Services that allow discovery of the registered Near-RT RIC APIs;
* Services to authenticate xApps for use of the Near-RT RIC APIs;
* Services that enable generic subscription and event notification;
* Means to avoid compatibility clashes between xApps and the services they access.

The API enablement services can be accessed by the xApps via one or more enablement APIs.

NOTE: The provided enablement APIs may need to consider the level of trust related to the xApp (e.g. 3rd party xApp, RIC-owned xApp, etc.).

### 6.2.9 AI/ML Support

#### 6.2.9.1 Data Pipeline

The AI/ML data pipeline in Near-RT RIC offers data ingestion and preparation for applications (xApps).

The input to the AI/ML data pipeline may include E2 node data collected over E2 interface, enrichment information over A1 interface, information from applications, and data retrieved from the Near-RT RIC database through the messaging infrastructure.

The output of the AI/ML data pipeline may be provided to the AI/ML training capability in Near-RT RIC.

#### 6.2.9.2 Training

The AI/ML training in Near-RT RIC offers training of applications (xApps) within Near-RT RIC [19]. The AI/ML training provides generic and use case-independent capabilities to AI/ML-based applications that may be useful to multiple use cases.

### 6.2.10 xApp Repository Function

* xApp Repository Function performs selection of xApps for A1 message routing based on policy type and operator policies;
* xApp Repository Function provides the policy types supported in Near-RT RIC to the A1 termination function. The supported policy types are based on policy types supported by the registered xApps and operator policies;
* xApp Repository Function enforces xApp access control to requested A1-EI type based on operator policies.

## 6.3 xApps

The xApps collaborate with the Near-RT RIC platform functions to support various specialized use cases.

Upon registration to the Near-RT RIC platform, an xApp informs the Near-RT RIC platform of its OAM and control information to enable relevant functionalities.

# 7 Near-RT RIC APIs

## 7.1 Overall Description

### 7.1.1 Introduction

The Near-RT RIC APIs are a collection of well-defined interfaces providing Near-RT RIC platform services. These APIs need to explicitly define the possible types of information flows and data models. The Near-RT RIC APIs are essential to host 3rd party xApps in an inter-operable way on different Near-RT RIC platforms.

Near-RT RIC provides the following Near-RT RIC APIs for xApps as shown in Figure 7.1-1:

* A1 related APIs: APIs allowing to access A1 related functionality;
* E2 related APIs: APIs allowing to access E2 related functionality and associated xApp Subscription Management and Conflict Mitigation functionality;
* Management APIs: APIs allowing to access management related functionality;
* SDL APIs: APIs allowing to access Shared Data Layer related functionality;
* Enablement APIs: APIs between xApps and API enablement functionality.

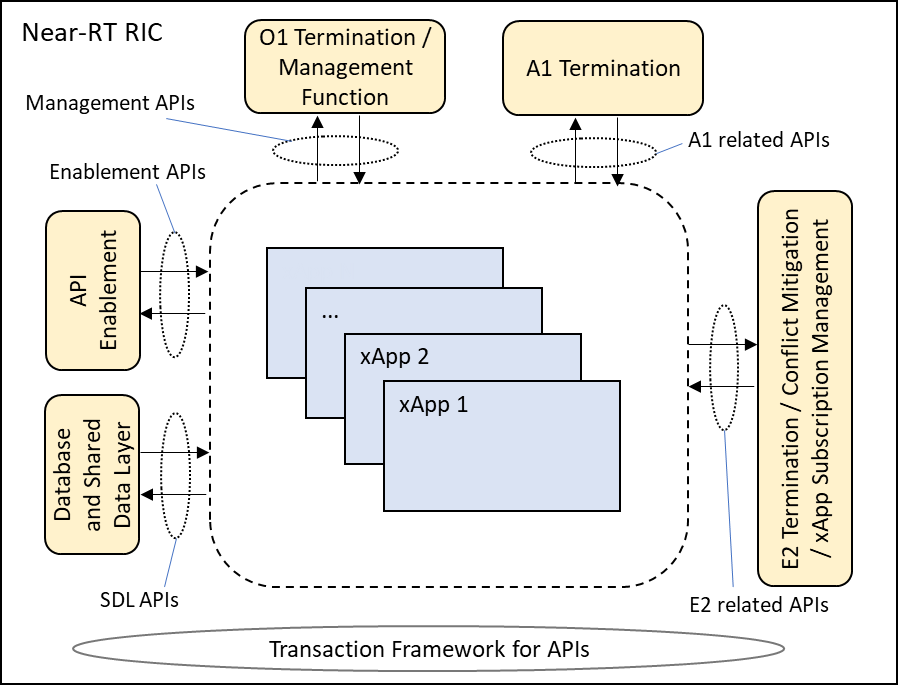


Figure 7.1-1: Overview of Near-RT RIC APIs

### 7.1.2 Near-RT RIC API approaches

Near-RT RIC APIs may be implemented using two approaches:

* **Network API approach**: Each relevant Near-RT RIC endpoint exposes a network endpoint and specifies a particular data encoding protocol and a network transport protocol that should be used to communicate with it. A different network API may be specified for each Near-RT RIC API reflecting a trade-off between different service requirements.
* **SDK approach**: The Near-RT RIC vendor provides an SDK (software development kit). This SDK is a software library which handles all connection management and exposes a simple API for the xApp to interact with the Near-RT RIC. The interface between the SDK library embedded in the xApp and the Near-RT RIC Platform may be either vendor proprietary or aligned to the specified Network API.

Both approaches use similar methodologies of interaction with Near-RT RIC as defined in sections 7.2-7.7 and chapter 9. Furthermore, the detailed messages and IEs for each RIC API that the xApp will use largely stay the same in the two approaches.

These two approaches are presented in figure 7.1.2-1 for the Network API approach and figure 7.1.2-2 for the SDK approach.

Both approaches assume that the Near-RT RIC platform provides appropriate support services. These services, labelled “Network API support” and “SDK support” respectively in figures 7.1.2-1 and 7.1.2-2, may be integrated into other platform services (i.e. A1 Termination may provide the support services for A1 related APIs) or implemented as independent services.

@startuml

skin rose

card Near-RT-RIC {

node xApp2 as "xApp"

node xApp1 as "xApp"

frame Platform {

agent Netsup as "Network API support"

node Serv1

node Serv2

node Serv3

}

}

xApp1 –- Netsup : “Network APIs”

xApp2 -– Netsup : “Network APIs”

Netsup -- Serv1

Netsup -- Serv2

Netsup -- Serv3

@enduml

图示

描述已自动生成

Figure 7.1.2-1: Near-RT RIC platform supporting RIC APIs using Network API approach

@startuml

skin rose

card Near-RT-RIC {

frame xApp2 {

collections Logic2 as "Business Logic "

agent SDK2 as "SDK library"

}

frame xApp1 {

collections Logic1 as "Business Logic "

agent SDK1 as "SDK library"

}

frame Platform {

agent SDKsup as "SDK support"

node Serv1

node Serv2

node Serv3

}

}

Logic1 -- SDK1 : SDK APIs

Logic2 -- SDK2 : SDK APIs

SDK1 -- SDKsup : "Not specified"

SDK2 -- SDKsup : "Not specified"

SDKsup -- Serv1

SDKsup -- Serv2

SDKsup -- Serv3

@enduml

图示

描述已自动生成

Figure 7.1.2-2: Near-RT RIC platform supporting RIC APIs using SDK approach

### 7.1.3 Network API approach

A Network API is defined in terms of a protocol stack including the definition:

* An application layer protocol used to carry a set of messages which normally contain multiple Information Elements (IEs);
* A data encoding protocol (ASN.1, Protobuf, JSON, etc.);
* A network transport protocol (SCTP, HTTPS, gRPC, etc.);
* Associated security and encryption methods.

For the Network API approach for Near-RT RIC APIs it is expected that each case listed in section 7.1.1 may require a unique protocol stack definition which offers an optimal solution balancing the trade-offs between service requirements, Near-RT RIC platform complexity and xApp implementation overheads.

Furthermore, each Network API will require a unique Near-RT RIC platform endpoint that the xApp will need to discover using a combination of configuration and use of the Enablement APIs (see section 7.7).

Open O-RAN specified Network APIs for each case listed in section 7.1.1 are defined in terms of the corresponding protocol stack.

### 7.1.4 SDK approach

An SDK (Software development Kit) is defined as a set of software tools/libraries provided by platform vendors which helps developers in building applications on the platform. An SDK simplifies application development by:

* Providing simple APIs to trigger commonly used functionality;
* Handling routine management tasks “under the hood”;
* Providing tools for debugging, building, testing applications.

For the SDK approach for Near-RT RIC APIs, it is expected that a given Near-RT RIC vendor offers an SDK library that provides a language specific implementation of each case listed in section 7.1.1. This provides a common interface to interact with all the Near-RT RIC endpoints as well as other xApps.

The Near-RT RIC SDK library functionality would include:

* Discovery of relevant endpoints for and connection to Near-RT RIC platform services and other xApps.
* Handles all connection management of the xApp with all relevant Near-RT RIC platform endpoints as well as other xApps. This includes:
* Authentication;
* Registration;
* Encryption/Decryption (if required);
* Failovers if any endpoint crashes.
* Presents a simple API as function definitions for sending/receiving messages;
* Provides tools for xApp debugging, building and testing.

These services would be provided by the SDK library and so would not be exposed to the business logic within an xApp.

Open O-RAN specified SDK APIs for each case listed in section 7.1.1 would need to be defined for each supported programming language.

The actual protocol stack implemented by the SDK library and used to communicate with the SDK support service in the Near-RT RIC platform may be either based on the Network APIs or left not specified and so may be vendor proprietary.

### 7.1.5 Near-RT RIC API support options

The two approaches to Near-RT RIC APIs are not necessarily mutually exclusive.

For example, an xApp designed to support the Network API approach may be implemented using either an internal SDK library that provides the Network API interface or a direct implementation the Network API interface. Both of these cases are presented in figure 7.1.5-1.

@startuml

skin rose

card Near-RT-RIC {

frame xApp2 {

collections Logic2 as "Business Logic "

agent IF2 as "Network API client"

}

frame xApp1 {

collections Logic1 as "Business Logic "

agent SDK1 as "SDK library \nincluding Network API client"

}

frame Platform {

agent Netsup as "Network API support"

node Serv1

node Serv2

node Serv3

}

}

Logic1 -- SDK1 : SDK APIs

Logic2 -- IF2 #line.dashed

SDK1 -- Netsup : "Network APIs"

IF2 -– Netsup : "Network APIs"

Netsup -- Serv1

Netsup -- Serv2

Netsup -- Serv3

@enduml

图示, 示意图

描述已自动生成

Figure 7.1.5-1: Near-RT RIC platform supporting RIC APIs using Network API approach with optional use of SDK APIs

Likewise, a Near-RT RIC Platform may be designed to support both the Network API approach and the SDK approach and so xApps implemented using either approach may access platform services. Both of these cases are presented in figure 7.1.5-2.

@startuml

skin rose

card Near-RT-RIC {

frame xApp2 {

collections Logic2 as "Business Logic "

agent SDK2 as "SDK library"

}

frame xApp1 {

collections Logic1 as "Business Logic "

agent SDK1 as "SDK library \nincluding Network API client"

}

frame Platform {

agent Netsup as "Network API support"

agent SDKsup as "SDK support"

node Serv1

node Serv2

node Serv3

}

}

Logic1 -- SDK1 : SDK APIs

Logic2 -- SDK2 : SDK APIs

SDK1 -- Netsup : “Network APIs”

SDK2 –- SDKsup : "Not specified"

Netsup -- Serv1

Netsup -- Serv2

Netsup -- Serv3

SDKsup -- Serv1

SDKsup -- Serv2

SDKsup -- Serv3

@enduml

图示

描述已自动生成

Figure 7.1.5-2: Near-RT RIC platform supporting both Network API and SDK approaches

## 7.2 A1 related APIs

The xApps in Near-RT RIC provide value added services based on the policies or enrichment information or both which are transferred through A1 interface by Non-RT RIC. A1 related APIs allow access to A1 related functionality, which includes:

Policy Enforcement API: used for policy enforcement request/response.

* A1 Policy Setup (Request, Result);
* A1 Policy Update (Request, Result);
* A1 Policy Delete (Request, Result);
* A1 Policy Query (Request, Result);
* A1 Policy Status (Status, Ack).

Enrichment Information API: used for enrichment information transfer/response.

* A1 EI Query (Request, Result);
* A1 EI Job Setup (Request, Result);
* A1 EI Job Update (Request, Result);
* A1 EI Job Delete (Request, Result);
* A1 EI Job Status Query (Request, Result);
* A1 EI Job Status Notify (Status, Ack);
* A1 EI Job Result Delivery (Result, Ack).

## 7.3 E2 related APIs

The xApps in Near-RT RIC provide value added services using RIC functional mechanisms and related procedures via E2 interface towards E2 Nodes. The E2 related APIs allow access to E2 related functionality and the associated xApp Subscription Management and Conflict Mitigation functionality, which includes:

* E2 Subscription (Request, Reject/Success/Failure);
* E2 Subscription Delete (Request, Reject/Success/Failure);
* E2 Subscription Delete Query (Required, Accept/Decline);
* E2 Subscription Delete Notification (Deleted);
* E2 Indication (Push, Failure);
* E2 Control (Request, Reject/Success/Failure);
* E2 Guidance (Request, Response, Modification).

NOTE 1: Support for E2 interface Global Procedures (E2 SETUP, RIC Service Update, etc.) is described in section 9.5.8 as use cases of SDL APIs.

NOTE 2: Addition Conflict Mitigation related messages are FFS (i.e. for "analysis results").

## 7.4 Management APIs

The Management APIs include the following APIs, including the xApp’s ML Model related APIs and the FCAPS related APIs.

The xApp’s ML Model related APIs include:

* ML Model Deployment Request;
* ML Model Update Request;
* ML Model Uninstall Request;

The FCAPS related APIs include:

* xApp Registration API: It supports an xApp to register to the Near-RT RIC platform after the xApp is deployed;
* xApp Deregistration API: It supports an xApp to deregister from the Near-RT RIC platform before the xApp is terminated;
* Configuration API: It transfers configurations from SMO to an xApp;
* PM API: It supports an xApp to provide PM related data to the O1 PM Consumer;
* FM API: It supports an xApp to provide faults and events information to the O1 FM Consumer.

NOTE: Trace related API is FFS.

## 7.5 Void

Void

## 7.6 SDL APIs

The SDL APIs provide a simple yet flexible way to store and retrieve data while hiding details such as type and location of database, management operations of database layer such as high availability, scaling, load-balancing. SDL APIs support defined data structures used to provide E2 Node related information.

The SDL APIs allow access to the following Shared Data Layer functionality:

* SDL Client Registration: allows xApps to request the SDL for permissions to access the database;
* SDL Client Deregistration: allows xApps to request the SDL to deregister;
* Fetch Data: allows xApps to fetch data from the database;
* Subscribe: allows xApps to subscribe to notifications or pushes related to updates of data;
* Notify: notifies xApps about updates of information in the database. When an update occurs, the SDL will send notifications to those xApps that are subscribed to receive them;
* Push: When a subscribed update occurs in the database, the SDL will, in a single message, send push data and information about the update to those xApps that subscribed to receive them;
* Store Data: allows xApps to store data in the database;
* Modify Data: allows xApps to modify or delete data from the database.

## 7.7 Enablement APIs

The Enablement APIs support exchange of API enablement related information between xApps and the API enablement functionality.

The API enablement functionality includes:

* API Registration: enabling the API producer to register the APIs it produces;

NOTE: An investigation of services for which the API producer is the xApp is FFS.

* API Discovery: enabling discovery of the Near-RT RIC APIs by xApps;
* API-related Event Subscription: enabling subscription/un-subscription to API-related events and event notifications.

# 8 External Interfaces of Near-RT RIC

## 8.1 E2 Interface

O-RAN-WG3.E2GAP [2] specifies E2 interface general aspects and principles.

O-RAN-WG3.E2AP [3] specifies E2 interface application protocols.

## 8.2 A1 Interface

O-RAN-WG2.A1.GA&P [8] specifies A1 interface general aspects and principles.

O-RAN-WG2.A1AP [9] specifies A1 interface application protocols.

## 8.3 O1 Interface

An implementation of O1 interface at Near-RT RIC depends on the deployment options described in [4], i.e. when Near-RT RIC is modelled as a stand-alone Managed Element.

O-RAN-WG1.O1-Interface [5] specifies O1 interface related aspects.

## 8.4 Y1 Interface

Y1 interface allows the Y1 consumers to subscribe to or request the RAN analytics information service(s) provided by Near-RT RIC [10].

NOTE: The Stage 2 and Stage 3 aspects of Y1 interface are not addressed in the present document.

# 9 Near-RT RIC API Procedures

## 9.1 Disclaimer

Procedures described in this section assume the Near-RT RIC platform contains the functions listed in section 6.2.  
API endpoints associated with the platform functions are for information and may vary depending on implementation.

## 9.2 A1 Related API Procedures

### 9.2.1 Introduction

The following procedures are described in this section:

**1) Policy Management Service.**

In this case the A1 Service Producer role (A1AP [9] section 3.1) resides in the Near-RT RIC.

a) Non-RT RIC (A1 Service Consumer) originated procedures:

* Create Single Policy procedure (A1AP [9] section 3.2.2.2.2) using **A1 Related API: A1 Policy Setup** (request, result);
* Query Single Policy procedure (A1AP [9] section 3.2.2.3.2) using **A1 Related API: A1 Policy Query** (request, result);
* Query All Policy Identifiers procedure (A1AP [9] section 3.2.2.3.5) using **A1 Related API: A1 Policy Query** (request, result);
* Query Policy Status procedure (A1AP [9] section 3.2.2.3.6) using **A1 Related API: A1 Policy Query** (request, result);
* Update Single Policy procedure (A1AP [9] section 3.2.2.4.2) using **A1 Related API: A1 Policy Update** (request, result);
* Delete Single Policy procedure (A1AP [9] section 3.2.2.5.2) using **A1 Related API: A1 Policy Delete** (request, result);
* Query All Policy Type Identifiers procedure (A1AP [9] section 3.2.2.7.2) using **A1 Related API: A1 Policy Query** (request, result);
* Query Single Policy Type procedure (A1AP [9] section 3.2.2.7.3) using **A1 Related API: A1 Policy Query** (request, result).

b) Near-RT RIC (A1 Service Producer) originated procedures:

* Policy Status Update procedure (A1AP [9] section 3.2.2.6.2) using **A1 Related API: A1 Policy Status** (status, ack).

**2) A1AP Enrichment Information (EI) Service**

In this case the A1 Service Producer role (A1AP [9] section 3.1) resides in the Non-RT RIC.

a) Near-RT RIC (A1 Service Consumer) originated procedures:

* Query EI Type Identifiers procedure (A1AP [9] section 3.3.2.2.2) using **A1 Related API: A1 EI Query** (request, result);
* Query EI Type procedure (A1AP [9] section 3.3.2.2.3) using **A1 Related API: A1 EI Query** (request, result);
* Query EI Job Identifiers procedure (A1AP [9] section 3.3.3.2.2) using **A1 Related API: A1 EI Query** (request, result);
* Create EI Job procedure (A1AP [9] section 3.3.3.3.2) using **A1 Related API: A1 EI Job Setup** (request, result);
* Query EI Job procedure (A1AP [9] section 3.3.3.3.3) using **A1 Related API: A1 EI Query** (request, result);
* Update EI Job procedure (A1AP [9] section 3.3.3.3.4) using **A1 Related API: A1 EI Job Update** (request, result);
* Delete EI Job procedure (A1AP [9] section 3.3.3.3.5) using **A1 Related API: A1 EI Job Delete** (request, result);
* Query EI Job Status procedure (A1AP [9] section 3.3.3.4.2) using **A1 Related API: A1 EI Job Status Query** (request, result).

b) Non-RT RIC (A1 Service Producer) originated procedures:

* Notify EI Job Status procedure (A1AP [9] section 3.3.3.4.3) using **A1 Related API: A1 EI Job Status Notify** (status, ack);
* Deliver EI Job Result procedure (A1AP [9] section 3.3.4.2.2) using **A1 Related API: A1 EI Job Result Delivery** (result, ack).

### 9.2.2 A1 Policy procedures

#### 9.2.2.1 A1 Policy Setup procedure

The purpose of A1 Policy Setup procedure in Near-RT RIC is to support the A1AP Create Single Policy procedure from the non-RT RIC and to request and activate an A1 policy enforcement toward a suitable xApp.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | The activation of a received but not activated A1 policy in Near-RT RIC. |
| Actors and Roles | * Non-RT RIC: originator of HTTP PUT request to activate an A1 policy; * xApp: responsible for the enforcement of assigned A1 policies; * Near-RT RIC platform:   + A1 Termination: originator of A1 related API request to activate an A1 policy;   + xApp Repository Function: handling of A1 policy mapping to xApp. |
| Assumptions | * A1 policy types supported in Near-RT RIC are accessible by A1 termination; * A1 policies created in Near-RT RIC are accessible by A1 termination; * xApp selection needed information is accessible by xApp Repository Function. xApp selection process is performed at xApp Repository Function. |
| Pre conditions | * Near-RT RIC is running normally; * A1 interface is connected and activated. |
| Begins when | Non-RT RIC determines to trigger the activation of one A1 policy towards the Near-RT RIC. |
| Step 1 (M) | Non-RT RIC requests Near-RT RIC to set up an A1 policy. |
| Step 2 (M) | A1 Termination checks whether the received policy type is supported in Near-RT RIC or not. |
|  | [ALT] Step 3 is executed if the received policy type is not supported in Near-RT RIC: |
| Step 3 (M) | A “400” response is directly returned to Non-RT RIC. |
|  | [ELSE] Steps 4-14 are executed if the received policy type is supported in Near-RT RIC: |
| Step 4 (M) | A1 Termination then checks whether the received A1 policy is mapped to any xApp or not. |
| Step 5 [Informative] | (NOTE 1) If the received A1 policy is not mapped to any xApp, A1 Termination triggers xApp selection procedure to identify a suitable xApp by sending xApp SELECTION REQUEST to xApp Repository Function. The message includes at least A1 policy received from Non-RT RIC. |
| Step 6  [Informative] | xApp selection process is performed at xApp Repository Function. |
|  | [ALT] Steps 7-12 are executed if one or more suitable xApps are identified: |
| Step 7  [Informative] | Candidate xApps are notified to A1 Termination. |
| Step 8 (M) | A1 Termination selects a suitable one from candidate xApps as the target of **A1 related API: A1 POLICY SETUP REQUEST** message. The message includes at least the corresponding A1 policy received from Non-RT RIC. |
|  | [ALT] Steps 9-10 are executed if the selected xApp accomplishes setup of the A1 policy: |
| Step 9-10 (M) | A success is notified to A1 termination in **A1 related API: A1 POLICY SETUP RESULT** message, and eventually to Non-RT RIC. |
|  | [ELSE] Steps 11-12 are executed if the selected xApp fails to set up the A1 policy: |
| Step 11-12 (M) | A failure is notified to A1 termination in **A1 related API: A1 POLICY SETUP RESULT** message, and eventually to Non-RT RIC. |
|  | [ELSE] Steps 13-14 are executed if no suitable xApps is identified: |
| Step 13-14 (M) | A failure is notified to A1 termination [Informative]. A “503” response is directly returned to Non-RT RIC. |
| NOTE 1: The create policy and update policy procedure in A1 interface has the identical HTTP PUT Request, so when a HTTP PUT Request is received, A1 Termination cannot decide whether it’s asked for create or update from a first glance. Only after further policy mapping check to find out the enforcement status of the received A1 policy, can the purpose of the received HTTP PUT request be identified. If the received A1 policy is not mapped to any xApp, A1 Termination triggers A1 Policy Setup procedure as shown here. Otherwise, A1 Termination triggers A1 Policy Update procedure. See 9.2.2.2 for details. | |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 8

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

Participant rf as “xApp Repository Function”

End box

Participant ap as “xApp”

autonumber

ric -> aot:<<A1AP>>PUT .../policytypes/{policyTypeId}/policies/{policyId}(PolicyObject)

aot<-aot: policy type check

alt policy type is not supported in Near-RT RIC

aot -> ric:<<A1AP>>400

else policy type is supported in Near-RT RIC

aot->aot: policy mapping check: not mapped

aot -> rf:xApp SELECTION REQUEST\n[A1 policy]

rf->rf:xApp selection process

alt suitable xApp is identified

rf ->aot:xApp SELECTION RESULT\n[success, candidate xApp(s)]

aot -> ap:A1 POLICY SETUP REQUEST

alt success

ap -> aot:A1 POLICY SETUP RESULT\n[success]

aot -> ric:<<A1AP>>201 Created(PolicyObject)

else failure

ap -> aot: A1 POLICY SETUP RESULT \n[failure, reason]

aot -> ric:<<A1AP>>error code xxx

end

else no suitable xApp is identified

rf ->aot:xApp SELECTION RESULT\n[failure, no candidate xApp]

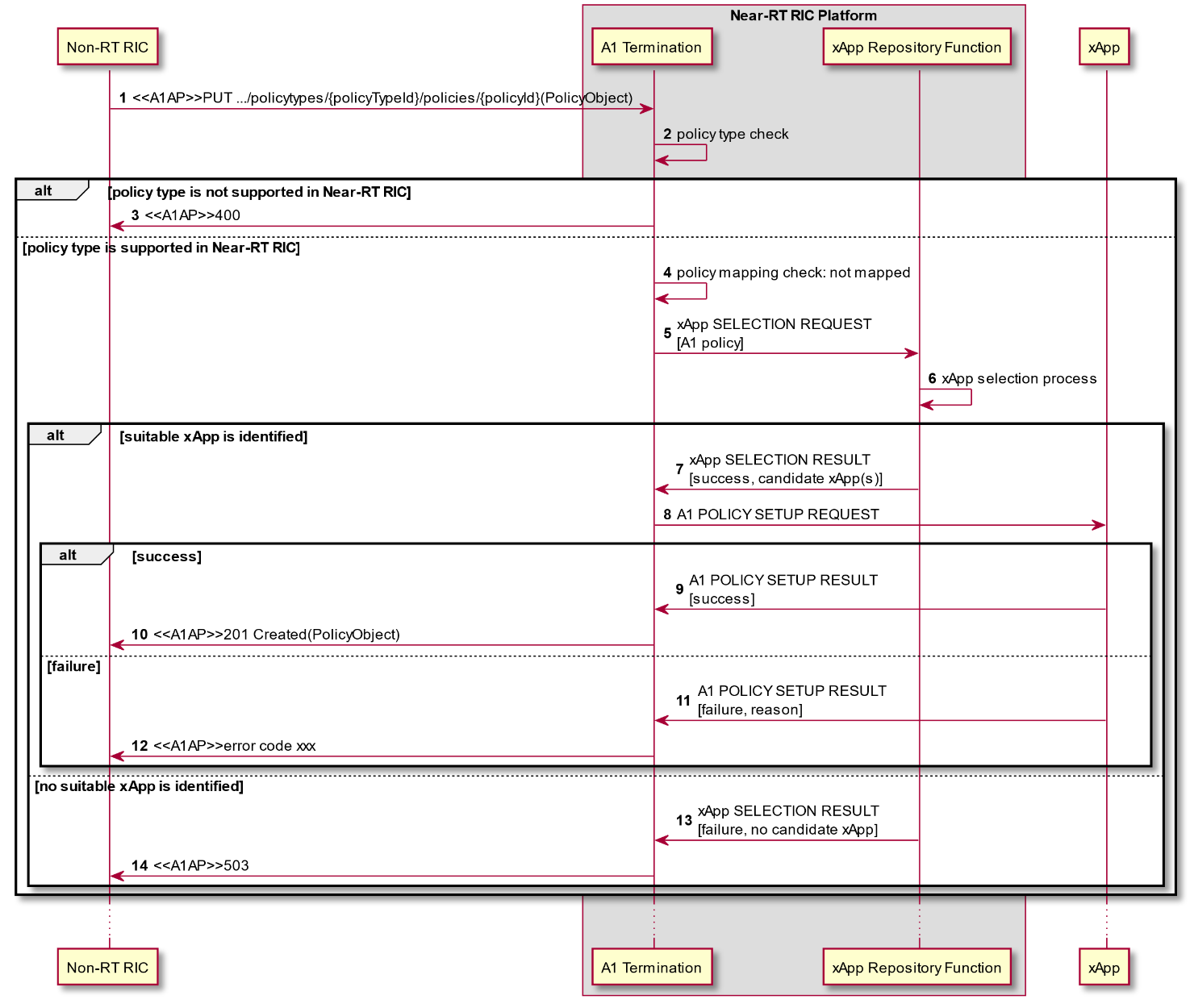
aot -> ric:<<A1AP>>503

end

end

...

@enduml



**Figure 9.2.2.1-1 A1 Policy Setup procedure**

#### 9.2.2.2 A1 Policy Update procedure

The purpose of A1 Policy Update procedure in Near-RT RIC is to support A1AP Update Single Policy procedure from Non-RT RIC.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | The update of a received and already activated A1 policy in Near-RT RIC. |
| Actors and Roles | * Non-RT RIC: originator of HTTP PUT request to update an A1 policy; * xApp: responsible for the enforcement of assigned A1 policies; * Near-RT RIC platform:   + A1 Termination: originator of A1 related API request to update a A1 policy. |
| Assumptions | * A1 policy types supported in Near-RT RIC are accessible by A1 termination; * A1 policies created in Near-RT RIC are accessible by A1 termination. |
| Pre conditions | * Near-RT RIC is running normally; * A1 interface is connected and activated. |
| Begins when | Non-RT RIC determines to trigger the update of one A1 policy towards the Near-RT RIC. |
| Step 1 (M) | Non-RT RIC requests Near-RT RIC to update an A1 policy. |
| Step 2 (M) | A1 Termination checks whether the received policy type is supported in Near-RT RIC or not. |
|  | [ALT] Step 3 is executed if the received policy type is not supported in Near-RT RIC: |
| Step 3 (M) | A “400” response is directly returned to Non-RT RIC. |
|  | [ELSE] Steps 4-9 are executed if the received policy type is supported in Near-RT RIC: |
| Step 4 (M) | A1 Termination then checks whether the received A1 policy is mapped to any xApp or not. |
| Step 5 (M) | (NOTE 1)If the received A1 policy is already mapped to one xApp, A1 Termination selects which as the target of **A1 related API: A1 POLICY UPDATE REQUEST** message. The message includes at least the corresponding A1 policy received from Non-RT RIC. |
|  | [ALT] Steps 6-7 are executed if the target xApp accomplishes update of the A1 policy: |
| Step 6-7 (M) | A success is notified to A1 termination in **A1 related API: A1 POLICY UPDATE RESULT** message, and eventually to Non-RT RIC. |
|  | [ELSE] Steps 8-9 are executed if the target xApp fails to update the A1 policy: |
| Step 8-9 (M) | A failure is notified to A1 termination in **A1 related API: A1 POLICY UPDATE RESULT** message, and eventually to Non-RT RIC. |
| NOTE 1: The create policy and update policy procedure in A1 interface has the identical HTTP PUT Request, so when a HTTP PUT Request is received, A1 Termination cannot decide whether it’s asked for create or update from a first glance. Only after further policy mapping check to find out the enforcement status of the received A1 policy, can the purpose of the received HTTP PUT request be identified. If the received A1 policy is already mapped to an xApp, A1 Termination triggers A1 Policy Update procedure as shown here. Otherwise, A1 Termination triggers A1 Policy Setup procedure. See 9.2.2.1 for details. | |

@startuml

skin rose

skinparam ParticipantPadding 3

skinparam BoxPadding 6

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

End box

Participant ap as “xApp”

autonumber

ric -> aot:<<A1AP>>PUT .../policytypes/{policyTypeId}/policies/{policyId}(PolicyObject)

aot<-aot: policy type check

alt policy type is not supported in Near-RT RIC

aot -> ric:<<A1AP>>400

else policy type is supported in Near-RT RIC

aot -> aot: policy mapping check: mapped

aot -> ap:A1 POLICY UPDATE REQUEST

alt success

ap -> aot:A1 POLICY UPDATE RESULT\n[success]

aot -> ric:<<A1AP>>200 OK(PolicyObject)

else failure

ap -> aot:A1 POLICY UPDATE RESULT\n[failure, reason]

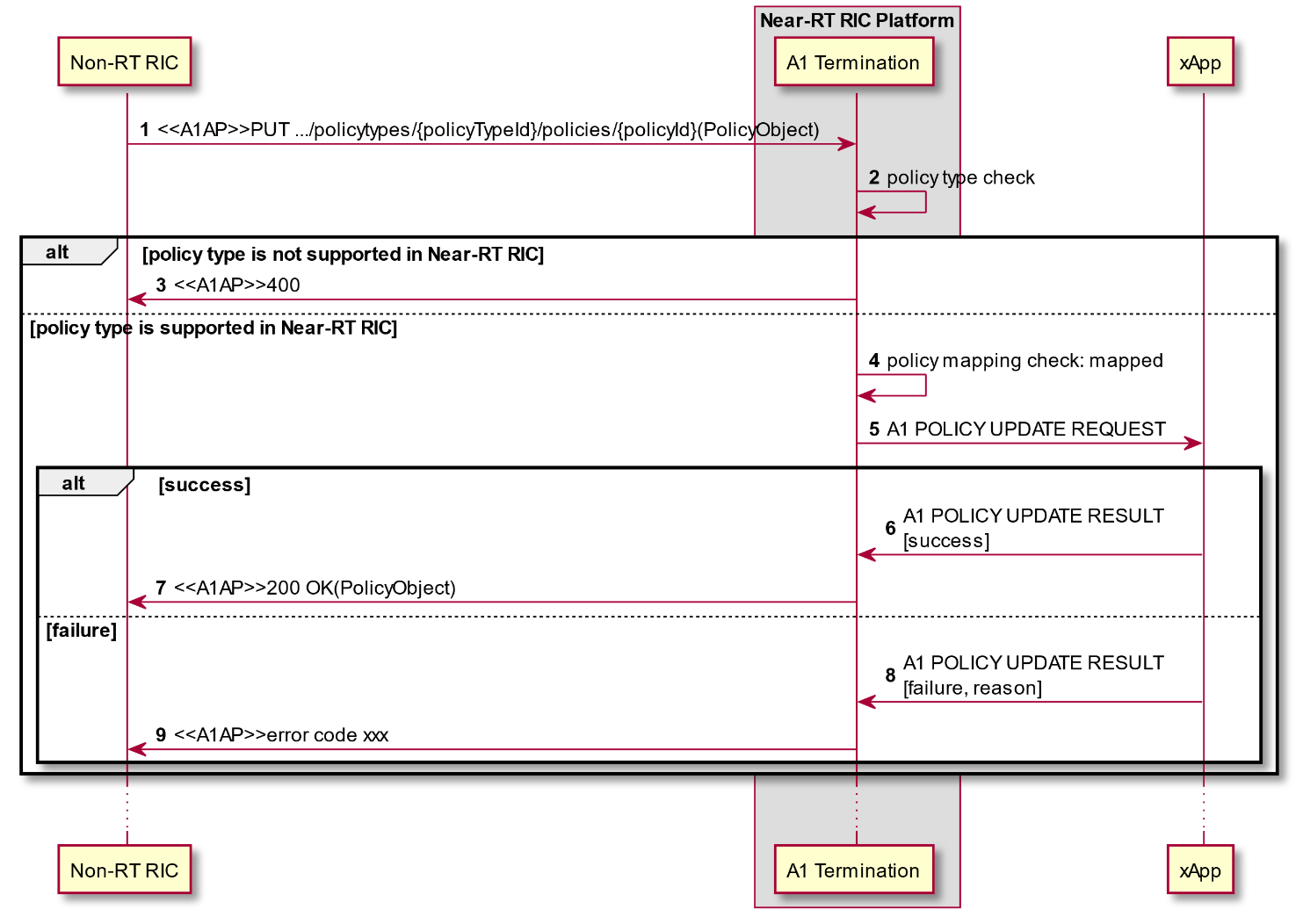
aot -> ric:<<A1AP>>error code xxx

end

end

...

@enduml



**Figure 9.2.2.2-1 A1 Policy Update procedure**

#### 9.2.2.3 A1 Policy Delete procedure

The purpose of A1 Policy Delete procedure is to support the A1AP Delete Policy procedure from Non-RT RIC.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | The deactivation of a received and already activated A1 policy in Near-RT RIC. |
| Actors and Roles | * Non-RT RIC: originator of HTTP DELETE request to deactivate an A1 policy; * xApp: responsible for the deactivation of created A1 policies; * Near-RT RIC platform:   + A1 Termination: originator of A1 related API request to delete an A1 policy. |
| Assumptions | * A1 policy types supported in Near-RT RIC are accessible by A1 termination; * A1 policies created in Near-RT RIC are accessible by A1 termination. |
| Pre conditions | * Near-RT RIC is running normally. * A1 interface is connected and activated. |
| Begins when | Non-RT RIC determines to trigger the deactivation of one A1 policy towards the Near-RT RIC. |
| Step 1 (M) | Non-RT RIC requests Near-RT RIC to delete an A1 policy. |
| Step 2 (M) | A1 Termination checks whether the received policy type is supported in Near-RT RIC or not. |
|  | [ALT] Step 3 is executed if the received policy type is not supported in Near-RT RIC: |
| Step 3 (M) | A “400” response is directly returned to Non-RT RIC. |
|  | [ELSE] Steps 4-10 are executed if the received policy type is supported in Near-RT RIC: |
| Step 4 (M) | A1 Termination then checks whether the received A1 policy is mapped to any xApp or not. |
|  | [ALT] Steps 5-9 are executed if the received A1 policy is already mapped to one xApp: |
| Step 5 (M) | A1 Termination selects which as the target of **A1 related API: A1 POLICY DELETE REQUEST** message. The message includes at least the corresponding A1 policy ID received from Non-RT RIC. |
|  | [ALT] Steps 6-7 are executed if the target xApp accomplishes deletion of the A1 policy: |
| Step 6-7 (M) | A success is notified to A1 termination in **A1 related API: A1 POLICY DELETE RESULT** message, and eventually to Non-RT RIC. |
|  | [ELSE] Steps 8-9 are executed if the target xApp fails to delete the A1 policy: |
| Step 8-9 (M) | A failure is notified to A1 termination in **A1 related API: A1 POLICY DELETE RESULT** message, and eventually to Non-RT RIC. |
|  | [ELSE] Step 10 is executed if the received A1 policy is not mapped to any xApp: |
| Step 10 (M) | A “503” response is directly returned to Non-RT RIC. |

@startuml

skin rose

skinparam ParticipantPadding 3

skinparam BoxPadding 6

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

End box

Participant ap as “xApp”

autonumber

ric -> aot:<<A1AP>>DELETE .../policytypes/{policyTypeId}/policies/{policyId}

aot<-aot: policy type check

alt policy type is not supported in Near-RT RIC

aot -> ric:<<A1AP>>400

else policy type is supported in Near-RT RIC

aot -> aot: policy mapping check

alt policy mapped

aot -> ap:A1 POLICY DELETE REQUEST

alt success

ap -> aot:A1 POLICY DELETE RESULT\n[success]

aot -> ric:<<A1AP>>204 No content

else failure

ap -> aot:A1 POLICY DELETE RESULT\n[failure, reason]

aot -> ric:<<A1AP>>error code xxx

end

else policy not mapped

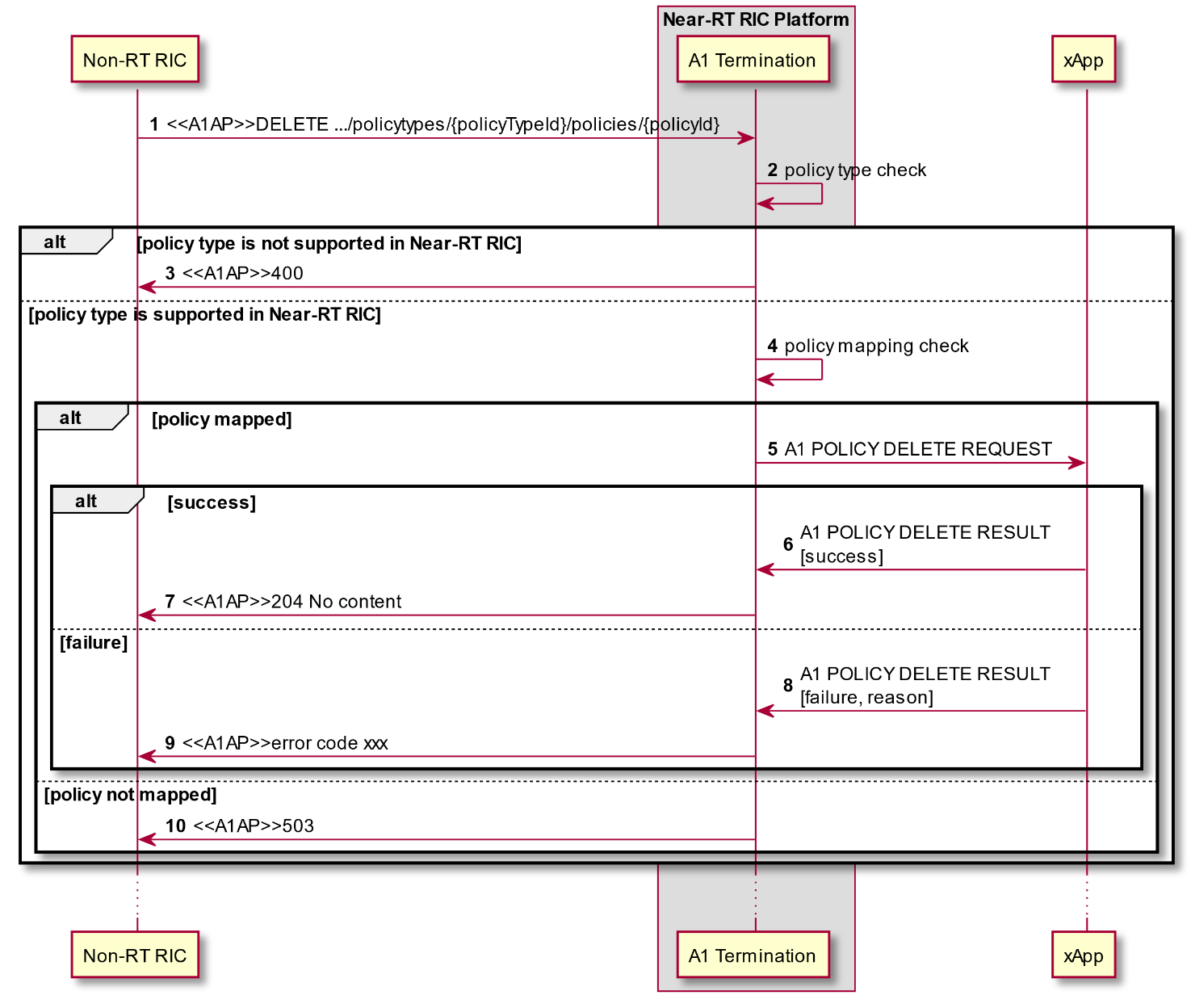
aot -> ric:<<A1AP>>503

end

end

...

@enduml



**Figure 9.2.2.3-1 A1 Policy Delete procedure**

#### 9.2.2.4 A1 Policy Query procedure

The purpose of A1 Policy Query procedure is to support the following A1AP procedures initiated by the Non-RT RIC:

* A1AP Query Single Policy procedure;
* A1AP Query All Policy Identifiers procedure;
* A1AP Query Policy Status procedure;
* A1AP Query All Policy Type Identifiers procedure;
* A1AP Query Single Policy Type procedure.

Table 9.2.2.4-1 provides the list of expected xApp response for each Near-RT RIC Platform Policy query. The query may come from Near-RT platform or from the Non-RT RIC.

**Table 9.2.2.4-1: Supported A1AP Policy Query procedures**

|  |  |  |
| --- | --- | --- |
| Query procedure | Platform Request | xApp Response |
| Query Single Policy | URI containing policyId | PolicyObject |
| Query All Policy Identifiers | URI containing policyTypeId | list of current policyId |
| Query Status | URI containing policyId | PolicyStatusObject |
| Query All Policy Type Identifiers | URI | list of currently supported PolicyTypeId |
| Query Single Policy Type | URI containing PolicyTypeId | PolicyTypeObject |

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | To obtain from an xApp information concerning A1 Policies, outcome depends on nature of Request:   * Query Single Policy: xApp provides PolicyObject for a given policyId; * Query All Policy Identifiers: xApp provides the list of current policyId for a given policyTypeId; * Query Status: xApp provides current PolicyStatusObject for a given policyId; * Query All Policy Type Identifiers: xApp provides list of currently supported PolicyTypeId; * Query Single Policy Type: xApp provides PolicyTypeObject of a given PolicyTypeId. |
| Actors and Roles | * Non-RT RIC: originator of query using HTTP GET request; * xApp: provides appropriate response corresponding to nature of request; * Near-RT RIC platform:   + A1 Termination: routes incoming A1AP request to one or more xApp, combines response from one or more xApp to build A1AP response;   + xApp Repository Function: provides A1T with list of appropriate xApps to handle request. |
| Assumptions | * xApp selection needed information is accessible by xApp Repository Function. xApp selection process is performed at xApp Repository Function; * A1 Termination may store information that was previously obtained from the xApp Repository Function; * xApp Repository Function may hold information related to Policy Query procedures. |
| Pre conditions | * Near-RT RIC is running normally; * A1 interface is connected and activated. |
| Begins when | Non-RT RIC determines need to trigger A1 query towards the Near-RT RIC. |
| Step 1 (M) | Non-RT RIC sends to Near-RT RIC one of the A1AP Policy Management service related Query procedures. |
| Step 2 [Informative] | A1 Termination inspects content of HTTP GET to determine nature of Query. |
| Step 3 [Informative] | A1 Termination checks whether the received A1 policy management service query is mapped to any xApp or not. |
|  | [OPT] Steps 4-7 may be executed if the received A1 policy management service query is not mapped to any xApp: |
| Step 4 {informative] | A1 Termination triggers xApp selection procedure to identify a suitable xApp by sending xAPP SELECTION REQUEST to xApp Repository Function. The message includes at least A1 policy query received from Non-RT RIC. |
| Step 5 [Informative] | xApp selection process is performed at xApp Repository Function. |
| Step 6-7 [Informative] | If one or more suitable xApps are identified, candidate xApps are notified to A1 Termination. Response may contain information held by xApp Repository Function concerning A1 Policy query. |
|  | [LOOP] Steps 8-11 are looped for each xApp selected by A1 Termination |
| Step 8 (O) | A1 Termination may send **A1 related API: A1 POLICY QUERY REQUEST** message. The message includes at least the corresponding A1 policy management service query resource URI received from Non-RT RIC. |
| Step 9-10 (O) | If the selected xApp(s) accomplishes the A1 policy management service query, a success is notified to A1 termination in **A1 related API: A1 POLICY QUERY RESULT** message. A1 Termination may forward result to xApp Repository Function for storage. |
| Step 11 (O) | If the selected xApp(s) fails to treat the A1 policy management service query, a failure is notified to A1 termination in **A1 related API: A1 POLICY QUERY RESULT** message. |
| Step 12 [Informative] | A1 Termination collects the responses from each xApp and/or from xApp Repository Function and consolidates into a response to Non-RT RIC. |
| Step 13 (M) | If A1 policy query response available, A1 Termination sends the appropriate A1AP Policy Management Service response to Non-RT RIC. |
| Step 14 (M) | If A1 policy query response not available, A1 Termination sends the appropriate A1AP Policy Management Service error to Non-RT RIC (A1AP [9] section 4.2.7.3). |
| Step 15 (O) | Depending upon the nature of the previous A1AP Policy Management Service query and the response from the Near-RT RIC, the Non-RT RIC may initiate a subsequent A1AP Policy Management Service query. |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 8

skinparam defaultFontSize 12

Participant NON as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant A1T as “A1 Termination”

Participant XRF as “xApp Repository Function”

End box

Collections XAPP as “xApp”

autonumber

NON -> A1T:<<A1AP>>GET (URI)

A1T->A1T: Inspect URI to determine nature of query

A1T->A1T: Query mapping check

opt Optional: A1 Termination consults xApp Repository Function

A1T -> XRF: xApp SELECTION REQUEST\n[A1 policy query, URI]

XRF->XRF : xApp selection process \nRecall stored information

alt Suitable xApp(s) identified

XRF -> A1T: xApp SELECTION RESULT\n[success, candidate xApp(s), stored information]

else No suitable xApp identified

XRF -> A1T: xApp SELECTION RESULT\n[failure, no candidate xApp]

end

end

loop For each xApp selected by A1 Termination

A1T -> XAPP: A1 POLICY QUERY REQUEST (URI)

alt Success

XAPP -> A1T: A1 POLICY QUERY RESULT\n[success, result]

A1T --> XRF: xApp Policy query information

else Failure

XAPP -> A1T: A1 POLICY QUERY RESULT \n[failure, reason]

end

end

A1T->A1T: Consolidate responses

alt Response available

A1T -> NON: <<A1AP>> 200 OK (response)

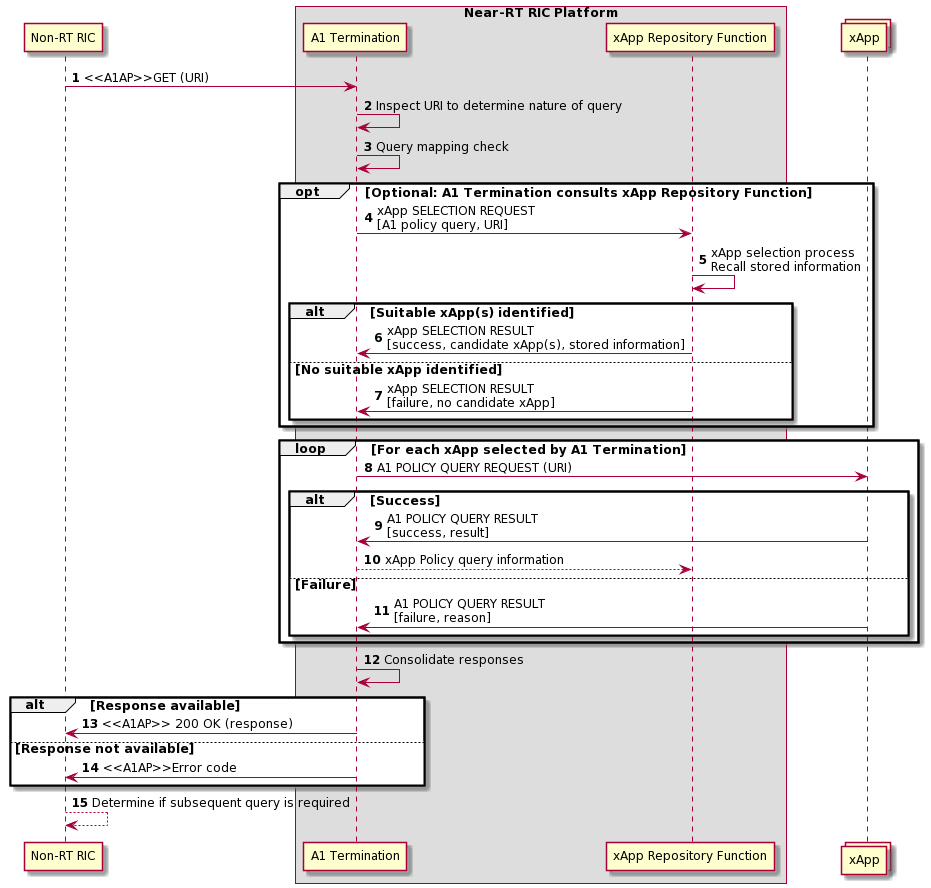
else Response not available

A1T ->NON: <<A1AP>>Error code

end

NON-->NON: Determine if subsequent query is required

@enduml



**Figure 9.2.2.4-1 A1 Policy Query procedure**

#### 9.2.2.5 A1 Policy Status Update procedure

The purpose of A1 Policy Status Update procedure in Near-RT RIC is to support the A1AP Policy Status Update procedure (A1AP [9] section 3.2.2.6) from the Near-RT RIC.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | The activation of a received but not activated A1 policy in Near-RT RIC. |
| Actors and Roles | * Non-RT RIC: receives the HTTP POST containing PolicyStatusObject; * xApp: originator of the PolicyStatusObject content to be sent to Non-RT RIC; * Near-RT RIC platform:   + A1 Termination: forwards the PolicyStatusObject from xApp using A1AP Policy Status Update procedure. |
| Assumptions | * A1 policy types supported in Near-RT RIC are accessible by A1 termination; * A1 policies created in Near-RT RIC are accessible by A1 termination; * xApp selection needed data is accessible by xApp Repository Function. xApp selection process is performed at xApp Repository Function. |
| Pre conditions | * Near-RT RIC is running normally; * A1 interface is connected and activated. |
| Begins when | xApp determines need to trigger the A1 policy status update to be sent to the Non-RT RIC. |
| Step 1 (M) | xApp sends **A1 related API: A1 POLICY STATUS UPDATE** (PolicyStatusObject information encoded as A1TD) message to A1 termination. |
|  | [ALT] Step 2 is executed if A1 Termination rejects request: |
| Step 2 (M) | A1 Termination sends A1 Related **POLICY STATUS UPDATE (Error)** to xApp. |
|  | [ELSE] Step 3-5 are executed if A1 Termination accepts request: |
| Step 3 (M) | A1 Termination forwards content within A1AP Policy Status Update message. |
| Step 4-5 (M) | Non-RT RIC sends A1AP acknowledgement, and A1 Termination forwards to xApp using **A1 Related POLICY STATUS UPDATE (Ack).** |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 8

skinparam defaultFontSize 12

Participant NON as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant A1T as “A1 Termination”

Participant XRF as “xApp Repository Function”

End box

Collections XAPP as “xApp”

autonumber

XAPP -> A1T: A1 POLICY STATUS UPDATE (PolicyStatusObject)

Alt If A1 Termination rejects message from xApp

A1T --> XAPP: A1 POLICY STATUS UPDATE (error)

Else IF A1 Termination accepts message from xApp

A1T -> NON: <<A1AP>>POST (PolicyStatusObject)

NON->A1T: <<A1AP>> 204 (No content)

A1T-> XAPP: A1 POLICY STATUS UPDATE (Ack)

End

@enduml

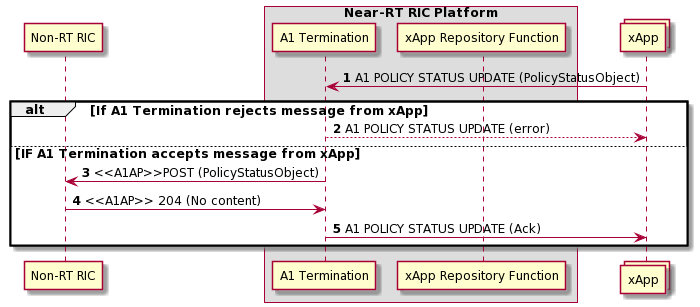


Figure 9.2.2.5-1 A1 Policy Status Update procedure

### 9.2.3 A1-EI related procedures

#### 9.2.3.1 A1 EI Query procedure

The purpose of A1 EI Query procedure is to support the following A1AP procedures initiated by the Near-RT RIC:

* Query EI Type Identifiers procedure;
* Query EI Type procedure;
* Query EI Job Identifiers procedure;
* Query EI Job procedure;
* Query EI Job Status procedure.

Table 9.2.3.1-1 provides the expected response from Near-RT RIC platform for each EI query request. The information may come from the platform or the Non-RT RIC.

**Table 9.2.3.1-1: Supported A1AP Query EI procedures**

|  |  |  |
| --- | --- | --- |
| Query procedure | xApp request | Platform Response |
| Query EI Type Identifiers | URI containing EiTypes | List of EiTypeId |
| Query EI Type | URI containing EiTypeId | List of EiTypeObject |
| Query EI Job Identifiers | URI containing EiJobs | List of current EiJobId |
| Query EI Job | URI containing EiJobId | EiJobObject |
| Query EI Job Status | URI containing EiJobId | EiJobStatusObjec |

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | To obtain from an xApp information concerning A1 Policies, outcome depends on nature of Request:   * Query EI Type Identifiers: Non-RT RIC provides list of EiTypeId for a given EiTypes; * Query EI Type: Non-RT RIC provides EiTypeObject for a given EiTypeId; * Query EI Job Identifiers: Non-RT RIC provides list of current EiJobId for a given EiJobs; * Query EI Job: Non-RT RIC provides EiJobObject for a given EiJobId; * Query EI Job Status: Non-RT RIC provides EiJobStatusObject for a given EiJobId with flag Status. |
| Actors and Roles | * Non-RT RIC: provides appropriate response corresponding to nature of request; * xApp: originator of query using HTTP GET request; * Near-RT RIC platform:   + A1 Termination: routes incoming A1 Related Query requests from xApp, routes A1AP response to xApp. |
| Assumptions | * A1 Termination is stateless and so does not store previous Query responses from Non-RT RIC |
| Pre conditions | * Near-RT RIC is running normally. * A1 interface is connected and activated. |
| Begins when | xApp determines need to trigger A1 EI query towards the Non-RT RIC. |
| Step 1 (M) | xApp sends **A1 Related EI QUERY REQUEST (URI)** to A1 Termination. |
|  | [ALT] Step 2 is executed if A1 Termination rejects the request: |
| Step 2 (O) | A1 Termination may reject request and send **A1 Related EI QUERY RESULT (error code)** to xApp. |
|  | [ELSE] Steps 3-8 are executed if A1 Termination accepts the request: |
| Step 3 (M) | A1 Termination builds appropriate A1AP EI Query message including URI and sends to non-RT RIC. |
| Step 4 [Informative] | Non-RT RIC inspects URI to determine nature of query and it is acceptable. |
|  | [ALT] Steps 5-6 are executed if Non-RT RIC accepts the query: |
| Step 5-6 (M) | Non-RT RIC builds appropriate response (EI information encoded using A1TD) and sends to Near-RT RIC, A1 Termination uses **A1 Related EI QUERY RESULT** to send response to xApp (with EI information encoded using A1TD). |
|  | [ELSE] Steps 7-8 are executed if Non-RT RIC rejects the query: |
| Step 7-8 (M) | Non-RT RIC builds appropriate response (error code) and sends to Near-RT RIC, A1 Termination used **A1 Related EI QUERY RESULT (error code)** to send response to xApp. |
| Step 9 (O) | Depending upon the nature of the previous A1AP EI query and the response from the Non-RT RIC, the xApp may initiate a subsequent A1AP EI query. |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 8

skinparam defaultFontSize 12

Participant NON as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant A1T as “A1 Termination”

Participant XRF as “xApp Repository Function”

End box

Collections XAPP as “xApp”

autonumber

XAPP -> A1T: A1 EI QUERY REQUEST (URI)

Alt EI query rejected by A1 Termination

A1T -> XAPP: A1 EI QUERY RESULT (error)

Else EI query accepted by A1 Termination

A1T -> NON: <<A1AP>>GET (URI)

NON->NON: Inspect URI to determine \nnature of query

alt EI query accepted

NON->A1T: <<A1AP>> 200 OK (EI information)

A1T-> XAPP: A1 EI QUERY RESULT (EI information)

else EI query rejected

NON->A1T: <<A1AP>>Error code

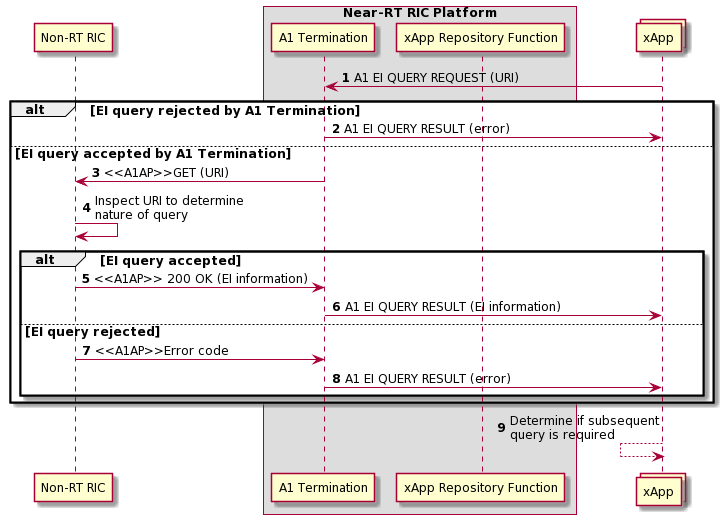
A1T->XAPP: A1 EI QUERY RESULT (error)

end

end

XAPP<--XAPP: Determine if subsequent \nquery is required

@enduml



**Figure 9.2.3.1-1 A1 EI Query procedure**

#### 9.2.3.2 A1-EI subcription setup procedure

The A1-EI subscription setup procedure enables xApp to setup a new A1-EI subscription toward Near-RT RIC Platform.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | To setup a new A1-EI subscription between xApp and Near-RT RIC Platform. |
| Actors and Roles | * Non-RT RIC: A1-EI producer. * xApp: requester of AI-EI subscription setup. * Near-RT RIC Platform:   + A1 Termination: handling of A1-EI subcription setup request from xApp.   + xApp Repository Function: check for xApp's A1-EI access permission. |
| Assumptions | * Created EiJobObjects in Near-RT RIC is readable by A1 Termination. * Non-RT RIC’s supported A1-EIs record in Near-RT RIC is readable by A1 Termination. * xApp subscribe multiple A1-EIs in integrated manner, i.e. subscribed A1-EIs indicated by **A1 related API: A1-EI SUBSCRIPTION SETUP REQUEST** message are dependent on each other, and only full success of all A1-EI subscriptions is viewed as message process success. * xApp's access permission to A1-EI is handled by xApp Reposiroty Function. |
| Pre-conditions | * Near-RT RIC Platform is running normal. * Near-RT RIC is connected to Non-RT RIC via A1 interface. |
| Begins when | xApp determines to request subscription of A1-EI(s). |
| Step 1 (M) | xApp requests Near-RT RIC Platform to set up an A1-EI subscription by sending **A1 related API: A1-EI SUBSCRIPTION SETUP REQUEST** message, with A1-EI subscription setup details included. |
| Step 2 [Informative] | A1 Termination checks with ARF whether the originating xApp is allowed to access all the requested A1-EI(s). |
|  | [ALT] Step 3 is executed if the originating xApp is not allowed to access some of the requested A1-EIs. |
| Step 3 (M) | A failure is responded to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION SETUP RESULT** message, possibly including appropriate reason. |
|  | [ELSE] Step 4 is executed if the originating xApp is allowed to access all the requested A1-EIs. |
| Step 4 [Informative] | A1 Termination checks whether all the requested A1-EIs is supported in Non-RT RIC. |
|  | [ALT] Step 5 is executed if some of the requested A1-EI(s) is NOT supported in Non-RT RIC. |
| Step 5 (M) | A failure is responded to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION SETUP RESULT** message, possibly including appropriate reason. |
|  | [ELSE] Step 6 is executed if all requested A1-EI(s) is supported in Non-RT RIC. |
| Step 6 [Informative] | A1 Termination checks whether created EiJobObject(s) satisfies the originating xApp’s request. |
|  | [ALT] Step 7 is executed if created EiJobObject(s) satisfies the originating xApp’s request. |
| Step 7 (M) | A1 Termination associates the identified created EiJobObject with the originating xApp, and responds success to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION SETUP RESULT** message. |
|  | [ELSE] Loop is executed if created EiJobObject(s) does not satisfy the originating xApp’s request. |
|  | Loop for each subscribed A1-EI that does not have EiJobObject association |
| Step 8 [Informative] | A1 Termination makes A1 interface EiJobObject either create or update decision. |
|  | [ALT] Step 9 is executed if A1 interface EiJobObject create decision is made. |
| Step 9 (M) | A1 Termination initiates create EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | [ELSE] Step 10 is executed if A1 interface EiJobObject update decision is made. |
| Step 10 (M) | A1 Termination initiates update EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | Loop end |
|  | [ALT] Step 11 is executed if A1 interface EiJobObject create/update process shots full success. |
| Step 11 (M) | A1 Termination responds success to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION SETUP RESULT** message. |
|  | [ELSE] Step 12 is executed if A1 interface EiJobObject create/update process shots not full success. |
| Step 12 (M) | A1 Termination responds failure to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION SETUP RESULT** message, possibly including appropriate reason. |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 8

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

Participant rf as “xApp Repository Function”

End box

Participant ap as “xApp”

autonumber

ap->aot:A1-EI SUBSCRIPTION SETUP REQUEST\n[A1-EI Subscription ID, A1-EI Subscription Setup Details]

aot<->rf:Check the originating xApp's\naccess permission to requested A1-EI(s)

alt The originating xApp is NOT allowed to access some of requested A1-EI(s)

aot->ap:A1-EI SUBSCRIPTION SETUP RESULT [Result{failure}, Reason]

else The originating xApp is allowed to access all requested A1-EI(s)

aot->aot:Checks for A1-EI(s) support in Non-RT RIC

alt Some of requested A1-EI(s) is NOT supported in Non-RT RIC

aot->ap:A1-EI SUBSCRIPTION SETUP RESULT [Result{failure}, Reason]

else All requested A1-EI(s) is supported in Non-RT RIC

aot->aot:Checks created EiJobObject(s) against\nthe originating xApp's request

alt Created EiJobObject(s) satisfies the originating xApp's request

aot->ap:A1-EI SUBSCRIPTION SETUP RESULT [Result{success}]

else Created EiJobObject(s) does not satisfy the originating xApp's request

loop for each requested A1-EI does not have created EiJobObject association

aot->aot:A1 interface EiJobObject\ncreate/update decide process

alt A1 interface EiJobObject create decision is made

aot<->ric:Create EI job procedure

else A1 interface EiJobObject update decision is made

aot<->ric:Update EI job procedure

end

end loop

alt EiJobObject(s) create/update full success

aot -> ap:A1-EI SUBSCRIPTION SETUP RESULT [Result{success}]

else EiJobObject(s) create/update NOT full success

aot -> ap:A1-EI SUBSCRIPTION SETUP RESULT [Result{failure}, Reason]

end

end

end

end

...

@enduml

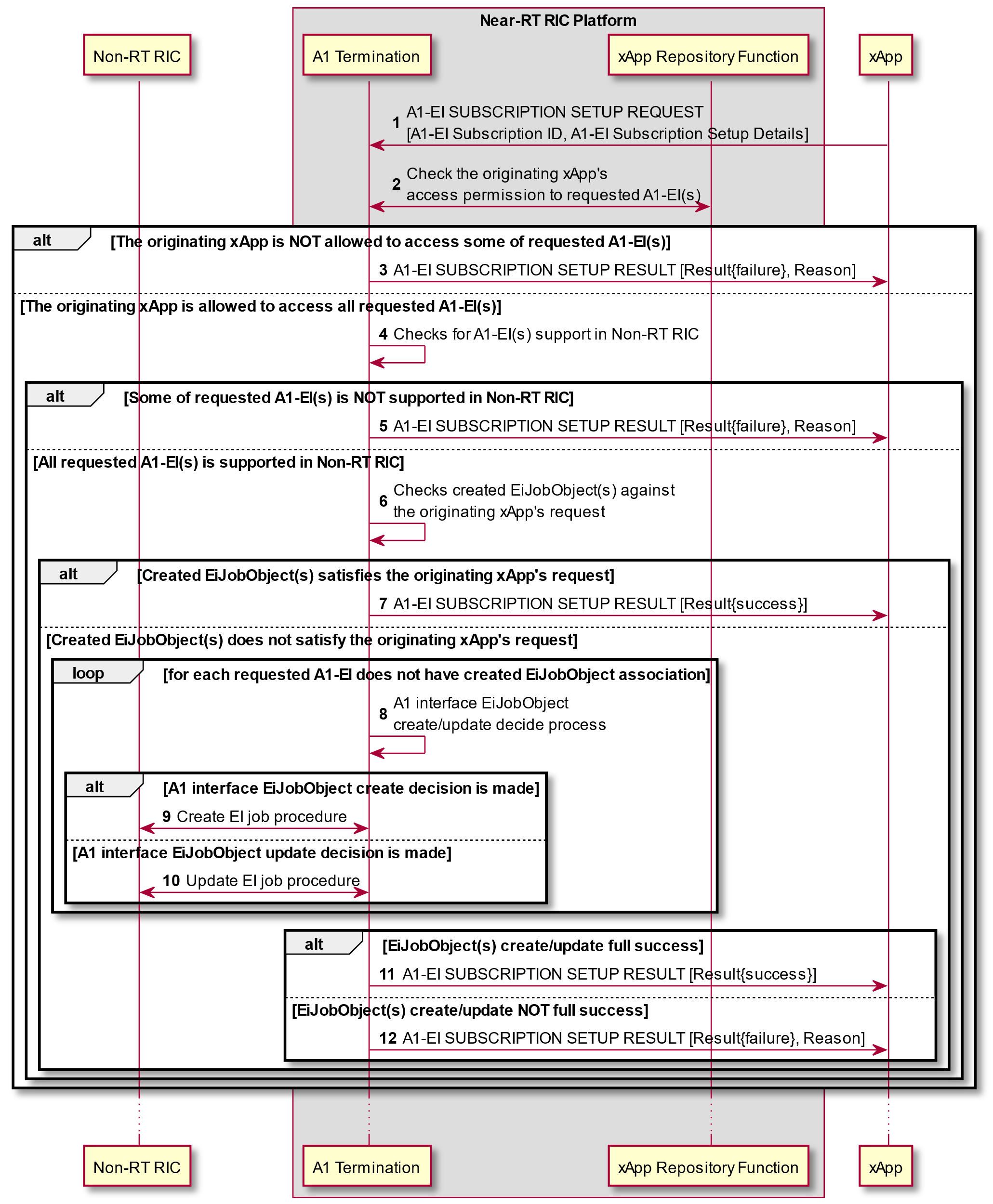


Figure 9.2.3.2-1 A1-EI subscription setup procedure

#### 9.2.3.3 A1-EI subcription update procedure

The A1-EI subscription update procedure enables xApp to update an existing A1-EI subscription toward Near-RT RIC Platform.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | To update an existing A1-EI subscription between xApp and Near-RT RIC Platform. |
| Actors and Roles | * Non-RT RIC: A1-EI producer. * xApp: requester of AI-EI subscription update. * Near-RT RIC Platform:   + A1 Termination: handling of A1-EI subscription update request from xApp.   + xApp Repository Function: check for xApp's A1-EI access permission. |
| Assumptions | * Created EiJobObjects in Near-RT RIC is readable by A1 Termination. * Non-RT RIC’s supported A1-EIs record in Near-RT RIC is readable by A1 Termination. * xApp subscribe multiple A1-EIs in integrated manner, i.e. subscribed A1-EIs indicated by **A1 related API: A1-EI SUBSCRIPTION SETUP REQUEST** message are dependent on each other, and only full success of all A1-EI subscriptions is viewed as message process success. * xApp's access permission to A1-EI is handled by xApp Reposiroty Function. |
| Pre-conditions | * Near-RT RIC Platform is running normal. * Near-RT RIC is connected to Non-RT RIC via A1 interface. |
| Begins when | xApp determines to request update of one A1-EI subscription. |
| Step 1 (M) | xApp requests Near-RT RIC Platform to update an A1-EI subscription by sending **A1 related API: A1-EI SUBSCRIPTION UPDATE REQUEST** message, with A1-EI subscription update details included. |
| Step 2 [Informative] | From **A1 related API: A1-EI SUBSCRIPTION UPDATE REQUEST** message, A1 Termination filters A1-EI(s) requested for 1) unsubscriing, and 2) subscribing (including new subscription needs creation and current subscription needs update). |
|  | For A1-EI(s) requested for unsubscribing, go step 3-6. |
| Step 3 [Informative] | A1 Termination deletes corresponding A1-EI subscription(s). |
|  | Loop for each requested A1-EI |
| Step 4 [Informative] | A1 Termination makes A1 interface EiJobObject either delete or update decision. |
|  | [ALT] Step 5 is executed if A1 interface EiJobObject delete decision is made. |
| Step 5 (M) | A1 Termination initiates delete EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | [ELSE] Step 6 is executed if A1 interface EiJobObject update decision is made. |
| Step 6 (M) | A1 Termination initiates update EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | Loop end |
|  | For A1-EI(s) requested for subscribing, go step 7-17. |
| Step 7 [Informative] | A1 Termination checks with ARF whether the originating xApp is allowed to access to the requested A1-EI(s). |
|  | [ALT] Step 8 is executed if the originating xApp is not allowed to access some of the requested A1-EI(s). |
| Step 8 (M) | A failure is responded to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION UPDATE RESULT** message, possibly including appropriate reason. |
|  | [ELSE] Step 9 is executed if the originating xApp is allowed to access all the requested A1-EI(s). |
| Step 9 [Informative] | A1 Termination checks whether the requested A1-EI(s) is supported in Non-RT RIC. |
|  | [ALT] Step 10 is executed if some of requested A1-EI(s) is NOT supported in Non-RT RIC. |
| Step 10 (M) | A failure is responded to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION UPDATE RESULT** message, possibly including appropriate reason. |
|  | [ELSE] Step 11 is executed if all requested A1-EI(s) is supported in Non-RT RIC. |
| Step 11 [Informative] | A1 Termination checks whether created EiJobObject(s) satisfies the originating xApp’s request. |
|  | [ALT] Step 12 is executed if created EiJobObject(s) satisfies the originating xApp’s request. |
| Step 12 (M) | A1 Termination associates the identified created EiJobObject with the originating xApp, and responds success to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION UPDATE RESULT** message. |
|  | [ELSE] Step13 is executed if created EiJobObject(s) does not satisfy the originating xApp’s request. |
|  | Loop for each requested A1-EI that does not have EiJobObject association |
| Step 13 [Informative] | A1 Termination makes A1 interface EiJobObject either create or update decision. |
|  | [ALT] Step 14 is executed if A1 interface EiJobObject create decision is made. |
| Step 14 (M) | A1 Termination initiates create EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | [ELSE] Step 15 is executed if A1 interface EiJobObject update decision is made. |
| Step 15 (M) | A1 Termination initiates update EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | Loop end |
|  | [ALT] Step 16 is executed if A1 interface EiJobObject create/update process shots full success. |
| Step 16 (M) | A1 Termination responds success to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION UPDATE RESULT** message. |
|  | [ELSE] Step 17 is executed if A1 interface EiJobObject create/update process shots not full success. |
| Step 17 (M) | A1 Termination responds failure to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION UPDATE RESULT** message, possibly including appropriate reason. |

@startuml

skin rose

skinparam ParticipantPadding 4

skinparam BoxPadding 8

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

Participant rf as “xApp Repository Function”

End box

Participant ap as “xApp”

autonumber

ap->aot:A1-EI SUBSCRIPTION UPDATE REQUEST\n[A1-EI Subscription ID, A1-EI Subscription Update Details]

aot->aot:(1)Filter A1-EI(s) requested for unsubscribing;\n(2)Filter A1-EI(s) requested for subscribing.

group For A1-EI(s) requested for unsubscribing

aot->aot:Delete corresponding A1-EI subscription(s)

loop For each requested A1-EI

aot->aot:A1 interface EiJobObject\ndelete/update decide process

alt A1 interface EiJobObject delete decision is made

aot<->ric:Delete EI job procedure

else A1 interface EiJobObject update decision is made

aot<->ric:Update EI job procedure

end

end loop

end group

group For A1-EI(s) requested for subscribing

aot<->rf:Check the originating xApp's\naccess permission to requested A1-EI(s)

alt The originating xApp is NOT allowed to access some of requested A1-EI(s)

aot->ap:A1-EI SUBSCRIPTION UPDATE RESULT [Result{failure}, Reason]

else The originating xApp is allowed to access all requested A1-EI(s)

aot->aot:Checks for A1-EI(s) support in Non-RT RIC

alt Some of requested A1-EI(s) is NOT supported in Non-RT RIC

aot->ap:A1-EI SUBSCRIPTION UPDATE RESULT [Result{failure}, Reason]

else All requested A1-EI(s) is supported in Non-RT RIC

aot->aot:Checks created EiJobObject(s) against\nthe originating xApp's request

alt Created EiJobObject(s) satisfies the originating xApp's request

aot->ap:A1-EI SUBSCRIPTION UPDATE RESULT [Result{success}]

else Created EiJobObject(s) does not satisfy the originating xApp's request

loop For each requested A1-EI does not have created EiJobObject association

aot->aot:A1 interface EiJobObject\ncreate/update decide process

alt A1 interface EiJobObject create decision is made

aot<->ric:Create EI job procedure

else A1 interface EiJobObject update decision is made

aot<->ric:Update EI job procedure

end

end loop

alt EiJobObject(s) create/update full success

aot -> ap:A1-EI SUBSCRIPTION UPDATE RESULT [Result{success}]

else EiJobObject(s) create/update NOT full success

aot -> ap:A1-EI SUBSCRIPTION UPDATE RESULT [Result{failure}, Reason]

end

end

end

end

end group

...

@enduml

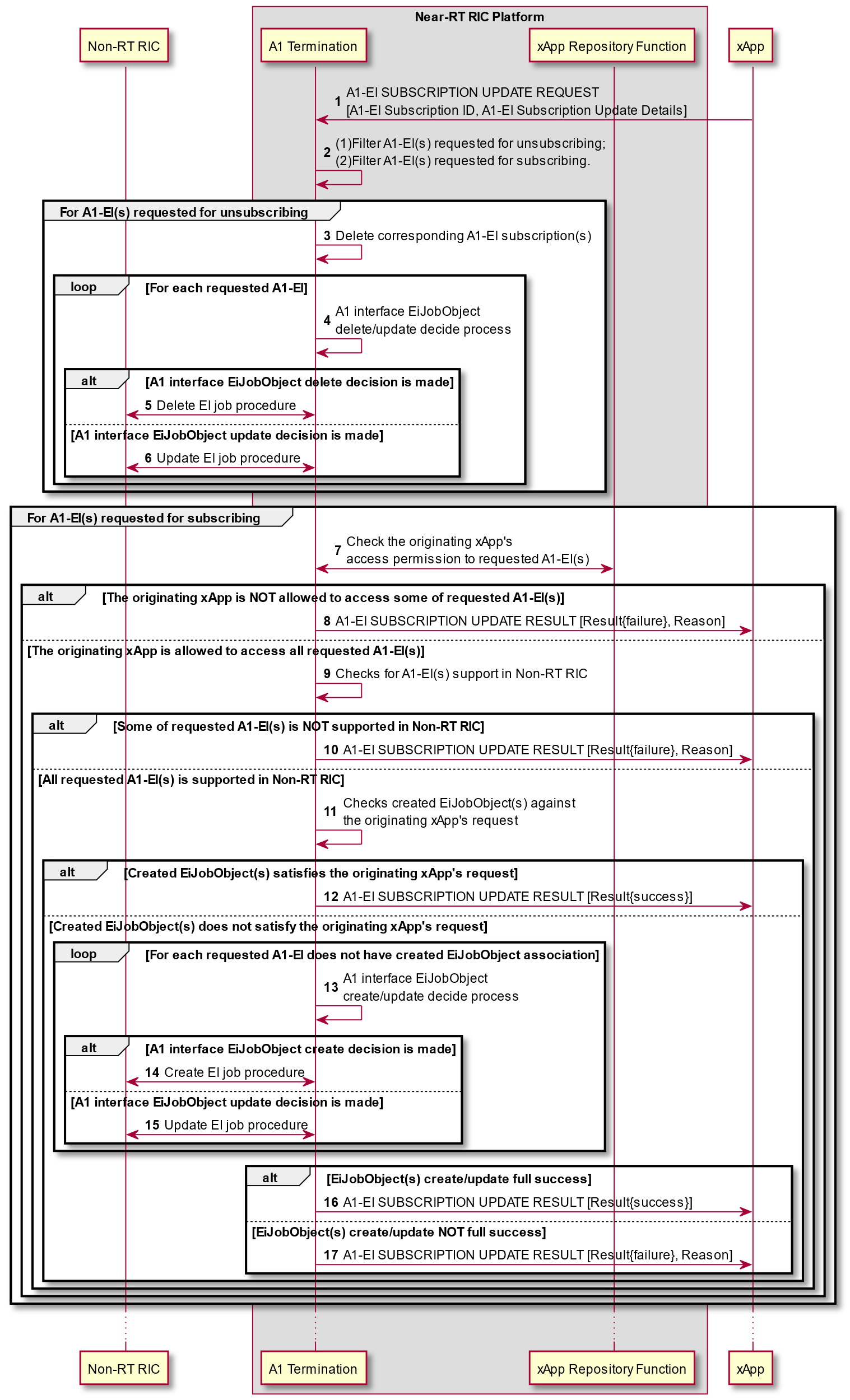


Figure 9.2.3.3-1 A1-EI subscription update procedure

#### 9.2.3.4 A1-EI subcription delete procedure

A1-EI subscription delete procedure allows xApp to delete an existing A1-EI subscription toward Near-RT RIC Platform.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | To delete an existing A1-EI subscription between xApp and Near-RT RIC Platform. |
| Actors and Roles | * Non-RT RIC: A1-EI producer. * xApp: requester of AI-EI subscription delete. * Near-RT RIC Platform:   + A1 Termination: handling of A1-EI subscription delete request from xApp. |
| Assumptions | * Created EiJobObjects in Near-RT RIC is readable by A1 Termination. * xApp's access permission to A1-EI subscription is handled by A1 Termination. |
| Pre-conditions | * Near-RT RIC Platform is running normal. * Near-RT RIC is connected to Non-RT RIC via A1 interface. |
| Begins when | xApp determines to request delete of one A1-EI subscription. |
| Step 1 (M) | xApp requests Near-RT RIC Platform to delete an A1-EI subscription by sending **A1 related API: A1-EI SUBSCRIPTION DELETE REQUEST** message, with A1-EI subscription ID included. |
| Step 2 [Informative] | A1 Termination checks whether the originating xApp is allowed to access to the requested A1-EI subscription. |
|  | [ALT] Step 3 is executed if the originating xApp is not allowed to access to the requested A1-EI subscription. |
| Step 3 (M) | A failure is responded to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION DELETE RESULT** message, possibly including appropriate reason. |
|  | [ELSE] Step 4 is executed if the originating xApp is allowed to access to the requested A1-EI subscription. |
| Step 4 [Informative] | A1 Termination finds the created EiJobObjects associated with the requested A1-EI subscription. |
|  | Loop for each created EiJobObjects associated with the requested A1-EI subscription |
| Step 5 [Informative] | A1 Termination makes A1 interface EiJobObject either delete or update decision. |
|  | [ALT] Step 6 is executed if A1 interface EiJobObject delete decision is made. |
| Step 6 (M) | A1 Termination initiates delete EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | [ELSE] Step 7 is executed if A1 interface EiJobObject update decision is made. |
| Step 7 (M) | A1 Termination initiates update EI job procedure towards Non-RT RIC and receives response from Non-RT RIC. |
|  | Loop end |
| Step 8 (M) | A1 Termination responds success to the originating xApp via **A1 related API: A1-EI SUBSCRIPTION DELETE RESULT** message. |

@startuml

skin rose

skinparam ParticipantPadding 3

skinparam BoxPadding 6

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

End box

Participant ap as “xApp”

autonumber

ap->aot:A1-EI SUBSCRIPTION DELETE REQUEST\n[A1-EI Subscription ID]

aot->aot:Check originating xApp's access\npermission to requested A1-EI subscription.

alt Originating xApp is not allowed to access to the requested A1-EI subscription

aot->ap:A1-EI SUBSCRIPTION DELETE RESULT\n[Result{failure}, Reason]

else Originating xApp is allowed to access to the requested A1-EI subscription

aot->aot:Finds the created EiJobObjects associated\nwith the requested A1-EI subscription

loop For each created EiJobObject associated\nwith the requested A1-EI subscription

aot->aot:A1 interface EiJobObject\ndelete/update decide process

alt A1 interface EiJobObject delete decision is made

aot<->ric:Delete EI job procedure

else A1 interface EiJobObject update decision is made

aot<->ric:Update EI job procedure

end

end loop

aot->ap:A1-EI SUBSCRIPTION DELETE RESULT\n[Result{success}]

end

...

@enduml

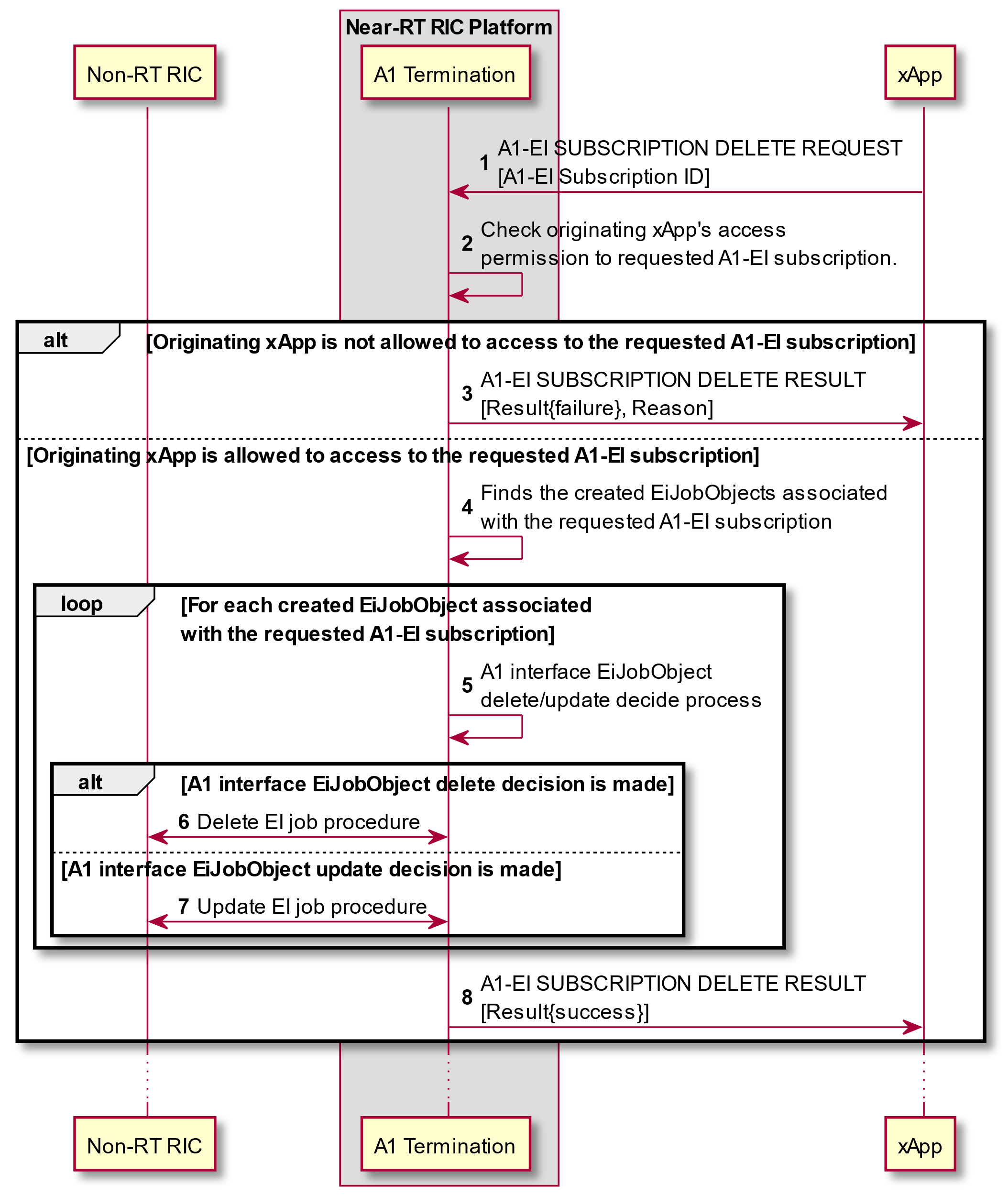


Figure 9.2.3.4-1 A1-EI subscription delete procedure

#### 9.2.3.5 A1-EI delivery procedure

The A1-EI delivery procedure allows xApp to be deliveried with subscribed A1-EI from Near-RT RIC Platform.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | The delivery of A1-EI from Near-RT RIC Platform. |
| Actors and Roles | * Non-RT RIC: A1-EI producer. * xApp: AI-EI consumer. * Near-RT RIC Platform:   + A1 Termination: handling of A1-EI delivery in Near-RT Platform. |
| Assumptions | * jobResultUri indicated by Non-RT RIC to target xApp mapping is handled by A1 Termination. |
| Pre-conditions | * Near-RT RIC Platform is running normal. * Near-RT RIC is connected to Non-RT RIC via A1 interface. |
| Begins when | Non-RT RIC has A1-EI produced for Near-RT RIC. |
| Step 1 (M) | Non-RT RIC sends a HTTP POST request to deliver the produced EiJobResultObject to Near-RT RIC Platform. |
| Step 2 [Informative] | A1 Termination maps jobResultUri indicated by Non-RT RIC to target xApps. |
|  | [ALT] If at least one target xApp is identified, go step 3-4. |
| Step 3 (M) | A1 Termination responds Non-RT RIC with status code "204". |
|  | Loop for each target xApp |
| Step 4 (M) | A1 Termination delivers the A1-EI to the target xApp by sending **A1 related API: A1-EI DELIVERY** message, with associated A1-EI subscription ID included. |
|  | Loop end |
|  | [ELSE] If NO target xApp is identified, go step 5. |
| Step 5 (M) | A1 Termination responds an appropriate error code and possibly error information to Non-RT RIC. |
| Step 6 (O) | A1 Termination initiates delete EI job procedure towards Non-RT RIC to delete the corresponding EI job and receives response from Non-RT RIC. |

@startuml

skin rose

skinparam ParticipantPadding 3

skinparam BoxPadding 6

skinparam defaultFontSize 12

Participant ric as “Non-RT RIC”

Box “Near-RT RIC Platform”

Participant aot as “A1 Termination”

End box

Participant ap as “xAPP”

autonumber

ric -> aot:<<A1AP>>POST {jobResultUri} (EiJobResultObject)

aot -> aot:Mapps jobResultUri to target xApps

alt Target xApps identified

aot -> ric:<<A1AP>>204 No Content

loop For each target xApp

aot -> ap:A1-EI DELIVERY\n[A1-EI Subscription ID, A1-EI]

end loop

else Target xApps NOT identified

aot -> ric:<<A1AP>>error code 4xx/5xx (error information)

aot <-> ric:Delete EI job procedure

end

...

@enduml

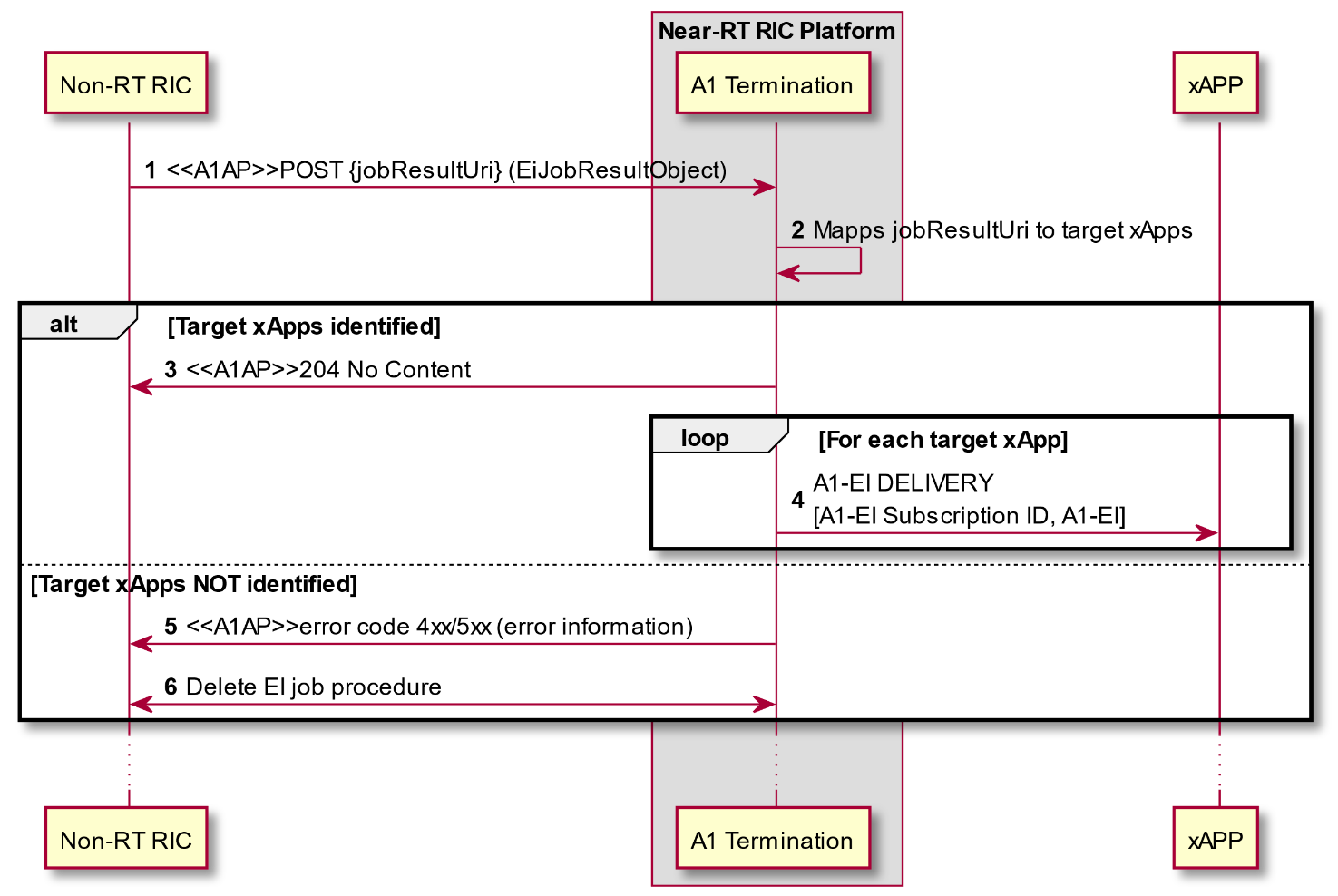


Figure 9.2.3.5-1 A1-EI delivery procedure

## 9.3 E2 Related API Procedures

### 9.3.1 Introduction

The following procedures are described in this section:

E2AP RIC Functional procedures:

* RIC Subscription using **E2 Related API: E2 Subscription** (request, reject, success, failure) and **E2 Related API: E2 Guidance** (request, response);
* RIC Subscription Delete using **E2 Related API: E2 Subscription Delete** (request, reject, success, failure);
* RIC Subscription Delete Required using **E2 Related API: E2 Subscription Delete Query** (required, accept, decline) and **E2 Related API: E2 Subscription Delete Notification** (Deleted);
* RIC Indication using **E2 Related API: E2 Indication** (push, failure);
* RIC Control using **E2 Related API: E2 Control** (request, reject, success, failure) and **E2 Related API: E2 Guidance** (request, response).

E2 Guidance related procedures:

* xApp initiated conflict mitigation using **E2 Related API: E2 Guidance** (request, response);
* xApp Subscription Management initiated conflict mitigation using **E2 Related API: E2 Guidance** (modification);
* Conflict mitigation related message monitoring using **E2 Related API: E2 Guidance** (modification);
* Conflict mitigation initiated conflict mitigation using **E2 Related API: E2 Guidance** (modification).

Other conflict mitigation related procedures are for future study.

### 9.3.2 RIC Functional Procedures

#### 9.3.2.1 E2 Subscription procedure

The purpose of the E2 Subscription procedure in the Near-RT RIC is to enable an xApp to request subscriptions for REPORT, INSERT and/or POLICY service(s) from E2 interface, and to ensure that only validated and non-duplicate subscriptions are maintained by the Near-RT RIC over the E2 interface to the E2 Node and that duplicated E2 Subscription Request messages from xApps are handled gracefully.

This procedure is based on the following assumptions:

* xApp may obtain guidance from Near-RT RIC Platform (i.e. Conflict Mitigation) to resolve potential conflicts and/or detect partial duplications prior to sending a E2 Subscription Request (see section 9.3.3.1);
* xApp has been configured with a trusted xApp ID;
* E2 related API routes E2 Subscription Request messages for a specific E2 Node from xApp towards an appropriate Near-RT RIC platform xApp Subscription Management function instance;
* xApp Subscription Management may recover from a platform database the list of all previous successful RIC Subscriptions towards the E2 Node listed in xApp’s E2 Subscription request and is able to detect duplications and recover E2 Node response messages;
* xApp Subscription Management may obtain guidance from Conflict Mitigation (see section 9.3.3.2);
* xApp Subscription Management decides initiation of appropriate E2AP procedure(s), if needed, to fulfil the E2 Subscription request.
* E2 Request ID is used on E2-related APIs to identify an active RIC subscription. Near-RT Platform maintains the 1:1 mapping between E2 Request ID, used on E2 related APIs, and RIC Request ID, used on E2 interface.
* xApp Subscription Management routes requests for E2AP procedure(s) to appropriate E2 Termination instance;
* E2 Termination systematically sends any result from E2AP procedure(s) to appropriate xApp Subscription Management instance;
* xApp Subscription management maintains a mapping of active Subscriptions (identified on E2 interface by E2 Node ID and RIC Request ID) of validated xApps. E2 Termination uses the mapping when sending E2 INDICATION messages to the correct xApp or xApps.

The procedure is initiated by an xApp using **E2 Related API: E2 Subscription** request for a specific E2 Node. The following outcomes are considered:

* Request fails following rejection by Near-RT RIC platform and an **E2 related API: E2 Subscription** response (Reject) is sent to xApp without E2 interface transaction. Response contains Cause (i.e. xApp not authorized to request specific subscription);
* Request towards a specific E2 Node is handled successfully following acceptance by Near-RT RIC platform but detected as duplicate and so acknowledged with an **E2 related API: E2 Subscription** response (Success) to xApp for a specific E2 Node without corresponding E2 interface transaction;
* Request towards a specific E2 Node is handled successfully following acceptance by Near-RT RIC Platform and acceptance by E2 Node and so acknowledged with an **E2 related API: E2 Subscription** response (Success) to xApp for a specific E2 Node following corresponding E2 interface transaction;
* Request towards a specific E2 Node fails following acceptance by Near-RT RIC Platform but rejected by E2 Node and so declined using an **E2 related API: E2 Subscription** response (Failure) to xApp for a specific E2 Node following corresponding E2 interface transaction. Response contains Cause (i.e. Request contents not accepted by E2 Node).

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | E2 Subscription API procedure from xApp initiation to E2 Node and response. |
| Actors and Roles | * xApp: Originator of E2 Subscription API request; * Near-RT RIC platform assumed to consist of:   + Database;   + xApp Subscription Management;   + Conflict Mitigation;   + E2 Termination. * E2 Node: RIC Subscription related procedure(s). |
| Assumptions | * xApp Subscription Management in Near-RT RIC platform has access to platform database; * For E2 Subscription API procedures, xApp Subscription Management is logically placed between xApp and E2 Termination; * Conflict Mitigation has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution; * Conflict Mitigation may initiate guidance towards other Platform Functions and/or xApp as an optional addition response to Guidance Request. |
| Pre-conditions | * E2 Node has active E2 interface to Near-RT RIC; * Near-RT RIC has recovered complete list of active RAN Functions on E2 Node and informed initiating xApp; * xApp has been authorized to issue E2 Subscription API Requests; * xApp has been authorized to request guidance from Conflict Mitigation; * xApp Subscription Management has been configured to permit E2 Subscription API requests from specific list of xApp; * E2 Termination has been configured to accept incoming requests for E2AP procedure(s) from xApp Subscription Management. |
| Begins when | xApp determines need to propose E2 Subscription API request for a specific E2 Node ID and defines message contents (target RAN Function ID, Event Trigger, Action List). |
| Step 1 (O) | xApp can request optional **E2 related API: E2 Guidance** from Conflict Mitigation (see section 9.3.3.1 for details). |
| Step 2 (M) | xApp sends **E2 related API: E2 Subscription** request with message contents (target RAN Function ID, Event Trigger, Action List) for a specific E2 Node. |
| Step 3 [Informative] | xApp Subscription Management, after optional guidance from Conflict Mitigation (see section 9.3.3.2 for details), accepts or rejects proposal from xApp for specific E2 Node. |
|  | [ALT] Step 4 is executed if xApp Subscription Management rejects the proposal: |
| Step 4 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription** response to xApp (Reject with Cause for specific E2 Node). |
|  | [ELSE] Steps 5-19 are executed if xApp Subscription Management accepts the proposal: |
| Step 5 [Informative] | xApp Subscription Management retrieves from a platform database the list of xApp associated with subscription for E2 Node target, RAN Function ID, Event Trigger, Action List. |
| Step 6 [Informative] | xApp Subscription Management checks the proposal against retrieved records for duplicate, and determines the subsequent operation. |
|  | [ALT] Steps 7-15 are executed if the subscription proposed by xApp necessitates E2AP procedure(s) (i.e. no record found for E2 Node ID with Subscription matching the same contents): |
| Step 7 [Informative] | xApp Subscription Management prepares request for appropriate E2AP procedure(s). |
| Step 8 [Informative] | xApp Subscription Management sends request for appropriate E2AP procedure(s) to E2 Termination, providing E2 Node ID. |
| Step 9 (M) | E2 Termination selects appropriate E2 interface and initiates appropriate E2AP procedure(s) to E2 Node. |
|  | [ALT] Steps 10-13 are executed if the E2AP procedure(s) in Step 9 succeeds: |
| Step 10 [Informative] | E2 Termination sends result from E2AP procedure(s) to xApp Subscription Management. |
| Step 11 [Informative] | For each Action in subscription corresponding to an Indication message, xApp Subscription Management requests E2 termination to add xApp ID to distribution list associated with E2 Node ID, RIC Request ID and Action ID. |
| Step 12 [Informative] | xApp Subscription Management creates corresponding record in the platform database. |
| Step 13 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription** response to xApp (Success, E2 Node ID, corresponding E2 Request ID(s) and E2 Action ID(s)). |
|  | [ELSE] Steps 14-15 are executed if the E2AP procedure(s) in Step 9 fails: |
| Step 14 [Informative] | E2 Termination sends result from E2AP procedure(s)to xApp Subscription Management. |
| Step 15 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription** response to xApp (Failure with Cause, E2 Node ID). |
|  | [ELSE] Steps 16-19 are executed if the proposal can be fulfilled with existing subscription(s) (i.e. one or more records found for E2 Node ID with Subscription matching the same contents): |
| Step 16 [Informative] | xApp Subscription Management extracts RIC Request ID from record. |
| Step 17 [Informative] | For each Action in subscription corresponding to an Indication message, xApp Subscription Management requests E2 termination to add xApp ID to distribution list associated with E2 Node ID, RIC Request ID and Action ID. |
| Step 18 [Informative] | xApp Subscription Management creates corresponding record in the platform database. |
| Step 19 (M) | xApp Subscription Management sends **E2 Related API: E2 Subscription** response to xApp (Success, E2 Node ID, corresponding E2 Request ID(s) and E2 Action ID(s)). |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

database “Database” as DB

Participant “Conflict Mitigation” as ConMit

Collections “xApp Subscription Management” as SubM

Collections “E2 Termination” as E2T

endbox

Collections “E2 Node” as ran

xApp <--> ConMit: 1: (E2 related API) E2 Guidance (Request/Response)

xApp ->SubM: 2: (E2 related API) E2 Subscription (Request, E2 Node ID)

SubM <--> ConMit: 3.1: Guidance (Request/Response)

SubM -> SubM: 3.2: Accept/Reject decision

Alt xApp Subscription Management function reject

SubM -> xApp: 4: (E2 related API) E2 Subscription (Reject, E2 Node ID, Cause)

Else xApp Subscription Management function accept

SubM <-> DB: 5: Fetch records (Subscription)

SubM -> SubM: 6: Detect duplication

alt E2AP procedure(s) needed

SubM -> SubM: 7: Prepare appropriate E2AP procedure(s)

SubM -> E2T: 8: Request for appropriate E2AP procedure(s)

E2T <-> ran: 9: <<E2>> Appropriate E2AP procedure(s)

alt E2AP procedure(s) succeeds

E2T -> SubM: 10: Result from E2AP procedure(s)(Success)

SubM <-> E2T: 11: Distribution list (Add)

SubM <-> DB: 12: Store record (Subscription)

SubM -> xApp: 13: (E2 related API) E2 Subscription (Success, E2 Node ID, E2 Request ID(s), E2 Action ID(s))

else E2AP procedure(s) fails

E2T -> SubM: 14: Result from E2AP procedure(s)(Failure)

SubM -> xApp: 15: (E2 related API) E2 Subscription (Failure, E2 Node ID, Cause)

end

else Existing subscription(s) can be reused

SubM -> SubM: 16: Extract RIC Request ID from record

SubM <-> E2T: 17: Distribution list (Add)

SubM <-> DB: 18: Store record (Subscription)

SubM -> xApp: 19: (E2 related API) E2 Subscription (Success, E2 Node ID, E2 Request ID(s), E2 Action ID(s))

end

end

@enduml

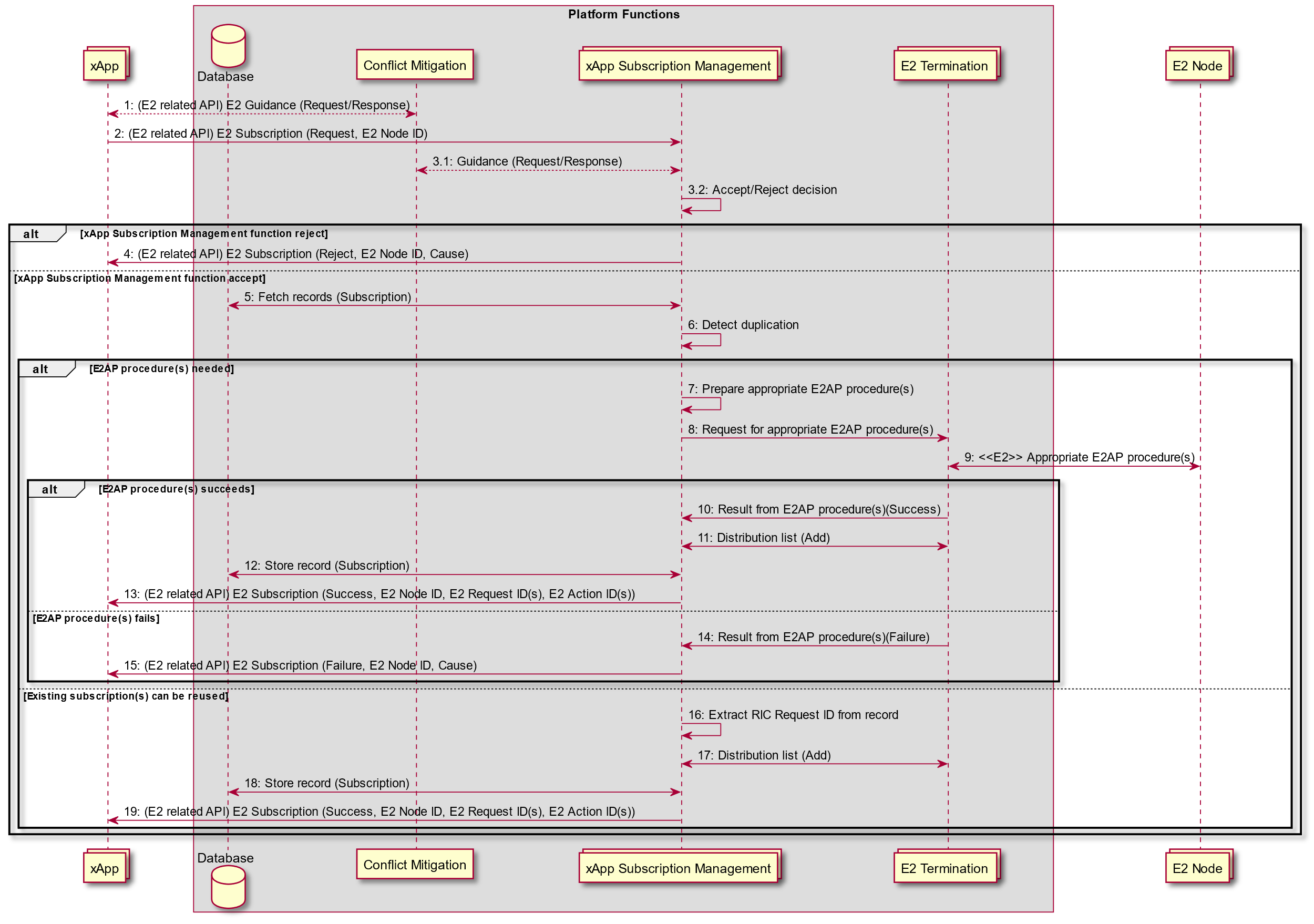


Figure 9.3.2.1-1: E2 Subscription API procedure

#### 9.3.2.2 E2 Subscription Delete procedure

The purpose of the E2 Subscription Delete procedure in the Near-RT RIC is to ensure that a) only validated RIC Subscription Delete Request messages are issued by the Near-RT RIC over the E2 interface to the E2 Node; b) deletion requests for duplicated Subscription Requests are handled gracefully.

This procedure is based on the following assumptions:

* E2 related API routes E2 Subscription Delete request for a specific E2 Node from xApp towards an appropriate xApp Subscription Management instance;
* xApp has been configured with a trusted xApp ID;
* xApp Subscription Management may recover from the platform database the list of all active subscriptions towards the E2 Node listed in xApp’s E2 Subscription Delete request and is able to detect when and if the delete request shall result in a corresponding E2 message;
* xApp Subscription Management routes the subscription delete requests to an appropriate E2 termination instance;
* E2 Termination systematically forwards any received RIC Subscription Delete Response or RIC Subscription Delete Failure to an appropriate xApp Subscription Management instance;
* xApp Subscription management maintains a mapping of active subscriptions (identified on E2 interface by E2 Node ID and RIC Request ID) of the validated xApps. E2 Termination uses the mapping when sending RIC INDICATION messages to the correct xApp;

The procedure initiated by an xApp uses **E2 Related API: E2 Subscription Delete** to request subscription deletion for a specific E2 Node. The following outcomes are considered:

* Request fails following rejection by xApp Subscription Management and an **E2 Related API: E2 Subscription Delete** response (Reject) is sent to xApp without E2 interface transaction (i.e. xApp not previously associated with validated subscription);
* Request succeeds following acceptance by xApp Subscription Management but other subscriber(s) detected for the subscription, and an **E2 Related API: E2 Subscription Delete** response (Success) is sent to xApp without corresponding E2 transaction;
* Request succeeds following acceptance by xApp Subscription Management and accepted by E2 Node, and an **E2 Related API: E2 Subscription Delete** response (Success) is sent to xApp following corresponding E2 transaction;
* Request fails following acceptance by xApp Subscription Management but rejected by E2 Node, and an **E2 Related API: E2 Subscription Delete** response (Failure) is sent to xApp following corresponding E2 interface transaction (i.e. RIC Request ID is not known by E2 Node);

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | E2 Subscription Delete procedure from xApp initiation to E2 Node and response. |
| Actors and Roles | * xApp: Originator of E2 Subscription Delete request; * Near-RT RIC platform:   + Database;   + xApp Subscription Management;   + E2 Termination. * E2 Node: RIC Subscription Delete procedure. |
| Assumptions | * xApp Subscription Management in Near-RT RIC has access to platform database; * For E2 Subscription Delete procedure, xApp Subscription Management is logically placed between xApp and E2 Termination. |
| Pre conditions | * E2 Node has active E2 interface to Near-RT RIC; * Near-RT RIC has recovered complete list of active RAN Functions on E2 Node and informed initiating xApp; * xApp Subscription Management has been configured to permit RIC Subscription Delete requests from xApp; * E2 Termination has been configured to accept incoming RIC subscription delete requests from xApp Subscription Management. |
| Begins when | xApp determines need to request E2 Subscription Delete for a specific E2 Node ID and E2 Request ID. |
| Step 1 (M) | xApp sends **E2 related API: E2 Subscription Delete** request for a specific E2 Node and E2 request ID. |
| Step 2 [Informative] | xApp Subscription Management retrieves list of xApp associated with subscription for E2 Node and E2 Request ID. |
| Step 3 [Informative] | xApp Subscription Management confirms that xApp proposing E2 Subscription Delete has an active subscription. |
| Step 4 [Informative] | xApp Subscription Management accepts or rejects proposal from xApp. |
|  | [ALT] Step 5 is executed if xApp Subscription Management rejects the proposal: |
| Step 5 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription Delete** response to xApp (Reject with Cause for specific E2 Node and E2 Request ID). |
|  | [ELSE] Steps 6-19 are executed if xApp Subscription Management accepts the proposal. |
| Step 6 [Informative] | xApp Subscription Management counts the number of retrieved records to determine whether or not xApp proposing Subscription Delete is the only associated xApp. |
|  | [ALT] Steps 7-16 are executed if the xApp proposing Subscription Delete is the only xApp associated with the subscription (i.e. a given RIC Request ID): |
| Step 7 [Informative] | xApp Subscription Management sends RIC Subscription Delete Request to E2 Termination, providing E2 Node ID and RIC Request ID. |
| Step 8 (M) | E2 Termination selects appropriate E2 interface and sends **RIC Subscription Delete Request** (RIC Request ID) to E2 Node. |
|  | [ALT] Steps 9-13 are executed if E2 Node accepts Subscription Delete: |
| Step 9 (M) | E2 Node responds with **RIC Subscription Delete Response.** |
| Step 10 [Informative] | E2 Termination forwards received message to xApp Subscription Management. |
| Step 11 [Informative] | xApp Subscription Management requests E2 termination to delete distribution list(s) associated with E2 Node ID and RIC Request ID. |
| Step 12 [Informative] | xApp Subscription Management deletes corresponding record in the platform database. |
| Step 13 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription Delete** response to xApp (Success). |
|  | [ELSE] Steps 14-16 are executed if E2 Node rejects Subscription Delete: |
| Step 14 (M) | E2 Node responds with **RIC Subscription Delete Failure** (with Cause). |
| Step 15 [Informative] | E2 Termination forwards received message to xApp Subscription Management. |
| Step 16 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription Delete** response to xApp (Failure with Cause). |
|  | [ELSE] Steps 17-19 are executed if the xApp proposing Subscription Delete is one of a number of xApp associated with the subscription (i.e. a given RIC Request ID): |
| Step 17 [Informative] | xApp Subscription Management requests E2 termination to remove xApp from distribution list(s) associated with E2 Node ID and RIC Request ID. |
| Step 18 [Informative] | xApp Subscription Management deletes corresponding record in the platform database. |
| Step 19 (M) | xApp Subscription Management sends **E2 related API: E2 Subscription Delete** response to xApp (Success). |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “xApp” as xApp

box “Platform functions”

Database “Database” as DB

Collections “xApp Subscription Management” as SubM

Collections “E2 Termination” as E2T

endbox

Collections “E2 Node” as ran

xApp ->SubM: 1: (E2 related API) E2 Subscription Delete (Request, E2 Node ID, E2 Request ID)

SubM <-> DB: 2: Fetch records (Subscription)

SubM -> SubM: 3: Confirm that xApp had previously subscribed \n(i.e. record found for xApp ID)

SubM -> SubM: 4: Accept/Reject decision

Alt xApp Subscription Management function reject

SubM -> xApp: 5: (E2 related API) E2 Subscription Delete (Reject, E2 Node ID, E2 Request ID, Cause)

Else xApp Subscription Management function accept

SubM -> SubM: 6: Count valid records

alt Single xApp record

SubM -> E2T: 7: RIC subscription delete request (E2 Node ID, RIC Request ID)

E2T -> ran: 8: <<E2>> RIC SUBSCRIPTION DELETE REQUEST (RIC Request ID)

alt E2 Node accepts delete request

ran -> E2T: 9: <<E2>> RIC SUBSCRIPTION DELETE RESPONSE

E2T -> SubM: 10: RIC subscription delete response

SubM <-> E2T: 11: Distribution list (Delete)

SubM <-> DB: 12: Delete record

SubM -> xApp: 13: (E2 related API) E2 Subscription Delete (Success, E2 Node ID, E2 Request ID)

else E2 Node rejects delete request

ran -> E2T: 14: <<E2>> RIC SUBSCRIPTION DELETE FAILURE

E2T -> SubM: 15: RIC subscription delete failure

SubM -> xApp: 16: (E2 related API) E2 Subscription Delete (Failure, E2 Node ID, E2 Request ID, Cause)

End

Else Multiple xApp records

SubM <-> E2T: 17: Distribution list (Remove)

SubM -> DB: 18: Delete record

SubM -> xApp: 19: (E2 related API) E2 Subscription Delete (Success, E2 Node ID, E2 Request ID)

end

end

@enduml



Figure 9.3.2.2-1: E2 Subscription Delete procedure, initiated by an xApp

#### 9.3.2.2A E2 Subscription Delete Query procedure and E2 Subscription Delete Notification procedure

The purpose of the E2 Subscription Delete Query procedure and E2 Subscription Delete Notification Procedure in the Near-RT RIC is to ensure that deletion request of the existing subscriptions from an E2 Node is handled properly.

The two procedures are based on the following assumptions:

* E2 Termination forwards any received RIC Subscription Delete Required from an E2 Node to an appropriate xApp Subscription Management instance. The xApp Subscription Management may recover from the platform database the list of xApps associated with the subscription removal request from an E2 Node and, if decides to accept, uses the E2 related API: E2 Subscription Delete Notification (Deleted) to delete subscriptions toward xApps. Prior to determining whether or not to accept the subscription delete required request, the xApp Subscription Management may send E2 related API: E2 Subscription Delete Query (required) to each xApp with an active subscription, each xApp may respond with E2 related API: E2 Subscription Delete (accept) or E2 related API: E2 Subscription Delete (decline).

The procedures may be initiated by the xApp Subscription Management, when RIC Subscription Delete Required is received from an E2 Node. The xApp Subscription Management may use the **E2 Related API: E2 Subscription Delete Query (required)** to consult xApps. The following outcomes are considered:

* xApp accepts request to delete subscription and sends E2 Related API: E2 Subscription Delete Query (accept) to accept subscription deletion.
* xApp declines request to delete subscription and sends E2 Related API: E2 Subscription Delete Query (decline) to inform xApp Subscription Management.

In both cases, the xApp Subscription Management may decide whether to delete the subscription.

The xApp Subscription Management may also decide whether to delete the subscription without consulting xApps.

If the xApp Subscription Management decides to delete the subscription, it uses **E2 Related API: E2 Subscription Delete Notification** (Deleted) to inform each xApp for the eventual outcome.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | RIC subscription removal initiated from an E2 Node. |
| Actors and Roles | * xApp: Recipient of E2 Subscription Delete query and notification; * Near-RT RIC platform:   + Database;   + xApp Subscription Management;   + E2 Termination. * E2 Node: Originator of RIC Subscription Delete Required. |
| Assumptions | * xApp Subscription Management in Near-RT RIC has access to platform database; * For E2 Subscription Delete procedure, xApp Subscription Management is logically placed between xApp and E2 Termination; |
| Pre conditions | * E2 Node has active E2 interface to Near-RT RIC; * Near-RT RIC has recovered complete list of active RAN Functions on E2 Node and informed xApps; * E2 Termination has been configured to accept incoming RIC subscription removal request from an E2 Node; * xApp Subscription Management has been configured to permit RIC Subscription removal request from an E2 Node. |
| Begins when | E2 Node determines a need to request removal of some RIC subscriptions that has been subscribed toward itself from Near-RT RIC. |
| Step 1 (M) | E2 Node sends **RIC Subscription Delete Required** (a list of RIC Request IDs and RAN Function IDs with cause). |
| Step 2  [Informative] | E2 Termination forwards received message to xApp Subscription Management. |
|  | [LOOP] Steps 3-15 are looped for each Subscription requested to be deleted: |
| Step 3 [Informative] | xApp Subscription Management retrieves list of xApps associated with RIC subscription requested to be deleted from the E2 Node. |
|  | [OPT] Steps 4-6 may be executed. For each xApp with RIC Subscription requested to be deleted: |
| Step 4 (O) | xApp Subscription Management sends **E2 related API: E2 Subscription Delete** **Query** Required to xApp (with E2 Node ID, E2 Request ID and Cause) to request xApp advice on required subscription deletion. |
| Step 5 (O) | If xApp accepts, xApp responds with **E2 related API: E2 Subscription Delete** **Query** Accept to xApp Subscription Management (with E2 Node ID, E2 Request ID). |
| Step 6 (O) | If xApp rejects, xApp responds with **E2 related API: E2 Subscription Delete** **Query** Decline to xApp Subscription Management (with E2 Node ID, E2 Request ID). |
| Step 7 (M) | xApp Subscription Management takes decision whether to accept/reject RIC subscription delete. |
|  | [OPT] Steps 8-17 are executed if xApp Subscription Management accepts: |
| Step 8 [Informative] | xApp Subscription Management sends RIC Subscription Delete Request to E2 Termination (with E2 Node ID and RIC Request ID). |
| Step 9 (M) | E2 Termination selects appropriate E2 interface and sends **RIC Subscription Delete Request** (RIC Request ID) to E2 Node. |
|  | [ALT] Steps 10-14 are executed if E2 Node accepts Subscription Delete: |
| Step 10 (M) | E2 Node responds with **RIC Subscription Delete Response.** |
| Step 11 [Informative] | E2 Termination forwards received message to xApp Subscription Management. |
| Step 12 [Informative] | xApp Subscription Management requests E2 termination to delete distribution list associated with E2 Node ID and RIC Request ID. |
| Step 13 [Informative] | xApp Subscription Management deletes record in database; |
| Step 14 (M) | For each concerned xApp, xApp Subscription Management sends **E2 related API: E2 Subscription Delete** **Notification** (with E2 Node ID, E2 Request ID, Cause) to inform xApp that subscription has been deleted. |
|  | [ELSE] Steps 15-17 are executed if E2 Node rejects Subscription Delete: |
| Step 15 (M) | E2 Node responds with **RIC Subscription Delete Failure** (with Cause). |
| Step 16 [Informative] | E2 Termination forwards received message to xApp Subscription Management. |
| Step 17 [Informative] | xApp Subscription Management decides subsequent processing (e.g., retry RIC Subscription Delete). The failure outcome is not needed to inform the xApp. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “xApp” as xApp

box “Platform functions”

Database “Database” as DB

Collections “xApp Subscription Management” as SubM

Collections “E2 Termination” as E2T

endbox

Collections “E2 Node” as ran

autonumber

ran -> E2T: <<E2>> RIC SUBSCRIPTION DELETE REQUIRED

E2T -> SubM: RIC Subscription Delete Required

Loop For each RIC Request ID

SubM <--> DB: Fetch records (RIC Request ID, E2 Node ID)

opt Optional

Loop For each xApp with E2 Request ID

SubM -> xApp: (E2 related API) E2 Subscription Delete Query (Required, E2 Node ID, E2 Request ID, Cause)

alt xApp accepts delete required

xApp ->SubM: (E2 related API) E2 Subscription Delete Query (Accept, E2 Node ID, E2 Request ID)

else xApp rejects delete required

xApp ->SubM: (E2 related API) E2 Subscription Delete Query (Decline, E2 Node ID, E2 Request ID)

end

end

end

SubM -> SubM: Accept/Reject decision

opt xApp Subscription Management function accept

SubM -> E2T: RIC Subscription delete request

E2T -> ran: <<E2>> RIC SUBSCRIPTION DELETE REQUEST

alt E2 Node accepts delete request

ran -> E2T: <<E2>> RIC SUBSCRIPTION DELETE RESPONSE

E2T -> SubM: RIC Subscription delete response

SubM <-> E2T: Distribution list (Delete)

SubM <-> DB: Delete record

loop For each xApp with E2 Request ID

SubM -> xApp: (E2 related API) E2 Subscription Delete Notification (Deleted, E2 Node ID, E2 Request ID, Cause)

end

else E2 Node rejects delete request

ran -> E2T: <<E2>> RIC SUBSCRIPTION DELETE FAILURE

E2T -> SubM: RIC subscription delete failure

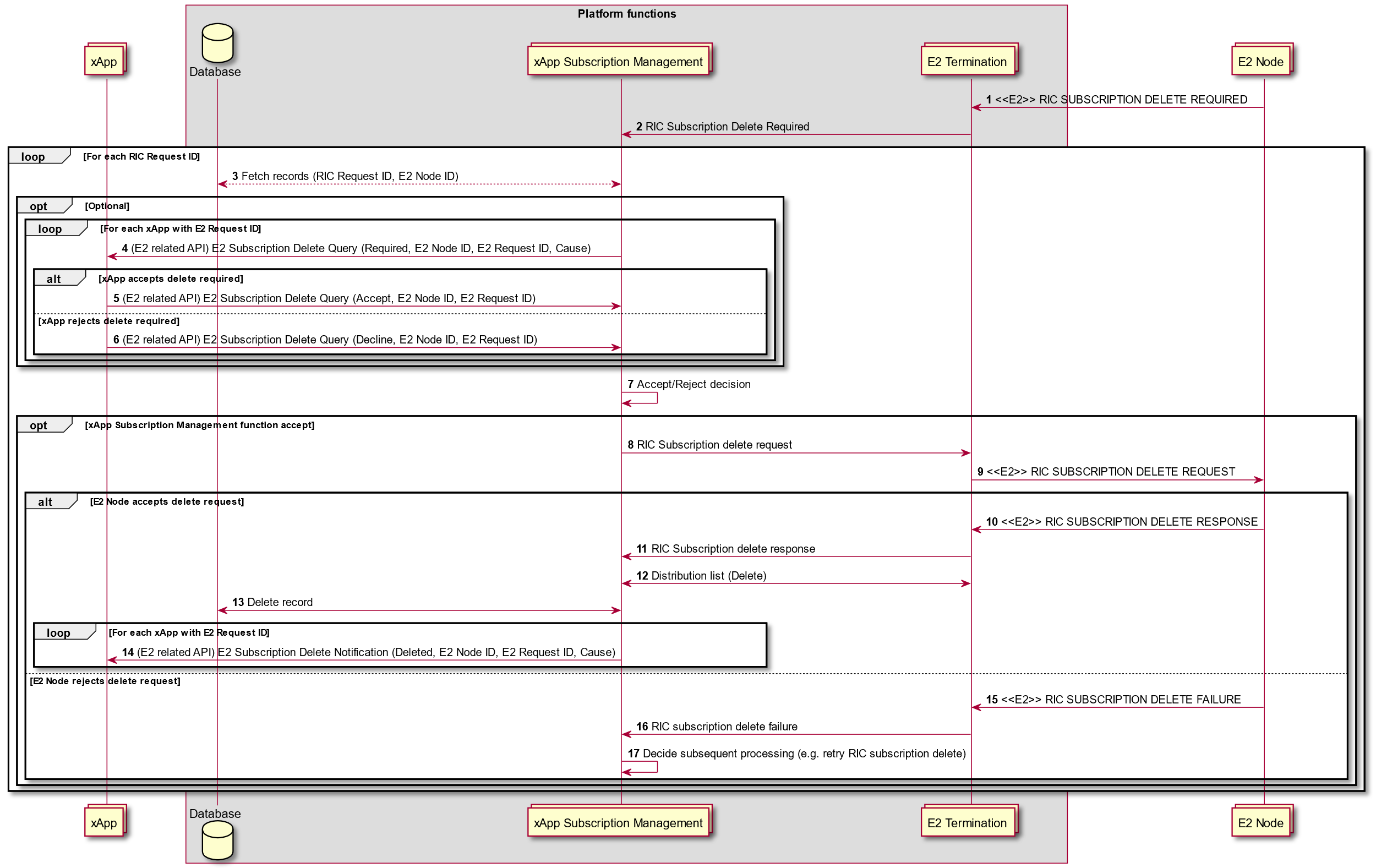
SubM -> SubM: Decide subsequent processing (e.g. retry RIC subscription delete)

end

end

end

@enduml



**Figure 9.3.2.2A-1: E2 Subscription Delete Query and E2 Subscription Notification procedures**

#### 9.3.2.3 E2 Indication procedure

The purpose of the E2 Indication procedure in the Near-RT RIC is to ensure delivery of RIC INDICATION messages to one or more validated xApps.

This procedure is based on the following assumptions:

* E2 Termination maintains a list for each E2 Node of validated xApps and the corresponding assigned E2 Request ID and E2 Action ID(s) associated with an E2 node ID, RIC REQUEST ID and RIC Action ID(s);
* E2 related API ensures delivery of E2 Indication messages from E2 termination to all validated xApp;
* Messaging infrastructure may support distribution of messages to multiple destinations.

The following outcomes are considered:

* Message is received, using E2 related API: E2 Indication (push), and successfully processed by xApp;
* Message is received, using E2 related API: E2 Indication (push), but an error is detected by the xApp and so declined using an E2 related API: E2 Indication (failure) from xApp for a specific E2 Node, E2 Request ID, E2 Action ID and E2 Indication SN. Response contains Cause (i.e. Request contents not accepted by xApp).

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | E2 Indication procedure from E2 Node initiation to xApp reception. |
| Actors and Roles | * xApp: Originator of RIC subscription request; * Near-RT RIC platform:   + Database;   + xApp Subscription Management;   + E2 Termination. * E2 Node: Originator of RIC Indication procedure. |
| Assumptions | * For RIC Indication procedures, E2 Termination uses E2 related API to directly send received messages to one or more xApp. |
| Pre conditions | * E2 Node has active E2 interface to Near-RT RIC; * Near-RT RIC has recovered complete list of active RAN Functions on E2 Node and informed initiating xApp; * E2 Termination has distribution list associating Indication messages with xApps. |
| Begins when | E2 Node creates RIC Indication Message (RIC Request ID, RAN Function ID, RIC Action ID, optional RIC Indication SN, Indication Type, Indication Header, Indication Message, optional Call Process ID). |
| Step 1 (M) | E2 Node sends **RIC Indication** message to E2 termination. |
| Step 2 [Informative] | If not available, E2 Termination fetches distribution list from platform database for E2 Node ID, RIC REQUEST ID, RAN Function ID, RIC Action ID. |
|  | [ALT] Step 3a is executed if messaging infrastructure in Near-RT RIC supports distribution of messages to list of destinations: |
| Step 3a (M) | E2 Termination sends **E2 related API: E2 Indication** (push, E2 Node ID, E2 Request ID, E2 Action ID, E2 Indication SN, Message) to list of xApps. |
|  | [ELSE] Step 3b is executed if messaging infrastructure in Near-RT RIC only supports distribution of messages to single destinations: |
| Step 3b (M) | E2 Termination sends **E2 related API: E2 Indication** (push, E2 Node ID, E2 Request ID, E2 Action ID, E2 Indication SN, Message) to each xApp in list. |
| Step 4 (O) | If an error is detected in the received message, the xApp may send **E2 related API: E2 Indication** (failure, E2 Node ID, E2 Request ID, E2 Action ID, E2 Indication SN, Cause). |
| Step 5 (O) | The Near-RT RIC may send an appropriate E2AP procedure message |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “xApp” as xApp

box “Platform functions”

Database "Database" as DB

Collections “E2 Termination” as E2T

endbox

Collections “E2 Node” as ran

ran -> E2T: <<E2>> 1: RIC INDICATION

E2T <-> DB: 2: Fetch distribution list

Alt Messaging system supports message distribution

E2T -> xApp: 3a: (E2 related API) E2 Indication (push)

Else Messaging system supports individual messages only

Loop For all xApp in distribution list

E2T -> xApp: 3b: (E2 related API) E2 Indication (push)

End

end

xApp --> E2T: 4: (E2 related API) E2 Indication (failure, Cause)

Opt

E2T --> ran: 5: <<E2AP>> E2AP Procedure

end

@enduml

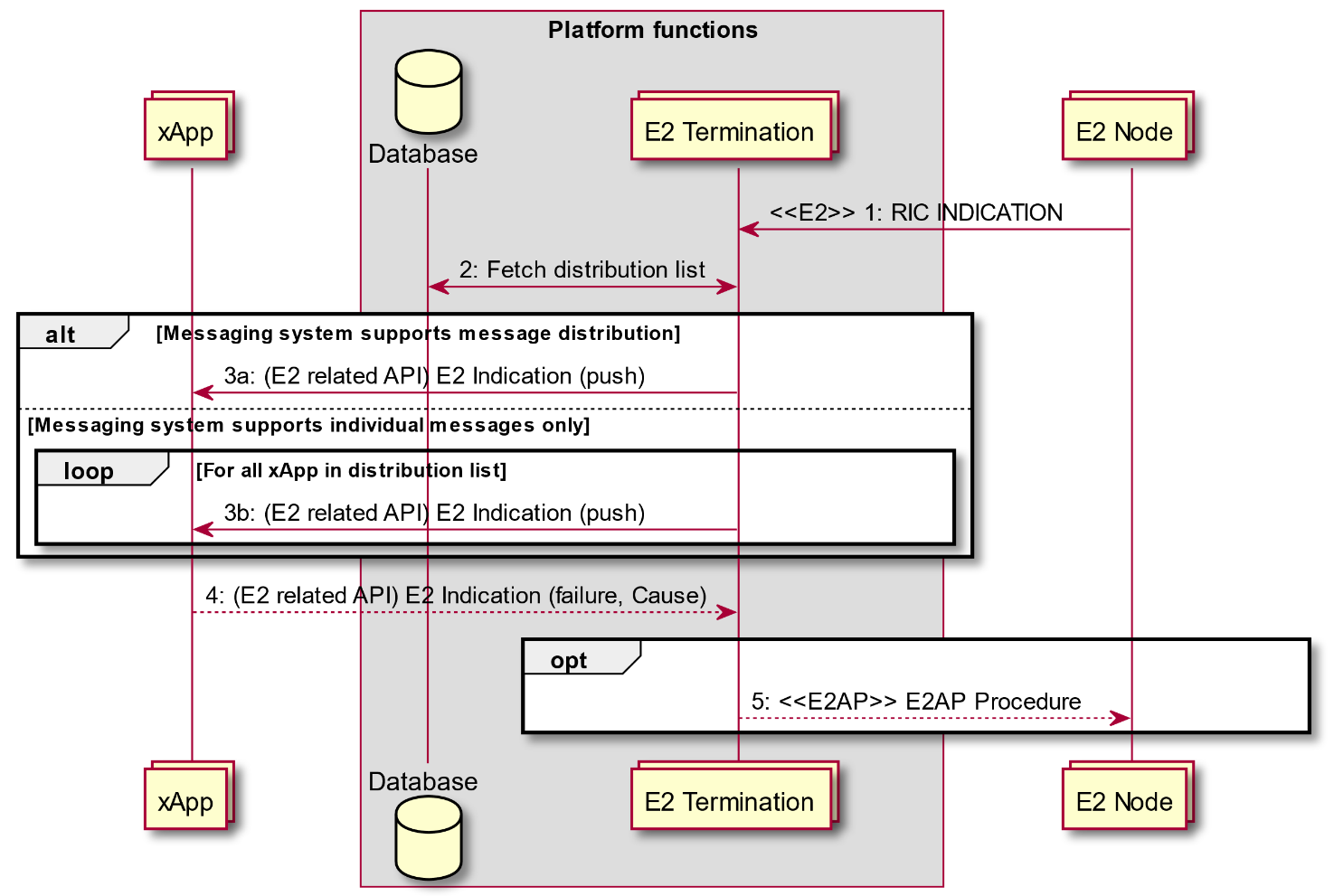


Figure 9.3.3-1: E2 Indication procedure

#### 9.3.2.4 E2 Control procedure

The purpose of the E2 Control procedure in the Near-RT RIC is to ensure that only authorized xApp may initiate RIC Control Request messages issued by the Near-RT RIC over the E2 interface to the E2 Node.

This procedure is based on the following assumptions:

* xApp may obtain guidance from Conflict Mitigation to resolve potential conflicts prior to sending a E2 Control Request (see section 9.3.3.1);
* xApp has been configured with a trusted xApp ID;
* E2 related API may be configured to route E2 Control Request messages for a E2 node from xApp either towards Conflict Mitigation for acceptance (see section 9.3.3.4) or directly towards an appropriate E2 Termination instance;
* E2 related API ensures that only authorized xApp may send E2 Control Request messages to appropriate E2 Termination instance;
* E2 related API ensures that E2 Termination systematically forwards any received RIC Control Response or RIC Control Failure to appropriate xApp.

The procedure is initiated by an xApp using E2 Related API: E2 Control request to send a request of a RIC Control for an E2 Node. The following outcomes are considered:

* Request successful following acceptance by Near-RT RIC platform and accepted by E2 Node and so an **E2 Related API: E2 Control** response (Success with outcome) is sent to xApp following corresponding E2 transaction;
* Request fails following acceptance by Near-RT RIC platform but rejected by E2 Node and so an **E2 Related API: E2 Control** response (Failure with cause) is sent to xApp following corresponding E2 interface transaction (i.e. Request contents is are compliant);
* Request rejected by Near-RT Platform and so an **E2 Related API: E2 Control** response (Reject) is sent to xApp (i.e. rejected by Conflict Mitigation).

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | E2 Control procedure from xApp initiation to E2 Node and response. |
| Actors and Roles | * xApp: Originator of E2 Control request; * Near-RT RIC platform:   + E2 Termination. * E2 Node: RIC Control procedure. |
| Assumptions | * For E2 Control procedures, xApp may directly send messages to E2 Termination; * Conflict Mitigation has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution. |
| Pre-conditions | * E2 Node has active E2 interface to Near-RT RIC; * Near-RT RIC has recovered complete list of active RAN Functions on E2 Node and informed initiating xApp; * xApp has been authorized to send RIC Control requests for a specific scope (E2 Node list, RAN Function, etc.); * xApp has been authorized to request guidance from Conflict Mitigation; * E2 Termination and E2 related API has been configured to accept incoming E2 Control requests from authorized xApp and forward either to Conflict Mitigation or directly to E2 Termination. |
| Begins when | xApp determines need to propose RIC Control procedure for a E2 Node and defines message contents (RAN Function ID, Call process ID, Control Header, Control message). |
| Step 1 (O) | xApp may request **E2 related API: E2 Guidance** from Conflict Mitigation (see section 9.3.3.1 for details) |
|  |  |
|  | [ALT] Step 2a is executed if **E2 related API: E2 Control** request message is routed directly to E2 Termination: |
| Step 2a (M) | xApp sends **E2 related API: E2 Control** request with message contents (RAN Function ID, Call process ID, Control Header, Control message) for a E2 Node, to E2 Termination. |
|  | [ELSE] Step 2b and 3-5 are executed if routed to Conflict Mitigation (see section 9.3.3.4 for details): |
| Step 2b (M) | xApp sends **E2 related API: E2 Control** request with message contents (RAN Function ID, Call process ID, Control Header, Control message) for a E2 Node, to Conflict Mitigation (see section 9.3.3.4 for details). |
| Step 3 [Informative] | Conflict Mitigation processes request. |
|  | [ALT] Step 4 is executed if Conflict Mitigation accepts proposed E2 Control message. |
| Step 4 [Informative] | Message is forwarded to appropriate E2 Termination instance. |
|  | [ELSE] Step 5 is executed if Conflict Mitigation rejects proposed E2 Control message: |
| Step 5 (M) | **E2 related API: E2 Control** response (Reject with Cause) message is sent to xApp. |
| Step 6 [Informative] | E2 Termination receives incoming control message, selects appropriate E2 interface and assigns an appropriate RIC Request ID. |
| Step 7 (M) | E2 Termination sends **RIC Control Request** (RIC Request ID, contents) to E2 Node. |
|  | [ALT] Steps 8-10 are executed if E2 Node accepts RIC Control Request: |
| Step 8 (M) | E2 Node responds with **RIC Control Response** |
|  | [ALT] Step 9a is executed if **E2 Related API: E2 Control** response is configured routed to xApp directly: |
| Step 9a (M) | E2 Termination sends **E2 Related API: E2 Control** response (Success with Outcome) to xApp directly. |
|  | [ELSE] Step 9b and 10 are executed if **E2 Related API: E2 Control** response is configured routed to xApp via Conflict Mitigation: |
| Step 9b [Informative] | E2 Termination sends RIC control acknowledge(Success with Outcome) to Conflict Mitigation. |
| Step 10 (M) | Conflict Mitigation sends **E2 Related API: E2 Control** response (Success with Outcome) to xApp. |
|  | [ELSE] Steps 11-13 are executed if E2 Node rejects RIC Control Request: |
| Step 11 (M) | E2 Node responds with **RIC Control Failure** (with Cause). |
|  | [ALT] Step 12a is executed if **E2 Related API: E2 Control** response is configured routed to xApp directly: |
| Step 12a (M) | E2 Termination sends **E2 Related API: E2 Control** response (Failure with Cause) to xApp directly. |
|  | [ELSE] Steps 12b and 13 are executed if **E2 Related API: E2 Control** response is configured routed to xApp via Conflict Mitigation: |
| Step 12b [Informative] | E2 Termination sends RIC control failure (Failure with Cause) to xApp via Conflict Mitigation. |
| Step 13 (M) | Conflict Mitigation sends **E2 Related API: E2 Control** response (Failure with Cause) to xApp. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “Conflict Mitigation” as ConMit

Collections “E2 Termination” as E2T

endbox

Collections “E2 Node” as ran

xApp <--> ConMit: 1: (E2 related API) Guidance (Request/Response)

alt E2 Related API configured to pass request directly to E2 termination

xApp -> E2T: 2a: (E2 related API) E2 Control (Request)

else E2 Related API configured to pass request via Conflict Mitigation

xApp -> ConMit: 2b: (E2 related API) E2 Control (Request)

ConMit -> ConMit: 3: Conflict mitigation processing

alt Conflict Mitigation accepts proposed message

ConMit -> E2T: 4: RIC control request

Else Conflict mitigation rejects proposed message

ConMit -> xApp: 5: (E2 related API) E2 Control (Reject)

end

end

E2T -> E2T: 6: Select appropriate E2 interface and assigns RIC Request ID

E2T -> ran: 7: <<E2>> RIC CONTROL REQUEST

alt E2 node accepts request

ran -> E2T: 8: <<E2>> RIC CONTROL ACKNOWLEDGE

alt E2 Related API configured to pass response directly to requesting xApp

E2T -> xApp: 9a: (E2 related API) E2 Control (Success)

else E2 Related API configured to pass response via Conflict Mitigation

E2T -> ConMit: 9b: RIC control acknowledge

ConMit -> xApp: 10: (E2 related API) E2 Control (Success)

end

else E2 node rejects request

ran -> E2T: 11: <<E2>> RIC CONTROL FAILURE

alt E2 Related API configured to pass response directly to requesting xApp

E2T -> xApp: 12a: (E2 related API) E2 Control (Failure)

else E2 Related API configured to pass response via Conflict Mitigation

E2T -> ConMit: 12b: RIC control failure

ConMit -> xApp: 13: (E2 related API) E2 Control (Failure)

end

end

@enduml

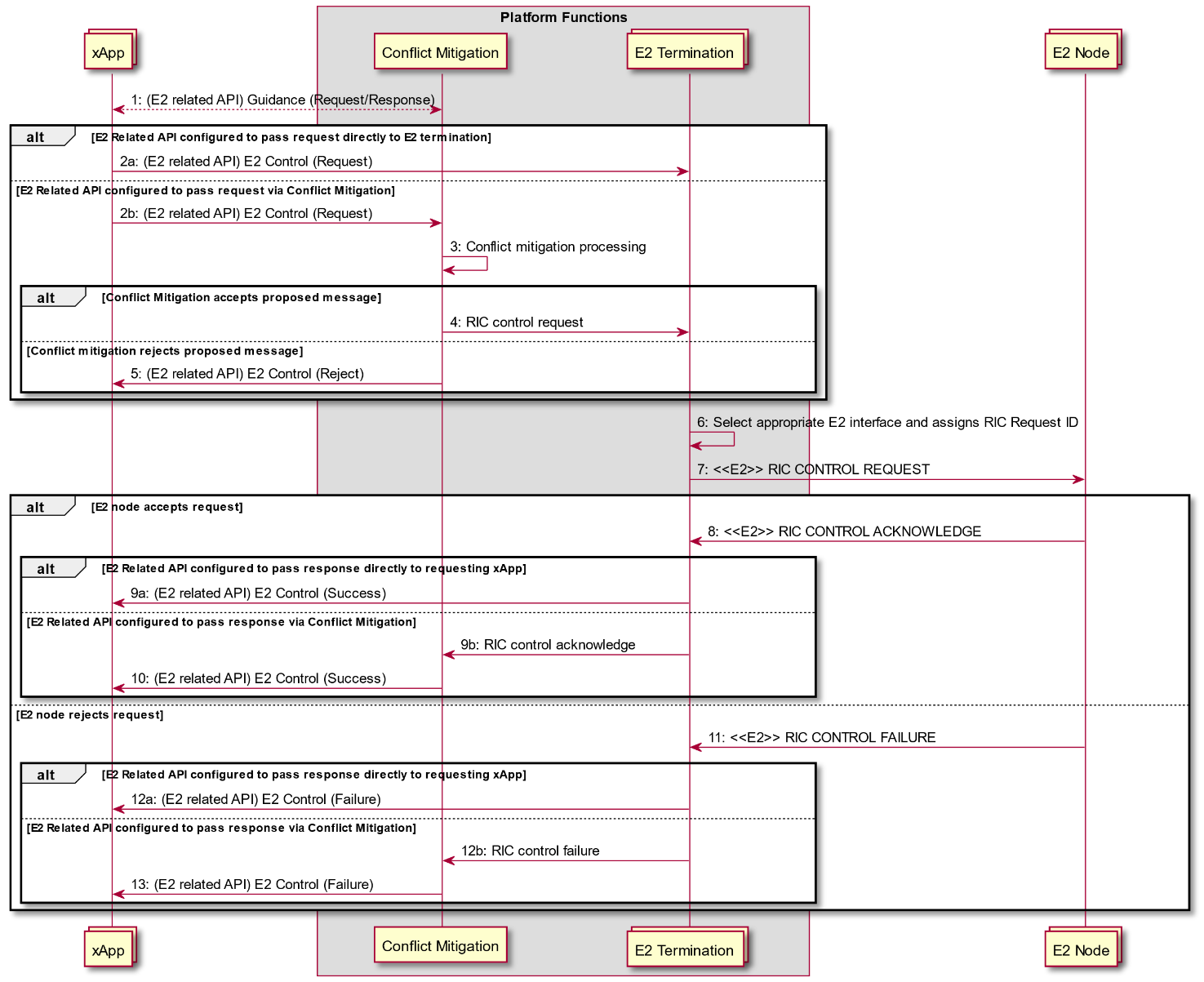


Figure 9.3.4-1 E2 Control Procedure

### 9.3.3 E2 Guidance related procedures

#### 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 Conflict Mitigation platform function prior to initiating an action.

Guidance from Conflict Mitigation 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 other platform functions.

This procedure is based on the following assumptions:

* xApp may use E2 Related API to obtain guidance from Conflict Mitigation to resolve potential conflicts prior to initiating a RIC function procedure;
* xApp may use Conflict Mitigation 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 other platform function using E2 Related API: E2 Guidance response.

|  |  |
| --- | --- |
| 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:   + Database;   + Conflict Mitigation. |
| Assumptions | * Conflict Mitigation has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution; * Conflict Mitigation may initiate guidance towards other Platform Functions and/or xApp as an optional addition response to Guidance Request. |
| Pre-conditions | * xApp has been authorized to request guidance from Conflict Mitigation; * xApp has been assigned xApp ID. |
| Begins when | xApp determines need to request guidance from Conflict Mitigation. |
| Step 1 (M) | xApp sends **E2 Related API E2 Guidance** Request to Conflict Mitigation. |
| Step 2 [Informative] | Conflict Mitigation may recover related information from platform database. |
| Step 3 [Informative] | Conflict Mitigation processes request. |
| Step 4 [Informative] | Conflict Mitigation may signal conflict and/or provide guidance to another platform function. |
| Step 5 (O) | Conflict Mitigation may signal conflict and/or provide guidance to another xApp using **E2 Related API E2 Guidance** (modification). |
| Step 6 (M) | Conflict Mitigation sends **E2 Related API E2 Guidance** response to xApp. |
| Ends with | xApp continues processing using Conflict Mitigation guidance response. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “Other xApp” as xApp

Collections “xApp” as xApp2

Box “Platform Functions”

Database "Database" as DB

Participant “Conflict Mitigation” as ConMit

endbox

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

ConMit <--> DB: 2: Fetch related information

ConMit -> ConMit: 3: Conflict mitigation processing

ConMit -->: 4: Guidance (Modification)

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

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

@enduml

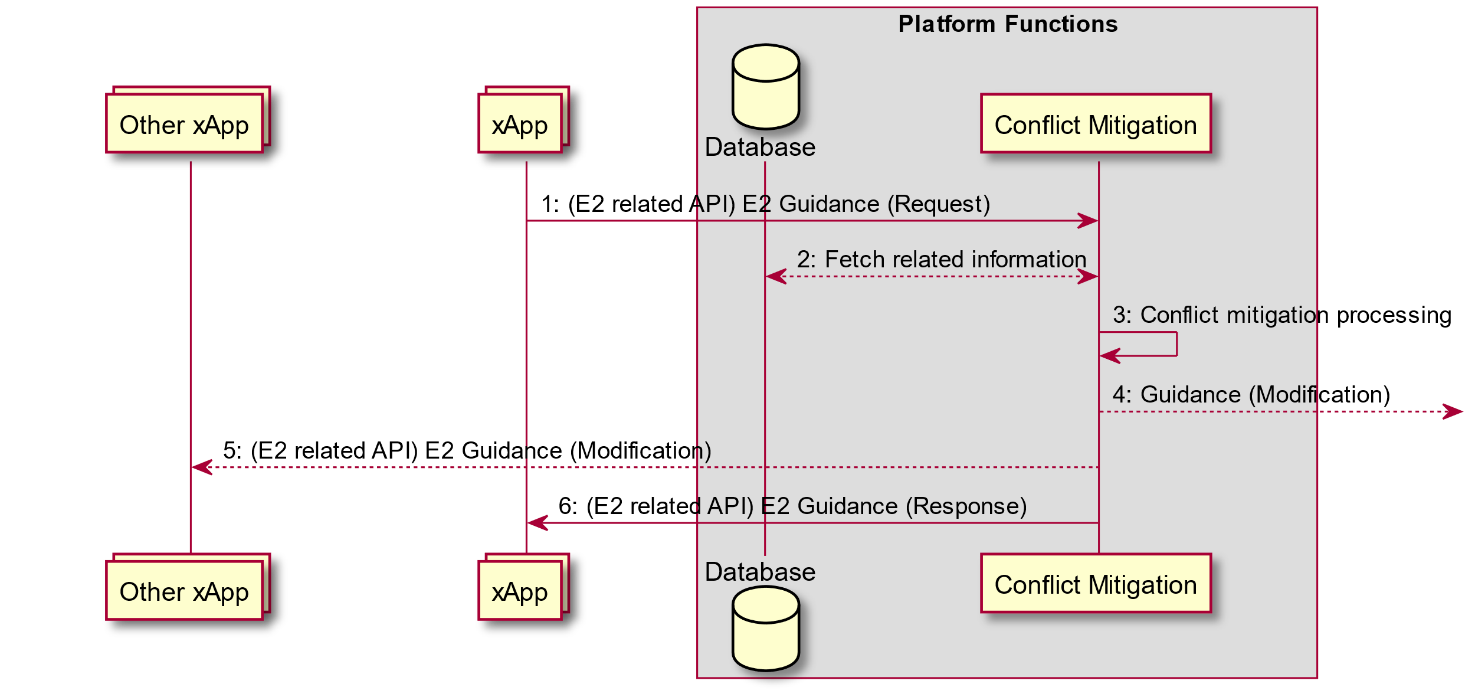


Figure 9.3.3.1-1: xApp initiated E2 guidance request/response procedure

#### 9.3.3.2 E2 Guidance modification procedure: xApp Subscription Management initiated

The purpose of the Near-RT RIC Platform internal xApp Subscription Management initiated E2 Guidance modification procedure is to allow Conflict Mitigation to signal a potential modification of E2 Guidance as the result of an xApp Subscription Management request to obtain guidance from the Conflict Mitigation platform function prior to handling an action.

This procedure is based on the following assumptions:

* xApp Subscription Management may request guidance from Conflict Mitigation to resolve potential conflicts prior to initiating a RIC function procedure;
* Conflict Mitigation may need to subsequentially modify previous provided Guidance to an xApp.

The following outcomes are considered:

* Conflict Mitigation provides guidance to xApp Subscription Management;
* Conflict Mitigation provides modified guidance to another Near-RT RIC platform function;
* Conflict Mitigation provides modified guidance to an xApp using **E2 Related API: E2 Guidance** modification.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | xApp Subscription Management initiation of Conflict Mitigation guidance resulting in potential E2 Related API: E2 Guidance modification sent to an xApp. |
| Actors and Roles | * Near-RT RIC platform:   + Database;   + Conflict Mitigation;   + xApp Subscription Management. |
| Assumptions | * Conflict Mitigation has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution; * Conflict Mitigation may initiate guidance towards other Platform Functions and/or xApp as an optional addition response to Guidance Request. |
| Pre-conditions | * xApp Subscription Management has been configured to request guidance from Conflict Mitigation. |
| Begins when | xApp Subscription Management receives incoming request. |
| Step 1 [Informative] | xApp Subscription Management requests guidance from Conflict Mitigation. |
| Step 2 [Informative] | Conflict Mitigation may recover related information from platform database. |
| Step 3 [Informative] | Conflict Mitigation processes request. |
| Step 4 [Informative] | Conflict Mitigation may signal conflict and/or provide guidance to another platform function. |
| Step 5 (O) | Conflict Mitigation may signal conflict and/or provide guidance to another xApp using **E2 Related API: E2 Guidance** (modification). |
| Step 6 [Informative] | Conflict Mitigation responds to xApp Subscription Management. |
| Step 7 [Informative] | xApp Subscription Management may consider Conflict Mitigation response to determine response to incoming request. |
| End with | xApp Subscription Management sends response to incoming request. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “Other xApp” as xApp

Collections “xApp” as xApp2

Box “Platform Functions”

Database "Database" as DB

Collections "xApp Subscription Management" as SubM

Participant “Conflict Mitigation” as ConMit

endbox

xApp2-> SubM: Message (Request)

SubM -> ConMit: 1: Guidance (Request)

ConMit <--> DB: 2: Fetch related information

ConMit -> ConMit: 3: Conflict mitigation processing

ConMit -->: 4: Guidance (Modification)

xApp <-- ConMit: 5: (E2 Related API) E2 Guidance (Modification)

ConMit -> SubM: 6: Guidance (Response)

SubM -> SubM: 7: Use guidance response as part of decision process

Alt Message accepted

SubM -> : Message (Request)

Else Message rejected

xApp2 <- SubM : Message (Reject)

end

@enduml

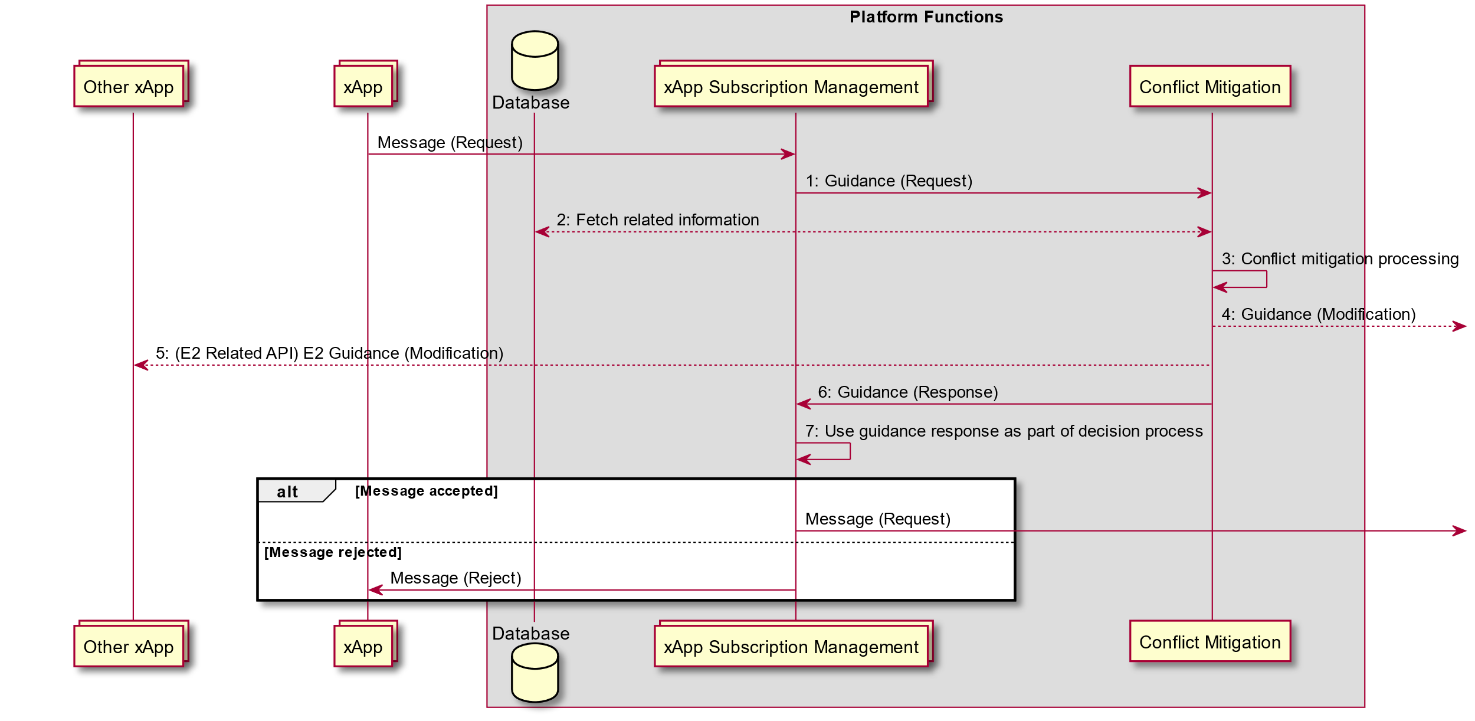


Figure 9.3.3.2-1: E2 Guidance modification API procedure: xApp Subscription Management initiated

#### 9.3.3.3 E2 Guidance modification procedure: message monitoring initiated

The purpose of the Conflict mitigation related message monitoring procedure in the Near-RT RIC is to allow the Conflict Mitigation platform function to monitor ongoing transactions involving other Platform Functions.

This procedure is based on the following assumptions:

* Other Platform functions (i.e. E2 termination, xApp Subscription Management) may be configured to forward messages related to xApp transactions to Conflict Mitigation;
* Conflict Mitigation may use information obtained when formulating responses in a subsequent procedure and/or modify previous Guidance to another Near-RT RIC platform function;
* Conflict Mitigation may use information to provide modified guidance to an xApp using **E2 Related API: E2 Guidance** modification.

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | Platform function forward xApp related messages to Conflict Mitigation resulting in potential E2 Related API: E2 Guidance modification sent to an xApp. |
| Actors and Roles | * Near-RT RIC platform:   + Conflict Mitigation;   + Other Platform Function:     - xApp Subscription Management;     - E2 Termination;     - Etc. |
| Assumptions | * Conflict Mitigation is capable of exploiting collected xApp related messages. |
| Pre-conditions | * Other Platform Functions have been configured to copy and forward selected xApp related message to Conflict Mitigation. |
| Step 1 [Informative] | Message received by other Platform Function. |
| Step 2 [Informative] | Other Platform Function handles incoming message and sends outgoing message. |
| Step 3 [Informative] | Other Platform Function applies message filter and determines that message is to be copied and sent to Conflict Mitigation. If filter conditions match, then forwards copy of incoming message to Conflict mitigation. |
| Step 4 [Informative] | Conflict Mitigation processes forwarded message. |
| Step 5 [Informative] | Conflict Mitigation may signal conflict and/or provide updated guidance to another platform function. |
| Step 6 (O) | Conflict Mitigation may signal conflict and/or provide updated guidance to another xApp using **E2 Related API: E2 Guidance** (modification). |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “Other xApp” as xApp

Box “Platform Functions”

Participant “Conflict Mitigation” as ConMit

Participant "Other Platform Function (i.e. E2 termination)" as Other

endbox

-> Other: 1: Message

Other -> Other : 2.1: Process incoming message

Other -> : 2.2: Message

Other -> Other : 3.1: Apply filter

Other -> ConMit: 3.2: Forward message

ConMit -> ConMit: 4: Process forwarded message

ConMit -->: 5: Guidance (Modification)

xApp <-- ConMit: 6: (E2 Related API) E2 Guidance (Modification)

@enduml

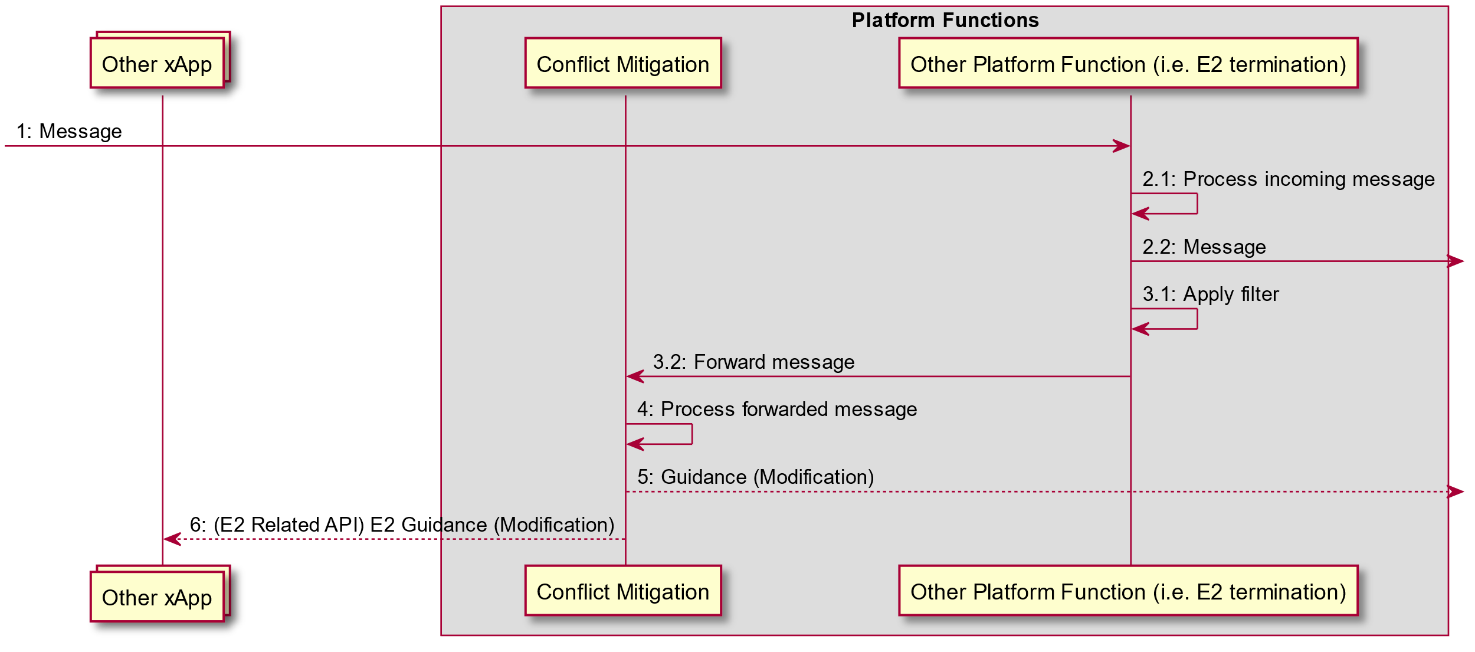


Figure 9.3.3.3-1: E2 Guidance modification API procedure: message monitoring initiated

#### 9.3.3.4 E2 Guidance modification procedure: Conflict Mitigation initiated

The purpose of the Conflict Mitigation initiated conflict mitigation procedure in the Near-RT RIC is to allow Conflict Mitigation to directly intervene during the handling an action.

This procedure is based on the following assumptions:

* Messages from xApp or other platform functions may be redirected towards Conflict Mitigation to resolve potential conflicts prior to initiating a RIC function procedure;
* xApp would not be aware of the redirection if conflict mitigation accepts the proposed message.

The following outcomes are considered:

* Conflict Mitigation accepts proposed message and forwards to other platform function;
* Conflict Mitigation rejects proposed message and sends a rejection message to the initiating xApp or other platform function providing both cause for rejection and optionally guidance.

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | Conflict Mitigation initiation of Conflict Mitigation guidance resulting in potential E2 Related API: E2 Guidance modification sent to an xApp. |  |
| Actors and Roles | * xApp or other platform function; * Near-RT RIC platform:   + Database;   + Conflict Mitigation. |  |
| Assumptions | * Messages from xApp or other platform function may be redirected to Conflict Mitigation; * Conflict Mitigation has access to sufficient information to both detect a potential conflict and take a decision on an optimal mitigation solution; * Conflict Mitigation may initiate guidance towards other Platform Functions and/or xApp as an optional addition response to Guidance Request. |  |
| Pre-conditions | * Message infrastructure has been configured to redirect specified messages to Conflict Mitigation. |  |
| Begins when | xApp or other platform function sends message to platform function. | RIC functional procedures |
| Step 1 [Informative] | Message infrastructure redirects message from xApp to Conflict Mitigation. |  |
| Step 2 [Informative] | Conflict Mitigation may recover related information from SDL. |  |
| Step 3 [Informative] | Conflict Mitigation processes request. |  |
|  | [ALT] Step 4 is executed If Conflict Mitigation accepts the request: |  |
| Step 4 [Informative] | Conflict Mitigation may forward to E2 termination for delivery. |  |
|  | [ELSE] Step 5 is executed If Conflict Mitigation rejects the request: |  |
| Step 5 [Informative] | Conflict Mitigation may send a rejection message to xApp including indication of cause and optionally guidance information. |  |
| Step 6 (O) | Conflict Mitigation may signal conflict and/or provide guidance to another platform function or xApp. |  |
| Step 7 (O) | Conflict Mitigation may signal conflict and/or provide updated guidance to another xApp using **E2 Related API: E2 Guidance** (modification). |  |
| Ends with | xApp receives eventual response from E2 Node. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections “Other xApp” as xApp

Box “Platform Functions”

Database "Database" as DB

Participant “Conflict Mitigation” as ConMit

Endbox

-> ConMit: 1: Message (Request)

ConMit <--> DB: 2: Fetch related information

ConMit -> ConMit: 3: Conflict mitigation processing

Alt Conflict mitigation accepts message

ConMit -> : 4: Message (Request)

Else Conflict Mitigation rejects message

<- ConMit: 5: Message (Reject, Cause)

End

ConMit -->: 6: Guidance

xApp <-- ConMit: 7: (E2 related) E2 Guidance (Modification)

@enduml

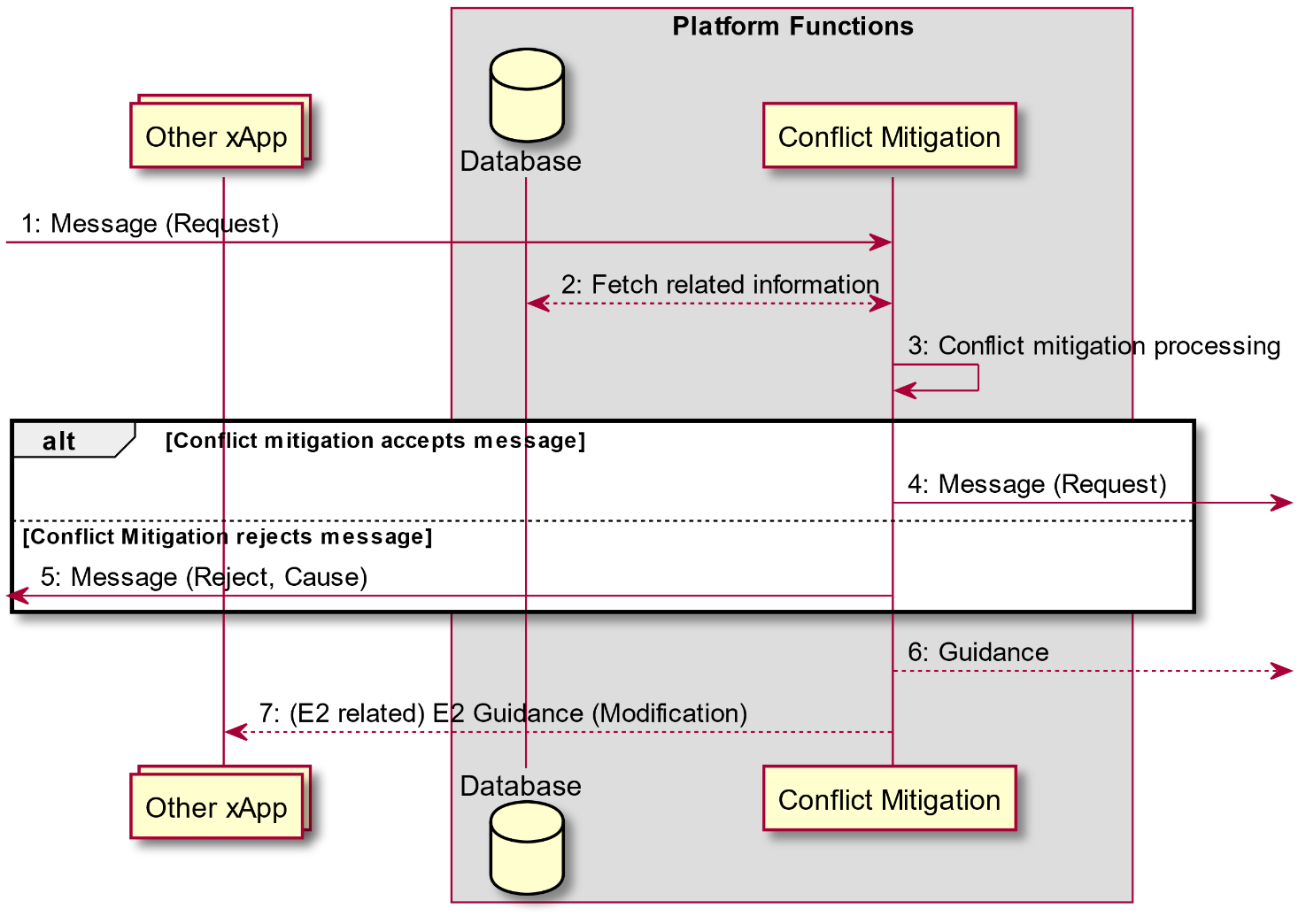


Figure 9.3.3.4-1: E2 Guidance modification API procedure: Conflict Mitigation initiated

## 9.4 Management API Procedures

### 9.4.1 xApp Registration procedure

This procedure registers the xApp to the Near-RT RIC platform so that it can be managed by the SMO via the Near-RT RIC platform. When the xApp is deployed, it will register itself as a managed application to the Near-RT RIC platform by passing the relevant info through the API. Relevant info may include the xApp name, vendor, software version, YANG schemas for configuration management, faults raised, metrics generated, ports for messaging, supported commands (e.g. health check, aliveness probe) and other information needed by the Near-RT RIC platform and/or the SMO for managing the xApp.

In response to an xApp Registration, Near-RT RIC platform performs the following:

* Authenticates the xApp;
* Validates that this xApp can run on this Near-RT RIC platform; e.g. RIC has the capacity;
* Assigns an ID to the xApp;
* Creates an xApp Managed Object Instance (MOI) in the Near-RT RIC Config DB and populates it with the relevant information passed from the xApp;
* Sends a Notify MOI Creation to SMO if subscribed.

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | To register the newly deployed xApp to the Near-RT RIC platform. |  |
| Actors and Roles | * xApp: originator of the xApp registration procedure; * Management Function in Near-RT RIC platform: handle registration related message from/to xApp and O1 termination and may do initial validation of xApp. |  |
| Assumptions | The API channel is established successfully between xApp and Management Function in the Near-RT RIC platform. |  |
| Pre conditions | The xApp is deployed onto the Near-RT RIC platform. |  |
| Begins when | An xApp is deployed and ready for registration. |  |
| Step 1 (M) | xApp send xApp registration request to the Management Function Component in the Near-RT RIC platform. passing relevant information needed to manage the xApp. |  |
| Step 2 (M) | Near-RT RIC platform processes the application registration, which includes (not an exhaustive list):   * Performing authentication and validity checks; * If the checks pass, assigning an xApp ID for the xApp; * Creating an xApp Managed Object Instance in the Near-RT RIC Config DB in compliance with the YANG schemas; * Populating xApp MOI with the xApp ID and the initial xApp configuration. |  |
| Step 3 (M) | The Management Function send the registration response to the xApp. If the response indicates the registration fails, it will send the registration response with failure results and failure cause. |  |
| Step 4 (O) | If SMO has subscribed to CM Notifications from the Near-RT RIC, O1 Terminations formats and sends a Notify MOI Creation to SMO to notify SMO that a MOI has been created for a new xApp. |  |

@startuml

skin rose

skinparam ParticipantPadding 5

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

Box “O-Cloud”

participant dms as “DMS”

endbox

participant xApp as "xApp"

Box “Near-RT RIC Platform”

participant o1t as “O1 Termination”

participant ms as “Management Function”

endbox

xApp -> ms : 1. <<MS>> registration request

rnote over o1t, ms

2. registration processing

validate, assign xApp ID, create xApp MOI

endnote

ms -> xApp : 3. <<MS>> registration response

Opt If SMO has subscribed

o1t -> smo : 4. <<O1>> Notify MOI Creation

End

@enduml

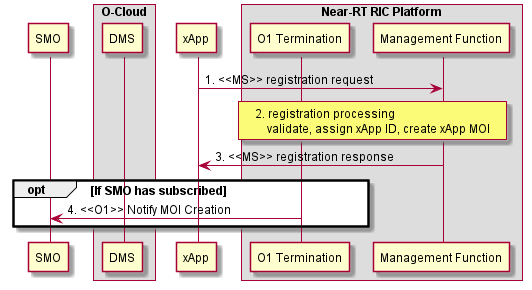


Figure 9.4.1-1: xApp Registration procedure

### 9.4.2 xApp Deregistration procedure

This procedure deregisters the xApp from the Near-RT RIC platform.

The message flow is FFS.

### 9.4.3 Void

### 9.4.4 Create MOI

An xApp must be instantiated before its associated MOI can be created. The SMO interacts with the MOI to manage the xApp. The Management Function is responsible for updating the xApp over the Management API to ensure that its attribute values are aligned with those of the MOI.

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | To Create a Managed Object Instance for an xApp. |  |
| Actors and Roles | * Near RT-RIC platform * xApp |  |
| Assumptions | The API channel is established successfully between xApp and Management Function in the Near-RT RIC platform. |  |
| Preconditions | * xApp is instantiated; * MOI is not yet created. |  |
| Begins when | Create MOI operation received over O1 |  |
| Step 1 (M) | Management Function sends an APIConfirgurationWrite updating the xApp with the attribute values from the newly created MOI |  |
| Step 2 (M) | After applying the configuration data to the xApp, a configuration response is sent over the management API to the Management Function. |  |
| Ends when | Create MOI Response is sent over O1. |  |

@startuml

skin rose

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

participant xApp as "xApp"

smo-> ric : <<O1>> createMOI(…)

ric -> xApp : 1. <<API>> xApp Configuration Request

ric <-- xApp : 2. <<API>> Result

smo<-- ric : <<O1>> Result

@enduml

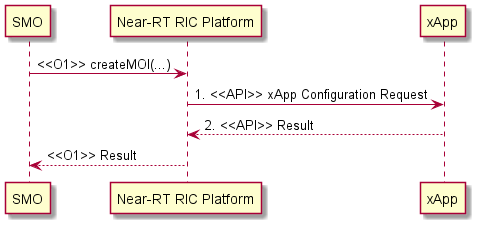


Figure 9.4.4-1: xApp Create MOI

### 9.4.5 Modify MOI attributes

The Management Function entity is responsible for updating the xApp over the Management API to ensure its attribute values are aligned with those of the MOI.

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | To change an existing xApp MOI configuration. |  |
| Actors and Roles | * Near RT-RIC platform * xApp |  |
| Assumptions | * The API channel is established successfully between xApp and the Near-RT RIC platform. |  |
| Pre conditions | * The xApp is installed successfully on the Near-RT RIC platform and registered to the SMO; * The MOI for the target xAPP already exists. |  |
| Begins when | O1 Termination / Management Function receives the configuration message from SMO to indicate that there’s a configuration update for the xApp. |  |
| Step 1 (M) | Management Function sends an APIConfirgurationWrite updating the xApp with the attribute values contained in the O1 Modify MOI message. |  |
| Step 2 (M) | The xApp sends a Configuration Response. |  |
| Ends when | O1 Termination notifies SMO of success or error. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

participant xApp as "xApp"

smo-> ric : <<O1>> ModifyMOIAttributes(…)

ric -> xApp : 1. (API) xApp Configuration Request

ric <-- xApp : 2. (API) Result

smo<-- ric : <<O1>> Result

@enduml

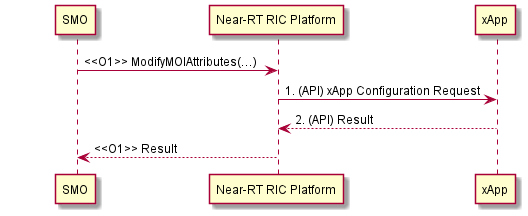


Figure 9.4.5-1: xApp Modify MOI

### 9.4.6 Delete MOI

This procedure removes the MOI for an an xApp when an edit-config delete request is received from the SMO over the O1 interface. See [5] clause 2.1.3. The SMO sends the deleteMOI request via a NETCONF command over the O1 to the O1 Termination in the Near-RT RIC platform. The Management Function may optionally take the xApp out of service sending a confirmation of the state change before the MOI gets deleted. Once the MOI is deleted, the Management Function sends a NETCONF response to the O1 Termination confirming the success or failure of the O1 request. The O1 Termination forwards the response to the MnS Consumer (SMO).

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | To remove an existing xApp MOI. |  |
| Actors and Roles | * O1 Termination * Management Function * xApps |  |
| Assumptions | The API channel is established successfully between xApp and Management Function in the Near-RT RIC platform. |  |
| Pre conditions | The MOI exists on the target Near-RT RIC. |  |
| Begins when | O1 Termination / Management Function receives the edit-config Delete message from SMO to indicate that the SMO wants to delete the MOI. |  |
| Step 1 (O) | Management Function may optionally lock the administrative state of the xApp. |  |
| Step 2 [Informative] | Management Function deletes the MOI. |  |
| Step 3 [Informative] | Management Function notifies O1 Termination of the result. |  |
| Ends when | The MnS Consumer (SMO) receives the O1 response. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

participant xApp as "xApp"

smo -> ric : <<O1>> DeleteMOIRequest

Opt Lock xApp state

ric->xApp: 1. API Configuration Write \n(Change Adminstrative State to “Locked”)

end

note over ric : MOI Deleted

smo <-- ric : <<O1>> Result

@enduml

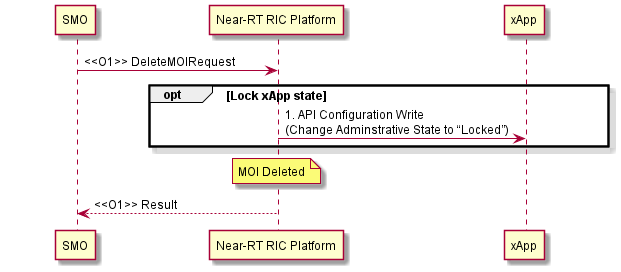


Figure 9.4.6-1: xApp Delete MOI

### 9.4.7 Read MOI attributes

The O1 Read MOI Attributes service is supported by the xApp API Configuration Read. On receipt of a Read MOI Attributes request for an xApp, the Management Function uses the API Configuration Read to retrieve the current configuration values.

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | To Read the current attribute values for an existing xApp MOI. |  |
| Actors and Roles | * O1 Termination * Management Function * xApps |  |
| Assumptions | * The API channel is established successfully between xApp and Management Function in the Near-RT RIC platform. |  |
| Pre conditions | * The MOI exists on the target Near-RT RIC. |  |
| Begins when | O1 Termination / Management Function receives the Read MOI Attributes request from SMO to indicate that the SMO wants the current xApp attributes. |  |
| Step 1 (O) | Management Function sends xApp Configuration Read  (Optional because the information is already available in the MOI in Management Function). |  |
| Step 2 (O) | xApp makes the operational configuration available to Management Function. |  |
| Ends when | The MnS Consumer (SMO) receives the O1 response. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

participant xApp as "xApp"

smo -> ric : <<O1>> Read MOI Attributes(…)

Opt Sync MOI with xApp

ric -> xApp : 1. (API) xApp Configuration Request

ric <--xApp : 2. (API) Result

end

smo <-- ric : <<O1>> Result

@enduml

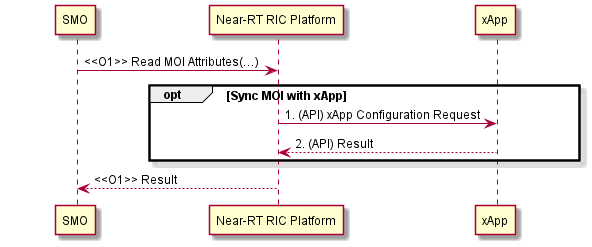


Figure 9.4.7-1: Read MOI Attributes

### 9.4.8 Notify MOI changes

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution / Specification | <<Uses>>  Related use |
| Goal | To notify Management Function that some change has occurred in the xApp configuration |  |
| Actors and Roles | * Near-RT RIC platform * xApps |  |
| Assumptions | * The API connection is established successfully between xApp and Management Function in the Near-RT RIC platform. |  |
| Pre conditions | * The MOI exists on the target Near-RT RIC. |  |
|  | [ALT 1] Attribute Value Change Notification (AVCN) |  |
| Begins when | The Near-RT RIC platform receives an xApp Notify change message from the xApp. |  |
| Step 1 [Informative] | The Near-RT RIC platform sends O1 Attribute Value Change Notification to the O1 Consumer (SMO). |  |
|  | [ALT 2] MOI Creation Notification |  |
| Step 1 [Informative] | The Near-RT RIC platform sends O1 MOI Creation Notification to the O1 Consumer (SMO). |  |
| Begins when | A Managed Object Instance representing xApp has been created. |  |
|  | [ALT 3] MOI Deletion Notification |  |
| Step 1 [Informative] | The Near-RT RIC platform sends O1 MOI Deletion Notification to the O1 Consumer (SMO). |  |
| Begins when | A Managed Object Instance representing xApp has been Deleted. |  |
|  | [ALL CASES] |  |
| Ends when | The MnS Consumer (SMO) receives the O1 response. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

participant xApp as "xApp"

alt Attribute Value Change Notification

ric <-xApp : <<API>> xApp Notify Change

loop for each subscription

smo<- ric : <<O1>> Notify MOI Change(AVCN)

end

else MOI Creation Notification

ref over ric: MOI Creation

loop for each subscription

smo<- ric : <<O1>> Notify MOI Change(MOI Created)

end

else MOI Deletion Notification

ref over ric: MOI Deletion

loop for each subscription

smo<- ric : <<O1>> Notify MOI Change(MOI Deleted)

end

end

@enduml

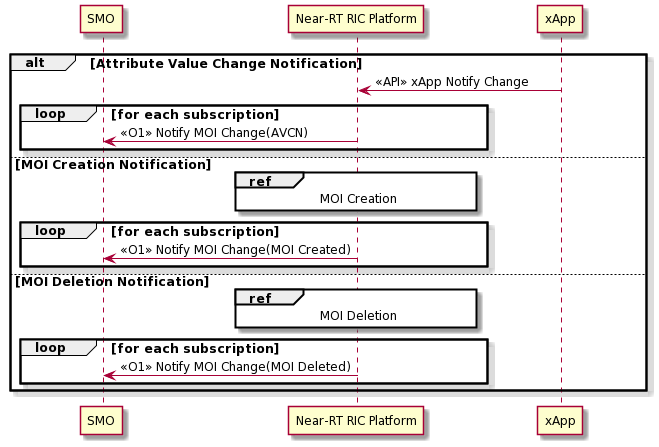


Figure 9.4.8-1: Notify MOI Changes

### 9.4.9 Subscription Control

|  |  |
| --- | --- |
| Use Case Stage | Evolution / Specification |
| Goal | Allows a MnS Consumer to subscribe to notifications emitted by a MnS Provider. |
| Actors and Roles | * O1 Termination * Management Function * xApps |
| Assumptions | * The API channel is established successfully between xApp and Management Function in the Near-RT RIC platform. |
| Pre conditions | * An MOI exists on the target Near-RT RIC representing the xApp functionality. |
| Begins when | O1 Termination / Management Function receives a Create MOI message with arguments defining the MOI as a Subscription and associating it with one or more xApps and a consumer (SMO). |
| Step 1 | A Subscription Control MOI reflecting the scope and filtering requirements for the subscription is created and associated with the instance of Managed Application representing the xApp. |
| Ends when | The MnS Consumer (SMO) receives the O1 response. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

participant xApp as "xApp"

smo-> ric : <<O1>> createMOI(xApp subscription)

Opt message received from xApp

ric <-xApp : 1. (API) xApp Notify Change

end

smo<-- ric : <<O1>> Notification

@enduml

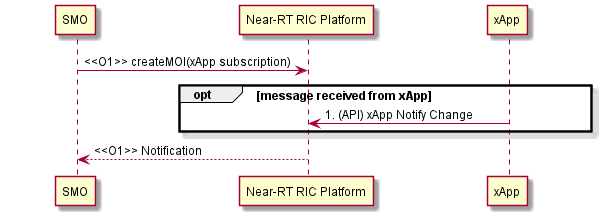


Figure 9.4.9-1: xApp Subscription Control

### 9.4.10 Fault Notification

This procedure enables the xApps to notify the Near-RT RIC of a Fault Condition. Management Function on the Near-RT RIC may create an Alarm notification based on the fault reported over the API, and supplement it with additional information from the Alarm Dictionary. The notification is also used to notify Alarms changes.

| **Use case stage** | **Evolution/Specification** | **<<Uses>> Related use** |
| --- | --- | --- |
| Goal | xApp informs the Near-Rt RIC of the existence of a Fault condition |  |
| Actors and Roles | * xApp * Near-RT RIC platform |  |
| Assumptions |  |  |
| Pre-conditions |  |  |
| Begins when | A fault condition is detected at the xApp. |  |
| Step 1 (M) | xApp Management API sends a fault notification. |  |
| Step 2 (M) | Management Function determines whether to create an Alarm notification for the fault. |  |
| Ends when | xApp sends Fault Notification to the Near-RT RIC. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

Collections xApp as "xApp"

xApp -> ric : 1. Fault condition detected

Opt Create Alarm

ric --> ric: 2. If the reported fault condition maps to a subscribed Alarm

end

smo<-- ric : <<01>> Fault Notification

@enduml

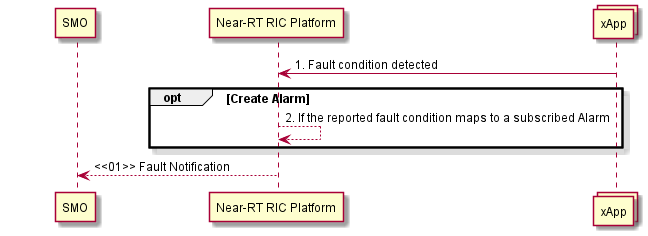


Figure 9.4.10-1: Fault Notification

### 9.4.11 Fault Supervision Control

The O1 Interface supports the 3GPP “getAlarmList”. The Alarm list for the xApps is maintained on the Near-RT RIC.   
The xApp can notify the Near-RT RIC Management Function of a detected fault. Each xApp shall maintain a list of any faults that are active. These faults are notified to the Near RT RIC where the necessary Alarms are created.

The SMO (consumer) may need to verify which alarms are currently active and so may request that the Near-RT RIC (producer) provides an updated list of Alarms. In order to verify that its Alarm list correctly reflects the state of faults at the xApp, the Near RT RIC can request an updated list of faults from the xApp.

| Use case stage | Evolution/Specification | <<Uses>> Related use |
| --- | --- | --- |
| Goal | To validate that the Near-RT RIC Alarm list for a given xApp correctly reflects the fault status of that xApp |  |
| Actors and Roles | * SMO * Near-RT RIC platform * xApp |  |
| Assumptions |  |  |
| Pre-conditions | Target xApp in service. |  |
| Begins when | O1 consumer (SMO) requests Near-RT RIC to fetch the current Alarm List. |  |
| Step 1 (M) | Near-RT RIC requests xApp to provide the list of current faults. |  |
| Step 2 (M) | xApp sends list of current faults to Near-Rt RIC. (Management Function). |  |
| Ends when | O1 consumer (SMO) receives current Alarm list. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

Collections xApp as "xApp"

smo-> ric : <<O1>> getAlarmList

ric-> xApp: 1. (API) Get Fault List

ric <-- xApp: 2. (API) Return Fault list

smo<-- ric : <<O1>> Alarm InformationList

@enduml

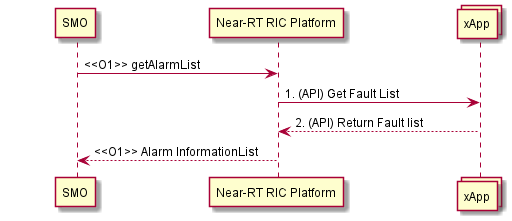


Figure 9.4.11-1: Get Alarm List

9.4.12 Performance Data File Reporting

| Use case stage | Evolution/Specification | <<Uses>> Related use |
| --- | --- | --- |
| Goal | xApp informs Management Function that a file is ready and provides the location for that file. |  |
| Actors and Roles | * SMO * Near-RT RIC platform * xApp |  |
| Assumptions |  |  |
| Pre-conditions | PerfMetricJob exists and includes a valid stream target value. |  |
| Begins when | PM Streaming configuration created. |  |
| Step 1 (M) | xApp sends API notification that a file is ready also providing the location of that file. |  |
| Step 2 [Informative] | <<O1>> File Ready notification sent. |  |
| Ends when | SMO receives notification. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

Collections xApp as "xApp"

ric <- xApp: (API) 1. xApp Notify File Ready

Note over ric : 1 New PM data file available

smo<- ric : 2. <<O1>> notifyFileReady Notification

@enduml

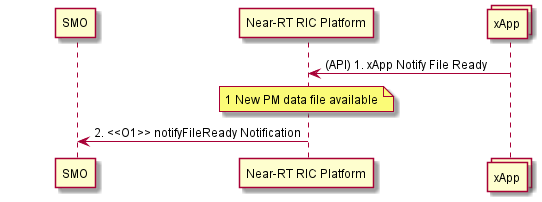


Figure 9.4.12-1: Performance Data File Reporting

9.4.13 Report Streamed Data

When the ReportingCtrl Attribute of the PerfMetricJob ( [21] Clause 4.3.33) is set to streamTarget, the Near-RT RIC shall stream the PM data to the target consumer destination based on the value of the “streamTarget” attribute.

| Use case stage | Evolution/Specification | <<Uses>> Related use |
| --- | --- | --- |
| Goal | Report Streamed Data |  |
| Actors and Roles | * SMO * Near-RT RIC platform * xApp |  |
| Assumptions |  |  |
| Pre-conditions | PerfMetricJob exists and includes a valid stream target value. A streaming connection has been established between the Near-RT RIC and the SMO. |  |
| Begins when | PM Streaming configuration created. |  |
| Step 1 (M) | (API) Management Function directs xApp to start streaming data. |  |
| Step 2 (M) | (API) xApp streams data to Management Function. |  |
| Step 3 [Informative] | O1 Termination streams PM data to subscriber. |  |
| Ends when | SMO receives streamed data. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

Collections xApp as "xApp"

Note over ric : Streaming target specified in PerfMetricJob

ric->xApp: 1. (API) Start Streaming

Loop While streaming is active

xApp ->ric: 2. (API) Sending streamed data

smo<-ric : 3. <<O1>> reportStreamData operation

end

@enduml

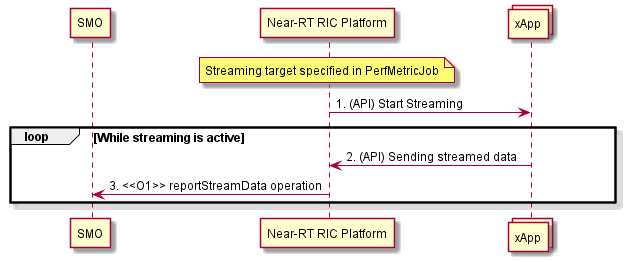


Figure 9.4.13-1 Report Streamed Data

9.4.14 Measurement Job Control

Both the selection of the data at the xApp and the formatting of the data at the Management Function are in scope for this procedure. The CreateMeasurementJobControl request received over O1 also determines whether the PM data would be file based or streamed.

| Use case stage | Evolution/Specification | <<Uses>> Related use |
| --- | --- | --- |
| Goal | To configure the collection of PM measurements |  |
| Actors and Roles | * SMO * Near-RT RIC platform * xApp |  |
| Assumptions |  |  |
| Pre-conditions | Target xApp in service. |  |
| Begins when | “Create PerfMetricJob” received on O1 interface. |  |
| Step 1 (M) | Configure xApp to collect and forward PM data. |  |
| Step 2 (M) | PM collection response. |  |
| Ends when | Successful creation of PM PerformanceMetric Job confirmed on O1. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

participant smo as “SMO”

participant ric as “Near-RT RIC Platform”

Collections xApp as "xApp"

smo-> ric : <<O1>> Create PerfMetricJobMOI

ric -> xApp : 1. (API) xApp Configuration Request\n (configure Measuremnt Job)

ric <-- xApp : 2. (API) Result

smo<-- ric : <<O1>> Result

@enduml

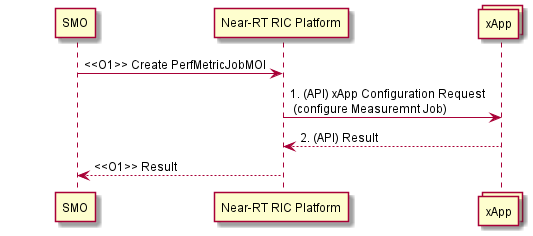


Figure 9.4.14-1 Measurement Job Control

## 9.5 SDL API Procedures

### 9.5.1 SDL Client Registration procedure

The SDL Client Registration procedure in the Near-RT RIC enables an xApp to register with the SDL for permission to access the database.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | xApp requests the SDL as a client for the permission to access database. |
| Actors and Roles | * xApp: Originator of SDL Client Registration request; * SDL: Response to the request of xApp. |
| Pre-conditions | * SDL API services have been registered and authorized by API Enablement. |
| Begins when | xApp determines to register with SDL to access the database in the Near-RT-RIC platform. |
| Step 1 (M) | xApp sends SDL Client Registration request to SDL to obtain permission to access the database. |
| Step 2  [Informative] | SDL arranges with the database the permission for the xApp to access the database (internal implementation). |
| Step 3 (M) | SDL sends the response of successful or failed registration to the xApp. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL” as SDL

Database "Database" as DB

endbox

xApp -> SDL: 1. SDL Client Registration request

SDL <--> DB: 2. Register to database

SDL -> xApp: 3. SDL Client Registration response

@enduml

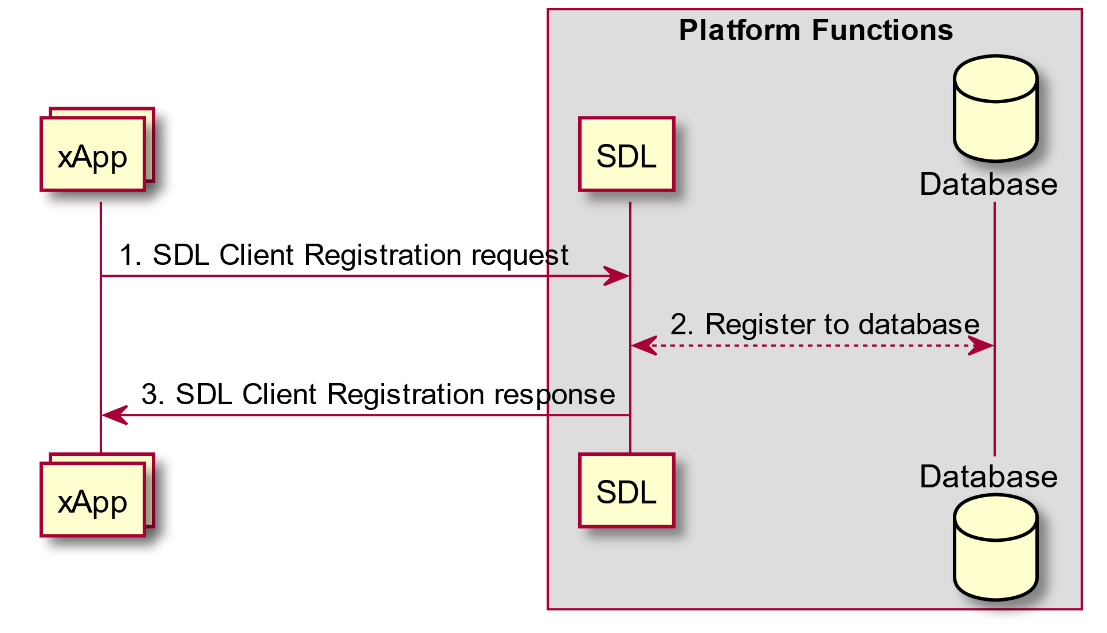


Figure 9.5.1-1: SDL Client Registration procedure

### 9.5.2 SDL Client Deregistration procedure

The SDL Client Deregistration procedure in the Near-RT RIC enables an xApp to request the SDL to release the registration.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | xApp requests the SDL to release the registration. |
| Actors and Roles | * xApp: Originator of SDL Client Deregistration request; * SDL: Response to the request of xApp. |
| Pre-conditions | * SDL API services have been registered with API Enablement; * xApp has successfully registered as a client to the SDL before. |
| Begins when | xApp determines to release the registration with SDL. |
| Step 1 (M) | xApp sends Client Deregistration request to SDL to release the connection with database. |
| Step 2  [Informative] | SDL deregisters from database (internal implementation). |
| Step 3 (M) | SDL sends the response of successful or failed deregistration to the xApp. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL” as SDL

Database "Database" as DB

endbox

xApp -> SDL: 1. SDL Client Deregistration request

SDL <--> DB: 2. Deregister from database

SDL -> xApp: 3. SDL Client Deregistration response

@enduml

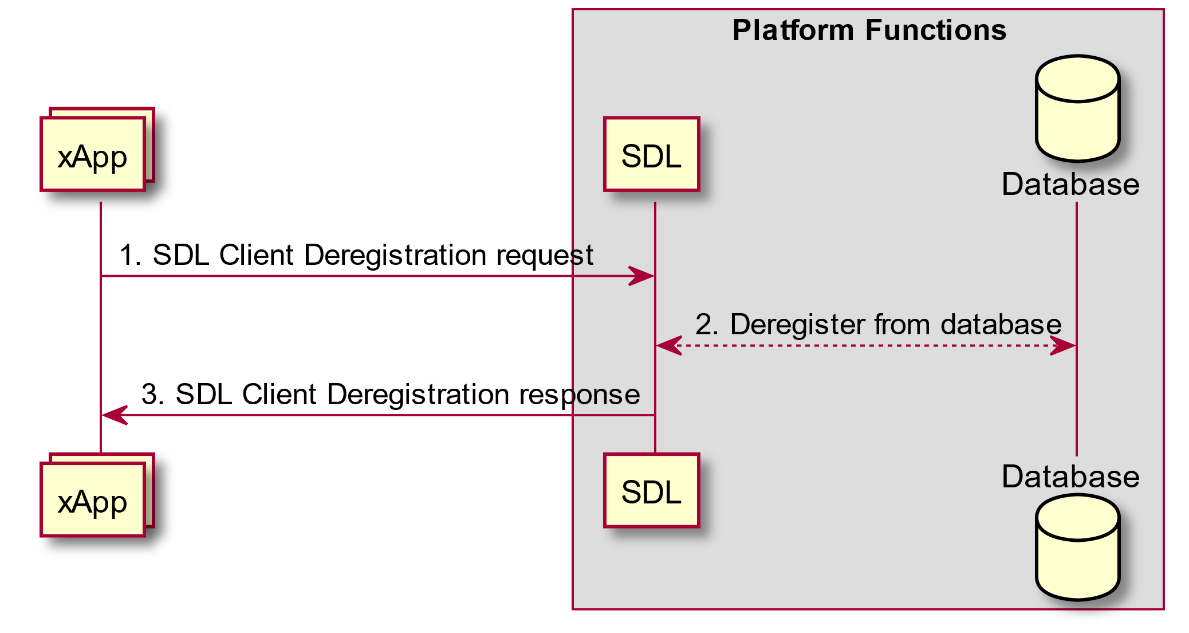


Figure 9.5.2-1: SDL Client Deregistration procedure

### 9.5.3 Fetch Data procedure

The Fetch Data procedure in the Near-RT RIC enables an xApp to request data for which it is authorized from the SDL for local processing.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | xApp fetches data from the SDL. |
| Actors and Roles | * xApp: Originator of Fetch Data request; * SDL: Response to the request of xApp. |
| Pre-conditions | * SDL API services have been registered with API Enablement; * xApp has successfully registered as a client to the SDL before; * xApp has been authorized to read the data in the database. |
| Begins when | xApp determines to fetch relevant data from the SDL in the Near-RT-RIC platform. |
| Step 1 (M) | xApp requests to SDL to fetch data from database. |
| Step 2  [Informative] | SDL fetches xApp required information from database (internal implementation). |
| Step 3 (M) | SDL sends the requested data or the failure response to specific xApp. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL” as SDL

Database "Database" as DB

endbox

xApp -> SDL: 1. Fetch data request

SDL <--> DB: 2. Fetch related information

SDL -> xApp: 3. Fetch data response

@enduml

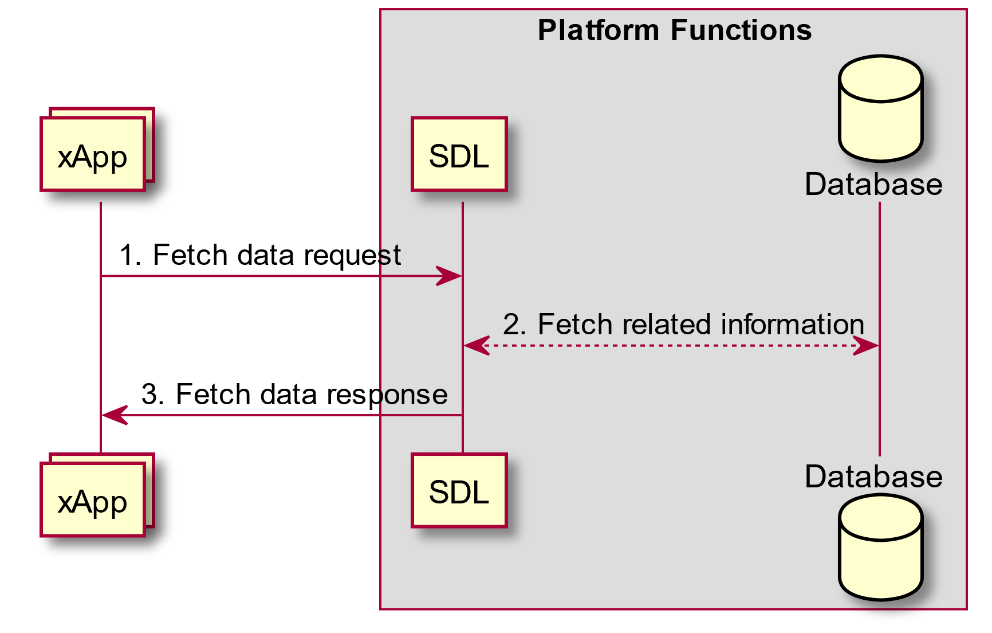


Figure 9.5.3-1: Fetch Data procedure

### 9.5.4 Subscribe/Notify procedure

The Subscribe/Notify procedure in the Near-RT RIC allows the xApp to subscribe to SDL for the authorized data changes in the database. The SDL will notify the information changes to the xApp, and the xApp may subsequently request updated data through the Fetch Data procedure.

The notifications are generated only for changes to the information after the subscription.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | XApp subscribes with the SDL for data changes and receives related notifications. |
| Actors and Roles | * xApp: Originator of Subscribe request; * SDL: Response to the request of xApp, and notify the subscribed information to xApp. |
| Pre-conditions | * SDL API services have been registered by API Enablement; * xApp has successfully registered as a client to the SDL before. |
| Begins when | xApp determines to subscribe for notifications related to information changes in the SDL in the Near-RT-RIC platform. |
| Step 1 (M) | xApp sends the request to SDL to subscribe for notifications related to information changes. |
| Step 2 (M) | SDL responds to the xApp subscription. |
| Step 3 (M) | SDL sends the notification for the subscribed information changes to the xApp. The xApp may further send a request to SDL to fetch the updated information. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL + Database” as SDL

endbox

xApp -> SDL: 1. Subscribe information request

SDL -> xApp: 2. Subscribe information response

SDL -> xApp: 3. Notify information changes

@enduml

Text, table

Description automatically generated with medium confidence

Figure 9.5.4-1: Subscribe/Notify procedure

### 9.5.5 Store Data procedure

The Store Data procedure in the Near-RT RIC ensures that the xApp may request SDL to insert data to the database.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | XApp requests to store data in the SDL. |
| Actors and Roles | * xApp: Originator of Store Data request; * SDL: Response to the request of xApp. |
| Pre-conditions | * SDL API services have been registered by API Enablement; * xApp has successfully registered as a client to the SDL before; * xApp has been authorized to write the data in the database. |
| Begins when | xApp determines to store relevant data to the database in the Near-RT-RIC platform. |
| Step 1 (M) | xApp sends a request to SDL to store related information and try to insert data to database. |
| Step 2  [Informative] | SDL stores xApp requested information in database (internal implementation). |
| Step 3 (M) | SDL responds to the xApp, including success and failure operation. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL” as SDL

Database "Database" as DB

endbox

xApp -> SDL: 1. Store data request

SDL <--> DB: 2. Store related data

SDL -> xApp: 3. Store data response

@enduml

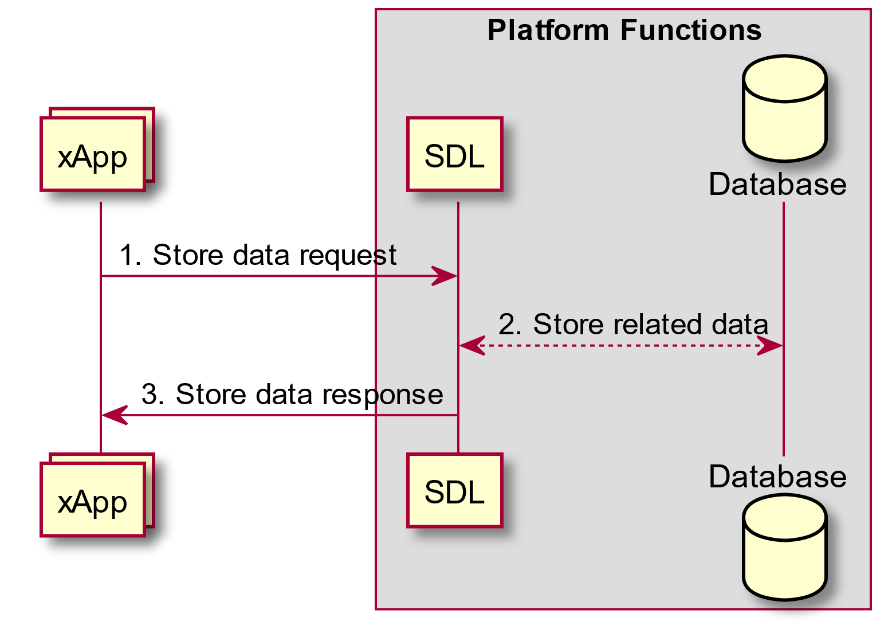


Figure 9.5.5-1: Store Data procedure

### 9.5.6 Modify Data procedure

The Modify Data procedure in the Near-RT RIC allows the xApp to request SDL to update or delete data from database.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | XApp intends to modify or delete data stored by the SDL. |
| Actors and Roles | * xApp: Originator of Modify Data request; * SDL: Response to the request of xApp. |
| Pre-conditions | * SDL API services have been registered by API Enablement; * xApp has successfully registered as a client to the SDL before; * xApp has been authorized to write the data in the database. |
| Begins when | xApp determines to modify relevant data stored by the database in the Near-RT-RIC platform. |
| Step 1 (M) | xApp sends a request to SDL to modify related information, including update and delete data in database. |
| Step 2  [Informative] | SDL modifies or deletes xApp requested information in database (internal implementation). |
| Step 3 (M) | SDL responds to the xApp, including success and failure operation. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL” as SDL

Database "Database" as DB

endbox

xApp -> SDL: 1. Modify data request

SDL <--> DB: 2. Modify related data

SDL -> xApp: 3. Modify data response

@enduml

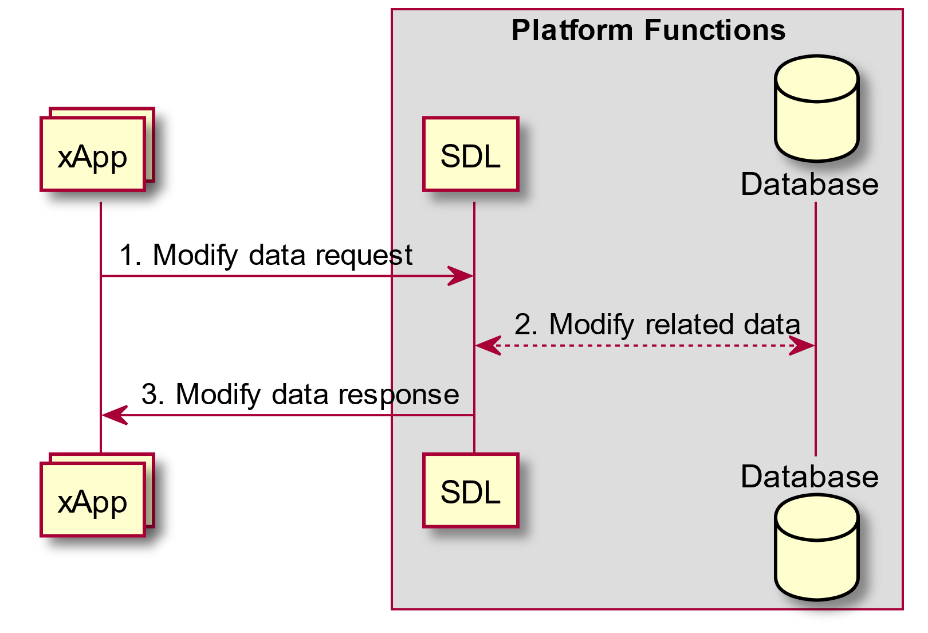


Figure 9.5.6-1: Modify Data procedure

### 9.5.7 Subscribe/Push procedure

The Subscribe/Push procedure in the Near-RT RIC allows the xApp to subscribe to the SDL for authorized data changes in the database. The SDL will send the xApp both the type of information changes (meta-data) and, in the same message, the updated data itself.

The pushes are generated only for changes to the information after the subscription.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | xApp subscribes with the SDL for data changes and receives related push messages. |
| Actors and Roles | * xApp: Originates the Subscribe request; * SDL: Responds to the xApp request and pushes the subscribed data and meta-data to the xApp. |
| Pre-conditions | * SDL API services have been registered by API Enablement; * xApp has successfully registered as a client to the SDL. |
| Begins when | xApp subscribes for push messages related to information changes in the SDL in the Near-RT RIC platform. |
| Step 1 (M) | xApp sends the request to SDL to subscribe for push messages related to information changes. |
| Step 2 (M) | SDL responds to the xApp subscription. |
| Step 3 (M) | SDL sends the push message with data and meta-data for the subscribed information changes to the xApp. |

@startuml

skin rose

skinparam ParticipantPadding 10

skinparam BoxPadding 5

skinparam lifelineStrategy solid

Collections “xApp” as xApp

Box “Platform Functions”

Participant “SDL + Database” as SDL

endbox

xApp -> SDL: 1. Subscribe information request

SDL -> xApp: 2. Subscribe information response

SDL -> xApp: 3. Push (data + meta-data)

@enduml

Text, table

Description automatically generated with medium confidence

Figure 9.5.7-1: Subscribe/Push procedure

### 9.5.8 Use cases of SDL APIs

#### 9.5.8.1 Subscribe/Notify for E2NodeInfo and E2NodeList

The SDL Subscribe/Notify procedure (section 9.5.4) along with the SDL fetch procedure (section 9.5.3) may be used for getting notifications and fetching E2 Node related information such as E2NodeList (section 9.5.9.2) and E2NodeInfo (section 9.5.9.3) changes (obtained as the result of an E2AP Global Procedure such as: E2 Setup, Reset, E2 Removal, RIC Service Update or E2 Node Configuration Update).

As noted in section 9.5.4, notifications will be generated only for changes to E2NodeInfo after the subscription. To get the complete state of E2NodeInfo, the xApp may do a fetch.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution / Specification** |
| Goal | Inform the xApp about changes to E2 Node information |
| Actors and Roles | * E2 Nodes: O-CU, O-DUs supporting the E2 interface. * xApps: Requires information about changes to E2 Node information. * Near-RT RIC:   + SDL: Shared Data Layer exposing an SDL API to the xApps;   + E2 Termination: Termination of E2 Node connections. |
| Pre-conditions | * SDL API services have been registered and authorized by API Enablement; * xApp has successfully registered as a client to the SDL before; * xApp has been authorized to read the data in the database. |
| Begins when | xApp decides it is interested in E2NodeInfo updates. |
| Step 1 (M) | xApp sends request to SDL to subscribe for notifications related to E2NodeInfo changes. |
| Step 2 (M) | SDL responds to xApp subscription. |
| Step 3 (M) | E2 Node or Near-RT RIC initiates an E2AP Global procedure (i.e. E2 Setup, Reset, E2 Removal, RIC Service Update or E2 Node Configuration Update). |
|  | [OPT] Steps 4-10 are executed if the E2AP Global Procedure succeeds: |
| Step 4 [Informative] | E2 Termination updates the Database (internal implementation). |
| Step 5 [Informative] | Database change triggers SDL procedure. |
| Step 6 [Informative] | SDL retrieves list of xApps that would be interested in this update (internal implementation). |
|  | [LOOP] Steps 7-10 are looped for each relevant xApp. |
| Step 7 (M) | SDL sends a notification of subscribed E2NodeList update or E2NodeInfo update to the xApp which includes the E2NodeID. |
| Step 8 [Informative] | xApp makes decision on whether to fetch the updated information from the SDL (xApp internal implementation) |
|  | [OPT] Steps 9-10 are executed if the xApp decides that it needs the update: |
| Step 9 (M) | xApp sends a fetch request to the SDL which includes the E2NodeID. |
| Step 10 (M) | SDL responds with a fetch response (E2NodeList or E2NodeInfo). |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections "xApp" as xApp

box "Platform functions"

Participant "SDL + Database" as SDL

Collections "E2 Termination" as E2T

endbox

Collections "E2 Node" as ran

xApp -> SDL: 1. <SDL API> Subscribe Information Request (Information type, Notify, Filter(O))

SDL -> xApp: 2. <SDL API> Subscribe Information Response

ran <-> E2T: 3. <<E2>> Global procedure

opt Handling of E2 global procedure succeeds

E2T --> SDL: 4. Update E2NodeList and/or E2NodeInfo

SDL -> SDL: 5. Initiate SDL procedure

SDL --> SDL : 6. Retrieve list of xApps relevant for this update

Loop for each xApp in relevant xApps

SDL -> xApp: 7. <SDL API> Notification (Information type, Notification unit)

xApp -> xApp: 8. Decide whether to fetch the update from SDL

Opt xApp fetches E2NodeInfo

xApp -> SDL: 9. <SDL API> Fetch Request (Information type, Filter)

SDL -> xApp: 10. <SDL API> Fetch Response (Information type, Information Unit)

End

End

End

@enduml

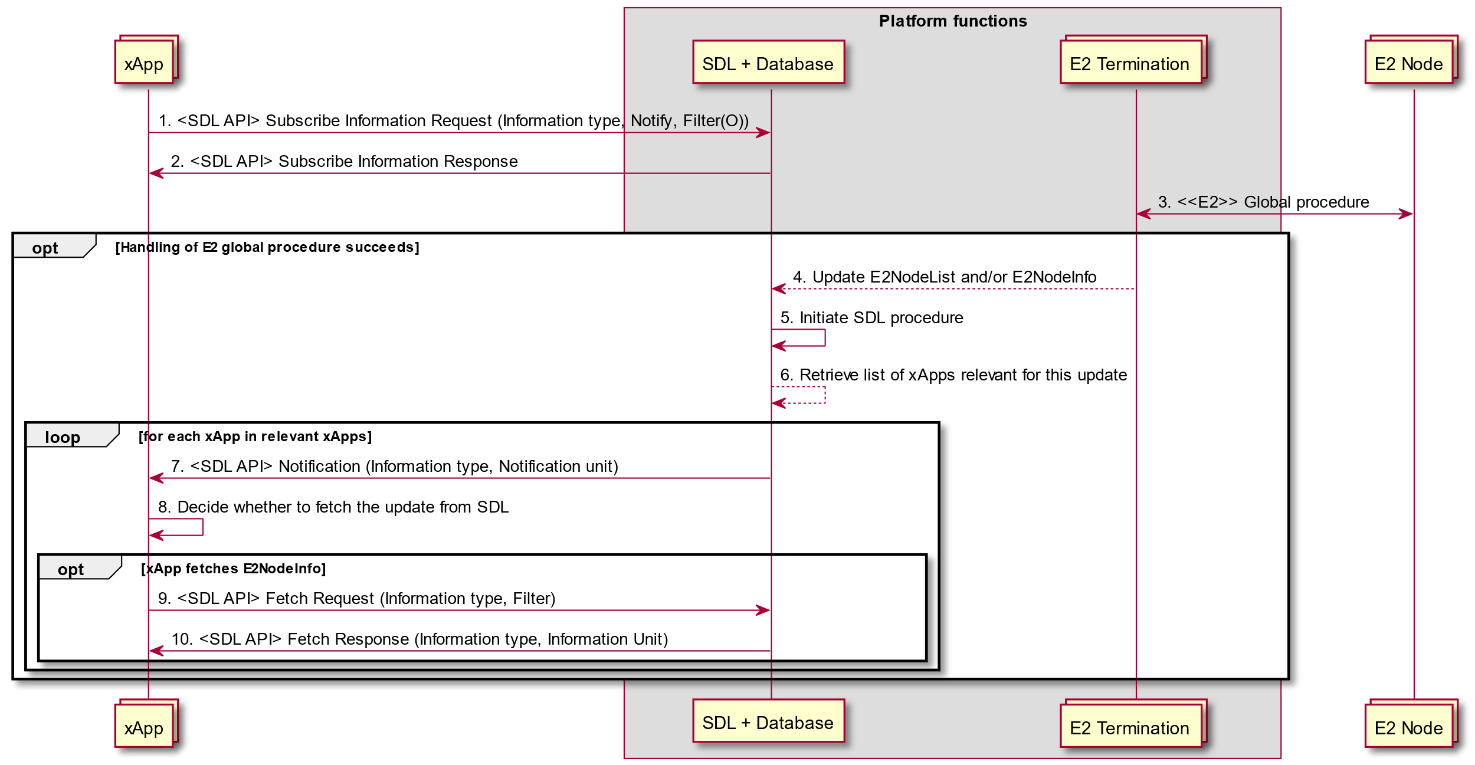


Figure 9.5.8.1-1: Subscribe/Notify procedures for E2NodeInfo and E2NodeList

#### 9.5.8.2 Subscribe/Push for E2NodeInfo and E2NodeList

The SDL Subscribe/Push procedure (section 9.5.7) may be used for getting E2 Node related information pushes such as E2NodeList (section 9.5.9.2) and E2NodeInfo (section 9.5.9.3) changes (obtained as the result of an E2 Global procedure such as E2 Setup, Reset, E2 Removal, RIC Service Update or E2 Node Configuration Update).

As noted in section 9.5.7, the pushes will be generated only for changes to E2 Node related information after the subscription. To get the complete state of E2 Node related information, the xApp may do a fetch.

|  |  |
| --- | --- |
| **Use Case Stage** | **Evolution/Specification** |
| Goal | Inform the xApp about changes to E2 Node information. |
| Actors and Roles | * E2 Nodes: O-CU, O-DUs supporting the E2 interface; * xApps: Requires information about changes to E2 Node information; * Near-RT RIC:   + SDL: Shared data layer exposing an SDL API to the xApps;   + E2 Termination: Termination of E2 node connections. |
| Pre-conditions | * SDL API services have been registered and authorized by API Enablement; * xApp has successfully registered as a client to the SDL before; * xApp has been authorized to read the data in the database. |
| Begins when | xApp decides it is interested in E2NodeInfo updates. |
| Step 1 (M) | xApp sends request to SDL to subscribe for pushes related to E2NodeList and/or E2NodeInfo changes. |
| Step 2 (M) | SDL responds to xApp subscription. |
| Step 3 (M) | E2 Node or Near-RT RIC initiates an E2AP Global Procedure (i.e. E2 Setup, Reset, E2 Removal, RIC Service Update or E2 Node Configuration Update). |
|  | [OPT] Steps 4-7 are executed if the E2AP Global procedure succeeds: |
| Step 4 [Informative] | E2 Termination updates the Database (internal implementation). |
| Step 5 [Informative] | Database change triggers SDL procedure. |
| Step 6 [Informative] | SDL retrieves list of xApps that would be interested in this update (internal implementation). |
| Step 7 (M) | For each relevant xApp, SDL sends a push of E2NodeList or E2NodeInfo update to the xApp which includes the E2NodeID. |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam lifelineStrategy solid

Collections "xApp" as xApp

box "Platform functions"

Participant "SDL + Database" as SDL

Collections "E2 Termination" as E2T

endbox

Collections "E2 Node" as ran

xApp -> SDL: 1. <SDL API> Subscribe Information Request (Information type, Push, Filter(O))

SDL -> xApp: 2. <SDL API> Subscribe Information Response

ran -> E2T: 3. <<E2>> Global Procedure

opt Handling of E2 global procedure succeeds

E2T --> SDL: 4. Update E2NodeList and/or E2NodeInfo

SDL -> SDL: 5. Initiate SDL procedure

SDL --> SDL : 6. Retrieve list of xApps relevant for this update

Loop for each xApp in relevant xApps

SDL -> xApp: 7. <SDL API> Push (Information type, Information unit)

End

End

@enduml

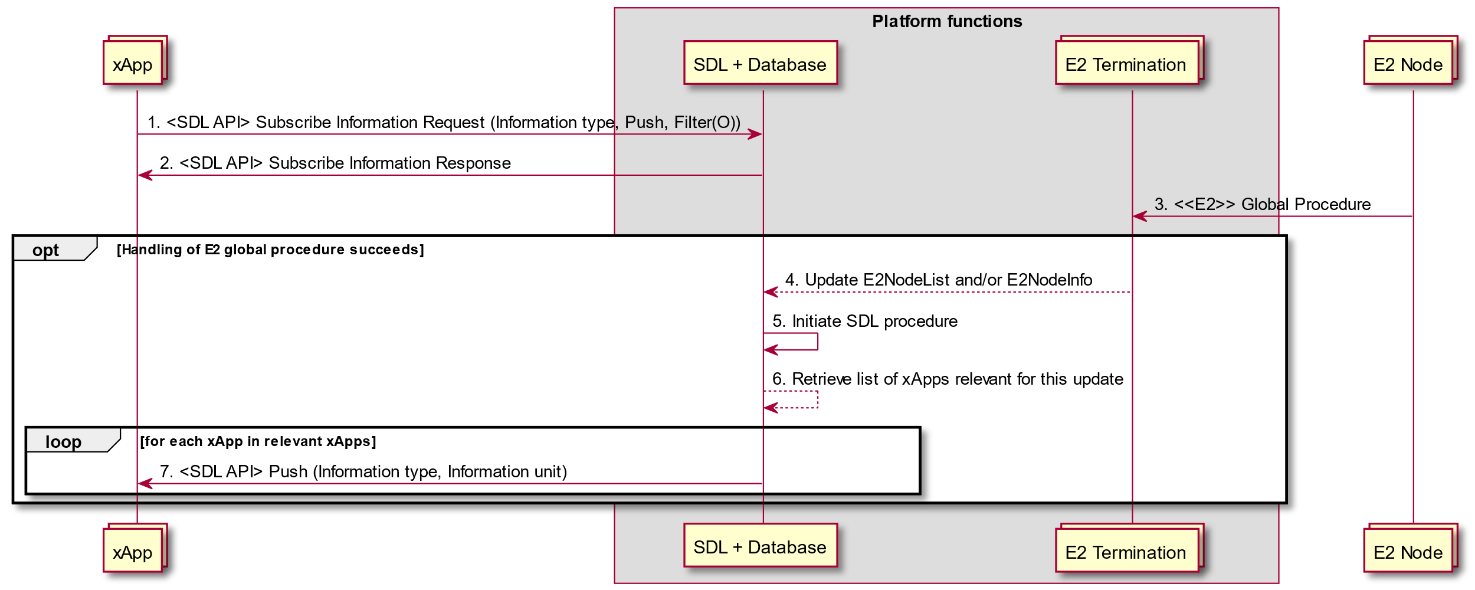


Figure 9.5.8.2-1: Subscribe/Push procedures for E2NodeInfo and E2NodeList

### 9.5.9 SDL API data structures

#### 9.5.9.1 Overview

SDL API data structures are used to inform xApps of change to E2 Node state and information as the result of E2AP Global Procedures.

The following data structures are defined:

* E2NodeList: List of known E2 Node ID and the associated E2 Node State
* E2NodeInfo: E2 Node component configuration list and RAN function list

The relationship between these data structures and E2AP Global Procedures is described in table 9.5.9.1-1.

Table 9.5.9.4-2 presents the actions that xApp may perform under different E2 Node State conditions.

NOTE 1: E2AP Error Indication procedure does not result in a change to SDL API data as most error cases are assumed to be handled internally within the Near-RT RIC platform. An appropriate E2 Related API (Failure, Cause) message may be used to inform xApp if required.

NOTE 2: E2AP Reset procedure handling is assumed to be assisted by an appropriate platform mechanism with the Near-RT RIC first using the appropriate SDL API mechanism (see section 9.5.8) to inform xApp that a RESET has occurred (i.e. E2 Node State = "Reset") and then a second SDL API mechanism is used to inform xApp that the E2 Node is again available (i.e. E2 Note State = "Connected"). It is assumed that the xApp Subscription Management function will delete all associated subscription records during the period between these two SDL API mechanisms and that xApp will use the information E2 Node is again available (i.e. E2 Note State = "Connected") to trigger any required E2 Subscription (Request) messages to re-establish E2 subscriptions.

NOTE 3: E2AP E2 Removal procedure handling and E2 link failure handling are assumed to be assisted by an appropriate platform mechanism with the Near-RT RIC using the appropriate SDL API mechanism (see section 9.5.8) to inform xApp that the E2 Node is not available (i.e. E2 Node State = "Disconnect").

**Table 9.5.9.1-1: SDL API handling of E2AP Global procedures**

| **Global procedure** | **E2NodeList change** | **E2NodeInfo change** |
| --- | --- | --- |
| E2 Setup | **Add E2NodeList record** (E2 Node ID, E2 Node State = "Connected") | **Add E2NodeInfo record** (E2 Node ID, RAN Function list, E2 Node component configuration list) |
| Reset | **Change E2NodeList record** (E2 Node State = "Reset")**.**  After using an appropriate Near-RT RIC mechanism, **Change E2NodeList record** (E2 Node State to "Connected") | - |
| RIC Service Update | - | **Add, Modify and/or Remove E2NodeInfo record** (E2 Node ID, RAN Function List) |
| E2 Node Configuration Update | - | **Add, Modify and/or Remove E2NodeInfo record** (E2 Node ID, E2 Node Component configuration list) |
| E2 Removal | **Change E2NodeList record** (E2 Node State = "Disconnected").  After **Delete E2NodeInfo record**,  **Delete E2NodeList record** (E2 Node ID) | **Delete E2NodeInfo record** (E2 Node ID) |
| E2 link failure | **Change E2NodeList record** (E2 Node State = "Disconnected").  After **Delete E2NodeInfo record**,  **Delete E2NodeList record** (E2 Node ID) | **Delete E2NodeInfo record** (E2 Node ID) |

**Table 9.5.9.1-2: Expected xApp responses to E2 Node State**

| **E2 Node State** | **xApp actions** |
| --- | --- |
| Connected | Discover E2 Node ID in E2NodeList and obtain E2NodeInfo (after E2 Setup).  Initiate E2 Related APIs (after E2 Setup and Reset). |
| Reset | Delete internal records of ongoing E2 Related API procedures.  Monitor for changes in E2 Node State. |
| Disconnected | Delete internal records of ongoing E2 Related API procedures.  Delete internal records of E2 Node ID. |

#### 9.5.9.2 E2NodeList

The E2NodeList data structure contains a list of known E2 Nodes and the associated E2 Node State. Each record contains:

* E2 Node ID
* E2 Node State ("Connected", "Reset", "Disconnected")

E2NodeList records may be deleted when the corresponding E2NodeInfo record has been deleted. It is recommended to maintain this record for sufficient time for all xApps to complete clean-up mechanisms.

#### 9.5.9.3 E2NodeInfo

The E2NodeInfo data structure contains information on a given E2 Node. Each record contains:

* E2 Node ID
* List of declared E2 Node component configuration information
* List of declared RAN Functions

E2NodeInfo records shall be deleted after corresponding E2NodeState is set to "Disconnected" and sufficient time has been allowed for all xApps to complete clean-up mechanisms.

## 9.6 API Enablement Procedures

### 9.6.1 Near-RT RIC API Discovery procedure

This procedure enables the xApp(s) to discover the Near-RT RIC APIs which are offered by the Near-RT RIC platform.

|  |  |  |
| --- | --- | --- |
| **Use Case Stage** | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | To allow the xApp to discover which APIs are provided by the Near-RT RIC platform and how to access them. |  |
| Actors and Roles | * xApp: Requester of the available Near-RT RIC APIs in RIC platform and potential Near-RT RIC API consumer. * API Enablement function: providing the Near-RT RIC API information to the xApps. * One or more Near-RT RIC platform functions * Database |  |
| Assumptions | For initiating the discovery, the xApp has registered to Near-RT RIC platform and is aware of the Enablement API information. |  |
| Pre conditions | * The xApp is deployed to Near-RT RIC platform. * The Enablement API information has been provided to the xApp (as pre-configuration by SMO or by other means. * One or more Near-RT RIC platform functions are configured to communicate with API Enablement function and have previously sent Enablement information to API Enablement function |  |
| Begins when | xApps wants to discover the APIs which are provided by the Near-RT RIC platform (via API Enablement function). |  |
| Step 1 (M) | xApp sends a Near-RT RIC API discovery request to the API Enablement function. It includes the xApp identity and includes query information. Query information may include criteria for discovering matching APIs (e.g. API type, Serving Area Information, interfaces, protocols). |  |
| Step 2  [Informative] | API Enablement function verifies the identity of the xApp. |  |
| Step 3 [Informative] | API enablement function may retrieve the stored Near-RT RIC API(s) information, as per the query information in the discovery request. |  |
| Step 4 [Informative] | API Enablement function applies the discovery policy and performs filtering of the retrieved Near-RT RIC APIs information. |  |
| Step 5 (M) | The API Enablement function sends a Near-RT RIC API discovery response to the xApp with the information about the APIs for which the xApp has the required authorization. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

Collections xApp as "xApp"

Box “Near-RT RIC Platform”

participant database as “Database”

participant ams as “API Enablement”

collections func as “Near-RT RIC platform function”

endbox

func <--> ams : Near-RT RIC platform's API information

ams <--> database : Store API information

...

xApp -> ams : 1. Near-RT RIC API discovery request

ams --> ams: 2. API Enablement verifies the identity of the xApp

ams <--> database : 3. API enablement may retrieve stored \nevent subscription

ams --> ams : 4. API Enablement applies discovery policy \nand information filtering

ams -> xApp : 5. Near-RT RIC API discovery response

@enduml

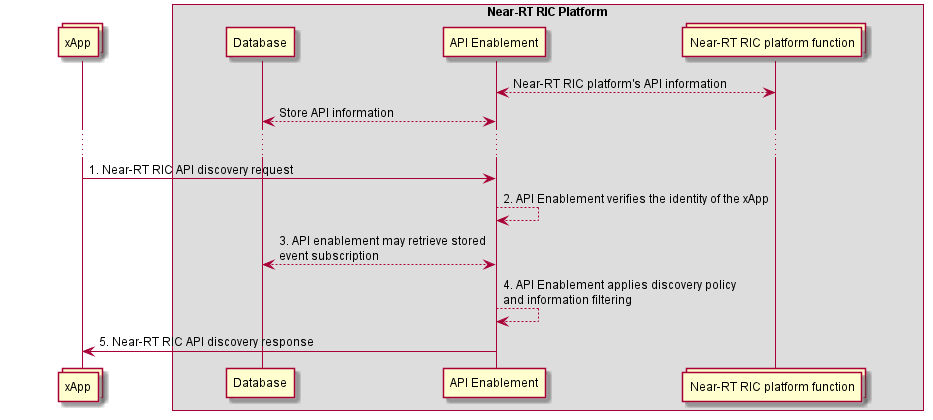


Figure 9.6.1-1: Near-RT RIC API Discovery procedure

### 9.6.2 Procedures related to API Event Subscription and Notification

In this section, the procedures for subscription / subscription deletion and API event notification are provided. These procedures allow the xApps to receive information on events.

Such events correspond to Near-RT RIC API related events like the following:

|  |  |
| --- | --- |
| **Events** | **Events Description** |
| Availability of Near-RT RIC APIs | Availability events of Near-RT RIC APIs (e.g. active, inactive) |
| Near-RT RIC API updated | Events related to change in Near-RT RIC API information |

#### 9.6.2.1 Procedure for Event Subscription

The purpose of this procedure is to enable xApps to subscribe to monitor events related to the provided Near-RT RIC APIs.

|  |  |  |
| --- | --- | --- |
| **Use Case Stage** | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | To allow the xApp to subscribe to receive Near-RT RIC API related event notifications. |  |
| Actors and Roles | * xApp: Requester to subscribe to API enablement to monitor events related to the near-RT RIC APIs; * API enablement function: Providing the subscription to Near-RT RIC API events. * Database |  |
| Assumptions | * For initiating the subscription, the xApp has registered to RIC platform and is aware of the Enablement API info. Also, xApp has discovered the provided Near-RT RIC APIs. |  |
| Pre-conditions | * The xApp is registered to Near-RT RIC platform; * The Enablement API information has been provided to the xApp (as pre-configuration by SMO; * The xApp has discovered the registered Near-RT RIC APIs (see 9.6.1). |  |
| Begins when | xApp has discovered the Near-RT RIC APIs as provided by the Near-RT RIC platform and wants to subscribe for event notifications. |  |
| Step 1 (M) | xApp sends an event subscription request to the API Enablement in order to receive notification of events. This request includes the subscribing xApp identifier as well as the event criteria:   * Event criteria may include event type information, e.g. API failure event, new API available event, API unavailable event, API version change event, API location change event, etc. |  |
| Step 2  [Informative] | The API enablement checks whether the xApp is authorized to monitor the requested event(s). |  |
| Step 3 [Informative] | The API enablement stores the approved subscription information. |  |
| Step 4 (M) | The API enablement sends an API event subscription response to the xApp. This response indicates the success or failure of the event subscription operation. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

Collections xApp as "xApp"

Box “Near-RT RIC Platform”

participant database as “Database”

participant ams as “API enablement”

endbox

xApp -> ams : 1. API event subscription request

ams --> ams : 2. Check authorization for API event subscription

ams <--> database : 3. store event subscription

ams -> xApp : 4. API event subscription response

@enduml

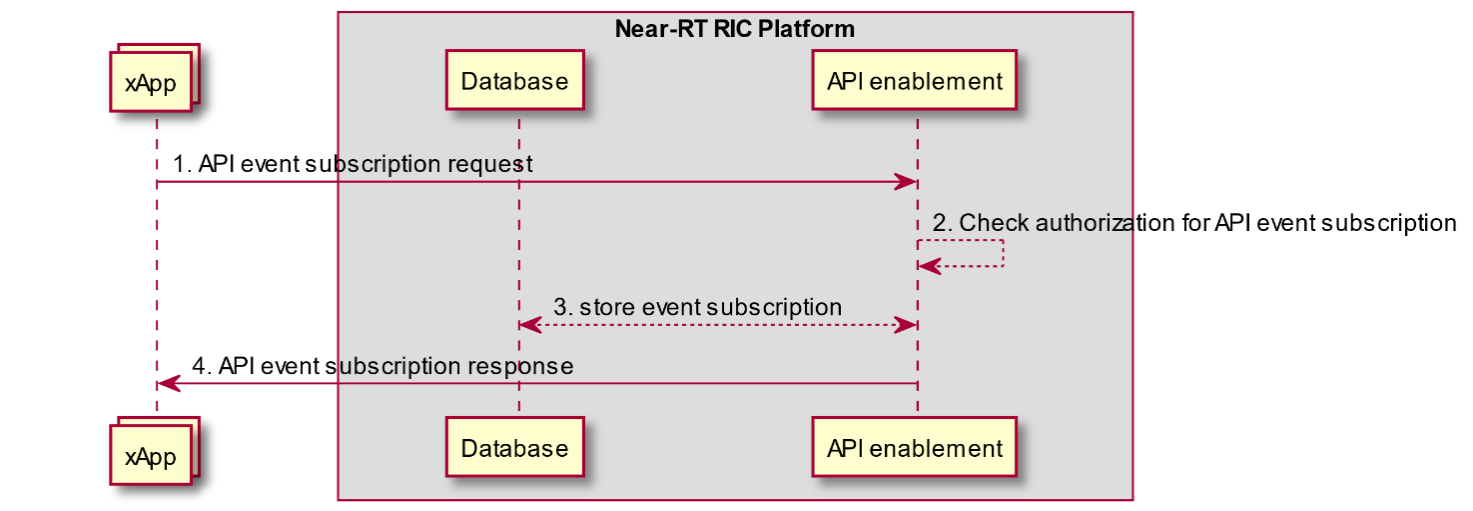


Figure 9.6.2.1-1: Subscription for API-related events

#### 9.6.2.2 Procedure for Event Subscription Delete

The purpose of this procedure is to enable xApps to un-subscribe from events monitoring related to the provided near-RT RIC APIs.

|  |  |  |
| --- | --- | --- |
| **Use Case Stage** | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | To allow the xApp to un-subscribe from Near-RT RIC API related event notifications. |  |
| Actors and Roles | * xApp: Requester to un-subscribe to API Enablement from monitoring events related to the Near-RT RIC APIs; * API Enablement: Removing the subscription of xApp to Near-RT RIC API events. * Database |  |
| Assumptions | For initiating the subscription delete, the xApp has registered to Near-RT RIC platform and is aware of the Enablement API information. Also, xApp has subscribed to events. |  |
| Pre-conditions | The xApp has subscribed to Near-RT RIC API related events (see 9.6.2.1). |  |
| Begins when | xApp wants to un-subscribe for Near-RT RIC API related events. |  |
| Step 1 (M) | xApp sends an event subscription delete request to the API Enablement with the information of the subscribed event and includes the xApp identifier and the event subscription identifier. |  |
| Step 2  [Informative] | The API Enablement, upon receiving the event subscription delete request from the xApp, checks for the event subscription corresponding to the xApp and further checks if the subscribing entity is authorized to un-subscribe. |  |
| Step 3 [Informative] | API Enablement deletes the subscription information for the xApp. |  |
| Step 3 (M) | The API Enablement sends an API event un-subscription response to the xApp. This response indicates the success or failure of the event un-subscription operation. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

Collections xApp as "xApp"

Box “Near-RT RIC Platform”

participant database as “Database”

participant ams as “API enablement”

endbox

xApp -> ams : 1. API event subscription delete request

ams --> ams : 2. Check authorization for API event subscription

ams <--> database : 3. Delete event subscription

ams -> xApp : 4. API event subscription delete response

@enduml

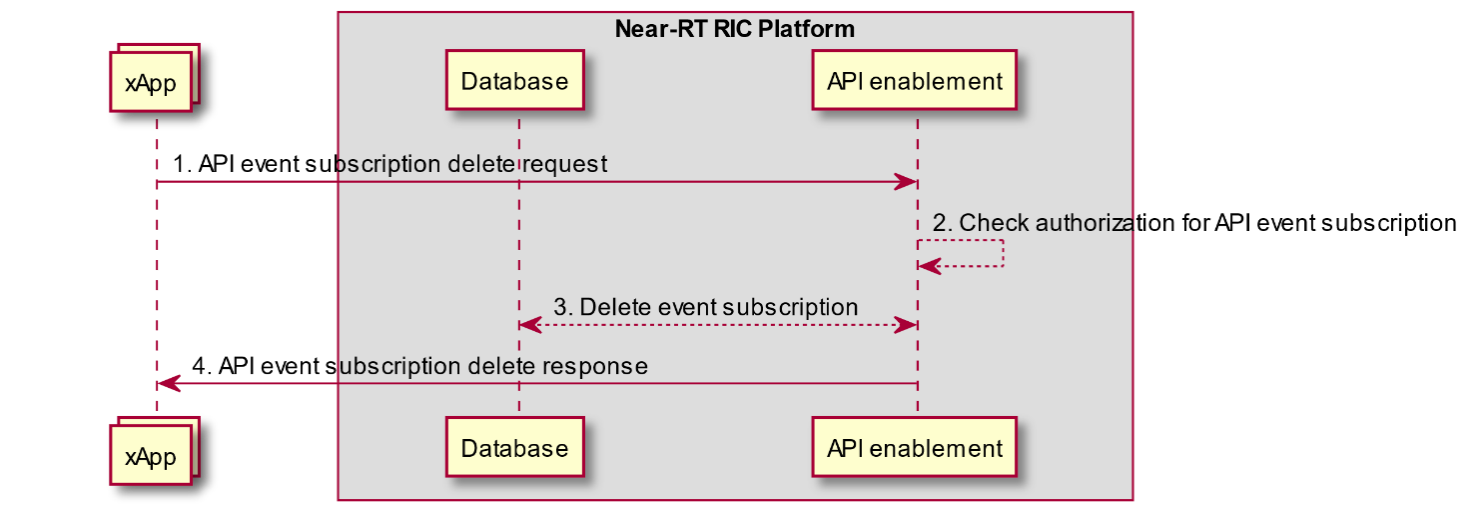


Figure 9.6.2.2-1: Subscription Delete for API-related events

#### 9.6.2.3 Procedure for Event Notification

This procedure shows sending the event notification to the subscribed xApp, based on a trigger event captured at the API Enablement.

|  |  |  |
| --- | --- | --- |
| **Use Case Stage** | **Evolution / Specification** | **<<Uses>>**  **Related use** |
| Goal | To notify the xApp on events related to the Near-RT RIC APIs, based on subscription. |  |
| Actors and Roles | * xApp: Receiving event notifications based on subscription * API Enablement: Generating/ triggering event notifications and sending them to the subscribed xApps * One or more Near-RT RIC platform functions * Database |  |
| Assumptions | For receiving a notification, the xApp has registered to Near-RT RIC platform and has subscribed to events. |  |
| Pre-conditions | The xApp has subscribed to Near-RT RIC API related events (see 9.6.2.1). |  |
| Begins when | A trigger event criterion is met at the API Enablement (i.e. enablement information update from a Near-RT RIC platform function). Such criterion has been configured during subscription of the xApp. |  |
| Step 1 [Informative] | API Enablement may retrieve the list of corresponding event subscriptions |  |
| Step 2 [Informative] | API Enablement generates an event related to APIs, which is to be consumed by the subscribed xApp(s) |  |
| Step 3 (M) | The API Enablement sends event notifications to all the subscribing xApp(s) that have subscribed for the event matching the criteria. |  |
| Step 4 (O) | The xApp may send an event notification acknowledgement to API Enablement. It depends on the capabilities of the transport used for notifications whether or not such acknowledgement can be sent. |  |

@startuml

skin rose

skinparam ParticipantPadding 50

skinparam BoxPadding 10

skinparam defaultFontSize 12

skinparam lifelineStrategy solid

Collections xApp as "xApp"

Box “Near-RT RIC Platform”

participant database as “Database”

participant ams as “API enablement”

collections func as “Near-RT RIC platform function”

endbox

func <--> ams : Near-RT RIC function enablement \ninformation update

ams <--> database : Store api information

...

ams <--> database: 1. API enablement may retrieve stored \nevent subscription information

ams -> ams : 2. Event notification information \ngeneration

ams -> xApp : 3. API Event notification

ams <-- xApp : 4. API Event notification ACK

@enduml

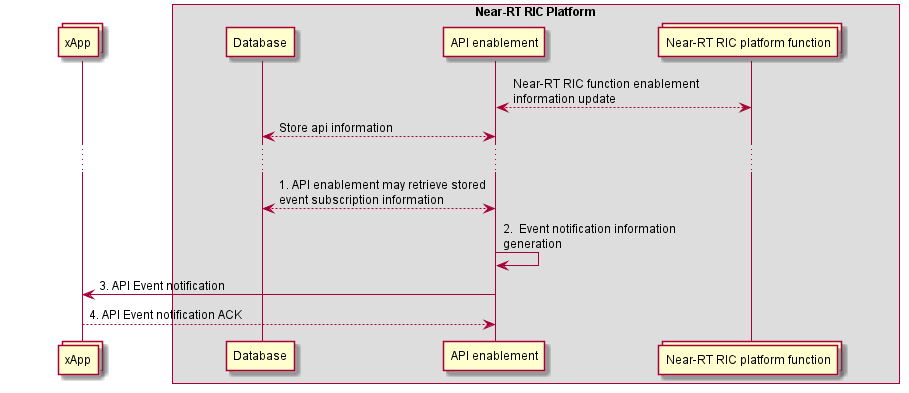


Figure 9.6.2.3-1: API Event Notification

# Revision history

|  |  |  |
| --- | --- | --- |
| Date | Revision | Description |
| 2022.09.16 | 03.00.00 | Intial baseline |
| 2022.11.11 | 03.00.01 | Addition of CRs:  <CMCC-WG3-CR-0005-RICARCH-Add Y1 descriptions for Near-RT RIC architecture-v2.docx>  <NOK.AO-2022.10.28-WG3-CR-0017-RICARCH 5.1.3 Management API Requirements\_rev2.docx>  <NOK.AO-2022.09.28-WG3-CR-0014-RICARCH 9.4.5 Management APIs\_rev5.docx>  <NOK.AO-2022.09.28-WG3-CR-0015-RICARCH 9.4.6 Management APIs\_rev4.docx>  <NOK.AO-2022.09.28-WG3-CR-0016-RICARCH 9.4.7 Management APIs\_rev5.docx>  Minor editorial clean-ups. |
| 2022.11.17 | 03.00.02 | Changes reflecting remarks received during WG3 approval process  - Update copyright year 2023, and release number R003.  - Fix xApp/Near-RT RIC platform in the “Actors and Roles” of tables in 9.4.8 and 9.4.13.  - Fix the labels for a few steps in 9.4.8, 9.4.12 and 9.4.13 as “[informative]” instead of “(M)”.  - Add missing step descriptions in 9.4.8.  - Other minor editorial clean-ups. |
| 2022.11.20 | 04.00 | Version incremented to 04.00 for TSC. |

# History

|  |  |  |
| --- | --- | --- |
| Date | Revision | Description |
| 2022.08.01 | 02.01.09 | Published as Final version 03.00 |
| 2021.11.08 | 02.01.04 | Published as Final version 02.01 |
| 2021.03.16 | 02.00.05 | Published as Final version 02.00 |
| 2020.11.25 | 01.00.05 | Published as Final version 01.01 |
| 2020.02.17 | 00.01.10 | Published as Final version 01.00 |