O-RAN.WG2.Non-RT-RIC-ARCH-TS-v02.01

Technical Specification

O-RAN Working Group 2 (Non-RT RIC and A1 interface WG)

Non-RT RIC Architecture

Contents

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 technical specification for the Non-RT RIC (RAN Intelligent Controller) architecture. This document collects requirements on the Non-RT RIC framework, hosted applications, and services of the R1 interface. This document specifies Non-RT RIC framework functionalities and services exposed to the applications.

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.WG1.O-RAN-Architecture-Description: “O-RAN Working Group 1; O-RAN Architecture Description”.

[3] O-RAN.WG2.Use-Case-Requirements: “O-RAN Working Group 2; Use Case and Requirements”.

[4] O-RAN.WG2.A1GAP: “O-RAN Working Group 2; A1 interface: General Aspects and Principles”.

[5] O-RAN.WG2.A1AP: “O-RAN Working Group 2; A1 interface: Application Protocol”.

[6]	O-RAN.WG2.A1TP: “O-RAN Working Group 2; A1 interface: Transport Protocol”.
[7]	O-RAN.WG2.A1TD: “O-RAN Working Group 2; A1 interface: Type Definitions”.
[8]	O-RAN.WG10.OAM-Architecture: “O-RAN Working Group 10; O-RAN Operations and Maintenance Architecture”.
[9]	O-RAN.WG2.R1GAP: “O-RAN Working Group 2; R1 interface: General Aspects and Principles”.

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.

Not Applicable.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply.
A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

Functions anchored inside the Non-RT RIC framework: Logical functions in the SMO framework that are part of the Non-RT RIC framework.

NOTE: The definition makes no assumption on the mandatory or optional qualifier of the function being anchored.

Functions anchored outside the Non-RT RIC framework: Logical functions in the SMO framework that are not part of the Non-RT RIC framework.

NOTE: The definition makes no assumption on the mandatory or optional qualifier of the function being anchored.

Non-RT RIC (O-RAN Non-Real-Time RAN Intelligent Controller): a logical function within SMO that enables non-real-time control and optimization of RAN elements and resources, AI/ML workflow including model training and updates, and policy-based guidance of applications/features in Near-RT RIC.

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 (e.g. UE basis, Cell basis) data collection and actions over E2 interface.

Non-anchored functions: Logical functions in the SMO framework that may or may not be part of the Non-RT RIC framework.

NOTE: The definition makes no assumption on the mandatory or optional qualifier of the function being non-anchored.

R1 Data Producer: An entity capable of producing data which can be consumed by other entities and of registering for discovery information about the types of these data in the context of the data management and exposure service.

R1 Data Consumer: An entity capable of consuming data originating from other entities in the context of the data management and exposure service.

3.2 Symbols

Void.

3.3 Abbreviations

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

AI	Artificial Intelligence
EI	Enrichment Information
ML	Machine Learning
Near-RT RIC	Near-real-time RAN Intelligent Controller
Non-RT RIC	Non-real-time RAN Intelligent Controller
RAN	Radio Access Network
SMO	Service Management and Orchestration

4 Overview

4.1 Non-RT RIC in O-RAN Overall Architecture

Non-Real Time RAN Intelligent Controller (Non-RT RIC) is the functionality internal to the Service Management and Orchestration (SMO) framework in O-RAN overall architecture, as shown in Figure 4.1-2 of O-RAN.WG1.O-RAN-Architecture-Description [2].

- Non-RT RIC represents a subset of functionality of the SMO framework.
- Non-RT RIC supports intelligent RAN operation and optimization.
- Non-RT RIC logically terminates the A1 interface, and it provides policy-based guidance, enrichment information, and AI/ML model management [FFS] to the Near-RT RICs.
- Non-RT RIC can access other SMO framework functionalities, for example influencing what is carried across the O1 and O2 interface.

4.2 Non-RT RIC Composition

Non-RT RIC is comprised of:

- Non-RT RIC Framework – Functionality internal to the SMO Framework that:

    - Logically terminates the A1 interface to the Near-RT RIC.
    - Exposes set of R1 services to Non-RT RIC Applications (rApps).
- Non-RT RIC Applications (rApps) – Applications that leverage the functionalities available in the Non-RT RIC Framework / SMO Framework to provide value added services related to RAN operation and optimization. The scope of rApps includes, but is not limited to, radio resource management, data analytics, and providing enrichment information.

5 Requirements

5.1 Requirements for the Non-RT RIC Framework

[REQ-NRTFWK-FUN1]	The Non-RT RIC framework shall support functionality to register services along with their service producers in the Non-RT RIC and SMO.

[REQ-NRTFWK-FUN2]	The Non-RT RIC framework shall support functionality to allow service consumers to discover services.

[REQ-NRTFWK-FUN3]	The Non-RT RIC framework shall support functionality to allow service consumers to subscribe/unsubscribe notifications about newly registered/updated/deregistered services.

[REQ-NRTFWK-FUN4]	The Non-RT RIC framework shall support functionality to notify subscribed service consumers about newly registered/updated/deregistered services.

[REQ-NRTFWK-FUN5]	The Non-RT RIC framework shall support functionality to authenticate service consumers.

[REQ-NRTFWK-FUN6]	The Non-RT RIC framework shall support functionality to authorize service consumers to access services.

[REQ-NRTFWK-FUN7]	The Non-RT RIC framework shall support functionality to send messages to and receive messages from the Near-RT RIC via the A1 interface.

[REQ-NRTFWK-FUN8]	The Non-RT RIC framework shall support functionality to allow Data Consumers (including rApps) to register data types they consume, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN9]	The Non-RT RIC framework shall support functionality to allow Data Producers (including rApps) to register data types they produce, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN10]	The Non-RT RIC framework shall support functionality to allow Data Consumers (including rApps) to subscribe/request instances of registered data types for consumption, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN10a]	The Non-RT RIC framework shall support functionality to allow Data Producers (including rApps) to offer instances of registered data types for collection and storage, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN11]	The Non-RT RIC framework shall support functionality to train AI/ML models, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN12]	The Non-RT RIC framework shall support functionality to allow service consumers to store and retrieve trained AI/ML models, if such functionality is not supported in the SMO framework

[REQ-NRTFWK-FUN13]	The Non-RT RIC framework shall support functionality to monitor the performance for deployed AI/ML models in runtime, if such functionality is not supported in the SMO framework

[REQ-NRTFWK-FUN14]	The Non-RT RIC framework may support functionality to collect external enrichment information from external enrichment information sources.

[REQ-NRTFWK-FUN15]	The Non-RT RIC framework may support functionality to retrieve trained ML models (and metadata) from external AI/ML service providers.

[REQ-NRTFWK-FUN16]	The Non-RT RIC framework may support functionality to allow external sources to inject RAN intents, suspend/resume/check rApps, and configure/check/initiate/suspend/resume/terminate AI/ML training processes.

[REQ-NRTFWK-FUN17]	The Non-RT RIC framework shall support functionality to consolidate the alarm information from multiple managed entities, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN18]	The Non-RT RIC framework shall support functionality to store alarm acknowledgement state for alarms originating from managed entities that do not support alarm acknowledgement state change, if such functionality is not supported in the SMO framework.

[REQ-NRTFWK-FUN19]	The Non-RT RIC framework shall have the capability to identify the potentially applicable Near-RT RIC(s) for A1 policy creation if the Near-RT RIC identifier is absent in the create A1 policy request received from the rApp.

NOTE: Selection of potentially applicable Near-RT RIC(s) may be based, e.g., on the content of the received create A1 policy request or on configuration.

5.2 Requirements for R1

O-RAN.WG2.Use-Case-Requirements [3] specifies R1 interface functional requirements.

6 Non-RT RIC Architecture

The following figure depicts the reference architecture of the Non-RT RIC as part of the SMO framework.

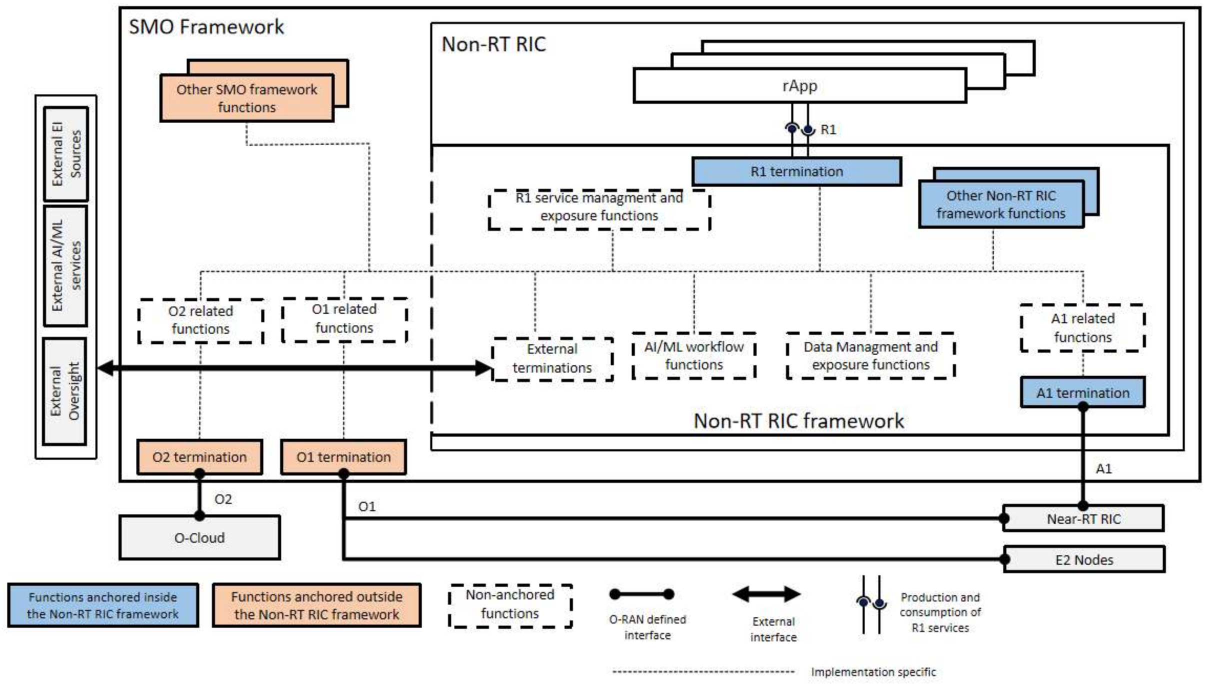


Figure 6-1: Non-RT RIC Reference Architecture

There are three categories of logical functions in Non-RT RIC framework and SMO framework:

- Functions anchored inside the Non-RT RIC framework, which are indicated in solid blue box in Figure 6-1.
- Functions anchored outside the Non-RT RIC framework, which are indicated in solid orange box in Figure 6-1.
- Non-anchored functions, which are indicated in dashed line box in Figure 6-1.

The categories are defined in Clause 3.1.

The decomposition of logical functions is not mandatory, and it is up to vendor’s implementation.

NOTE 1: There might be other SMO framework functions or other Non-RT RIC framework functions which may provide additional services exposed via the R1 interface or provide EI for A1 enrichment information services. It is FFS.

NOTE 2: Other SMO framework functions may be defined by other O-RAN WGs and the potential impacts on Non-RT RIC architecture is FFS.

NOTE 3: Whether the A1-related functions are anchored functions or not is FFS.

The R1 services are a collection of services produced by logical functions in the Non-RT RIC framework / SMO framework or by rApps.

R1 service management and exposure functions enable (not limited to): Service Registration, Service Discovery, Service Notification, Authorization, Authentication, Communication support, and optional bootstrap and heartbeat services.

The R1 termination enables the Non-RT RIC framework and rApps to exchange messages to access the R1 services via the R1 interface.

The implementation-specific interfaces, which are indicated by the dashed line in Figure 6-1, allow the logical functions in Non-RT RIC framework/SMO framework to communicate with each other by means outside the scope of the present document.

7 R1 Service Definitions

7.1 General Description

A service is a set of capabilities related to each other, offered by a service producer to service consumer(s) for consumption. The capabilities within a service typically have a common purpose and can be inter-dependent. In order to improve re-usability, the mutual coupling across any two services is usually avoided or minimized. Services are produced by service producers and consumed by service consumers through service endpoints (typically APIs). The interoperability of service producers and service consumers is ensured by support of the related service endpoints, but it does not depend on the decomposition of logical functions. Being a service producer and/or a service consumer is a role of an rApp or of a logical function in the SMO or the Non-RT RIC framework (see Clause 6).

The set of R1 services which can be accessed via the R1 interface includes (but is not limited to):

- Service management and exposure services, which include service registration and service discovery services, authentication services, authorization services and communication support services. The service management and exposure services support extensibility of the set of R1 services.
- Data management and exposure services, which deliver data created or collected by Data Producers to Data Consumers according to their needs.
- A1-related services, which provide access to functionality related to A1.
- AI/ML workflow services, which provide access to AI and ML workflow.
- O1-related services, which provide access to functionality related to O1.
- O2-related services, which provide access to functionality related to O2.

NOTE: A part of the services listed above may be used on other O-RAN-defined interfaces as well.

7.2 Service management and exposure services

The following service management and exposure services are defined:

- Bootstrap (Optional): It enables an rApp to discover the endpoints of the service management and exposure service.
- Services registration: It allows to register services which includes storing and maintaining profile about each service.
- Services deregistration: It allows to deregister services and delete the service profile.
- Service discovery: It supports service discovery for a service consumer to find services based on its selection criteria if it is provided.
- Service notification: Based on the subscription, it notifies its service consumers about the newly registered/updated/deregistered services.
- Heartbeat (Optional): It enables an rApp to maintain its R1 service registration status with the Service management and exposure services Producer.
- Authentication: It allows the authentication of a service producer and/or a service consumer.
- Authorisation: It supports to grant Service Consumers access to registered R1 services and to allow Service Producers to produce R1 services.
- Communication support: It enables R1 service consumers and producers to interact with each other.

7.3 Data management and exposure services

The following data management and exposure services are defined:

- Data registration and discovery service: Data registration allows data producers to communicate information about the data types that they are able to produce. Data discovery allows Data Consumers to discover available data types in the SMO/Non-RT RIC framework. Data Consumers need to be authorized to discover the data types.
- Data request and subscription service: Using the Subscribe data procedure, it allows Data Consumers to consume data continuously from the SMO/Non-RT RIC framework and it further allows the SMO/Non-RT RIC framework to collect data continuously from Data Producers. Using the Request data procedure, it allows Data Consumers to consume data once from the SMO/Non-RT RIC framework and it further allows the SMO/Non-RT RIC framework to collect data once from Data Producers.
- Data delivery services: These allow to deliver data for consumption from the SMO/Non-RT RIC framework to Data Consumers. They further allow to deliver data for collection from Data Producers to the SMO/Non-RT RIC framework.

- Data offer service: It allows a Data Producer to trigger the collection of a data instance by the SMO/Non-RT RIC framework from this Data Producer.
- Data processing services [Optional]: These allow service consumers to access data processing in the SMO/Non-RT RIC framework.

NOTE: Data processing services are FFS.

Data Producers (e.g., rApps) communicate information about the data types that they produce in data registration. Data Producers can communicate constraints related to the authorization of the Data Consumers to discover registered data types and access produced data. Data Producers communicate information about how the data are collected by the SMO/Non-RT RIC framework (e.g., collection periodicity, event-trigger conditions, push or pull, etc.).

Data Consumers (e.g., rApps) communicate information about the data types that they want to discover. Data Consumers subscribe/request data from the SMO/Non-RT RIC framework for consumption with specific data type, periodicity, and delivery (or reporting) method, scope, etc. The delivery (or reporting) method contains information about how the subscribed/requested data are delivered from the SMO/Non-RT RIC framework (e.g., delivery periodicity, event-trigger conditions, push or pull, etc.).

A Data Producer that intends to trigger the collection of a data instance that it produces defines a data offer by providing to the SMO/Non-RT RIC framework information about the data instance it intends to produce. The information about the data offer includes the properties of the data instance and the supported delivery services. The collection of the data instance starts when the data offer creation has been confirmed by the SMO/Non-RT RIC framework.

SMO/Non-RT RIC framework collects data produced by Data Producers. SMO/Non-RT RIC framework delivers data to Data Consumers for consumption. SMO/Non-RT RIC framework stores collected data. If the collection of the data was triggered by a data subscription or data request, data storage is optional. SMO/Non-RT RIC framework can optionally process collected and/or stored data (e.g., quantization, normalization, labelling, correlation, etc.).

7.4 A1-related Services

7.4.1 A1 policy related services

A1 policy related services provided by the SMO/Non-RT RIC framework enable:

- Discovery of supported A1 policy types.
  - rApp queries supported A1 policy types.
  - rApp subscribes/unsubscribes notification for changes of supported A1 policy types. SMO/Non-RT RIC framework notifies the subscribed events (e.g., new policy types are added into the supporting list, existing policy types are removed, etc.).
- Creating, updating, querying, and deleting A1 policies.
  - rApp creates A1 policies in the SMO/Non-RT RIC framework. SMO/Non-RT RIC framework assigns unique policy identifiers for created A1 policies.
  - rApp updates A1 policies created by itself using assigned policy identifiers.

    - rApp queries A1 policies in the SMO/Non-RT RIC framework.
        - rApp queries A1 policies applied to a given scope (e.g., a UE, a group of UEs, a slice, a QoS flow, a cell, a Near-RT RIC, a policy type, etc.). SMO/Non-RT RIC framework returns queried A1 policies along with assigned policy identifiers.
        - rApp queries A1 policies using policy identifiers, if identifiers are known to the rApp.
        - rApp can query A1 policies created by itself.
        - rApp can query A1 policies created by other rApp with SMO/Non-RT RIC framework's authorization.
    - rApp deletes A1 policies created by itself using assigned policy identifiers.
- Querying enforcement status of A1 policies.
    - rApp queries enforcement status of A1 policies using policy identifiers, if identifiers are known to the rApp.
    - rApp can query enforcement status of A1 policies created by itself.
    - rApp can query enforcement status of A1 policies created by other rApp with SMO/Non-RT RIC framework's authorization.
- Subscription to event notifications related to A1 policies.
    - rApp subscribes/unsubscribes notifications for the creation, modification, and deletion of A1 policies created by other rApps. SMO/Non-RT RIC framework notifies rApp of the subscribed events (e.g., a new A1 policy is created, an A1 policy is updated, an A1 policy is removed, etc.), if the rApp is authorized by the SMO/Non-RT RIC framework to receive notifications related to A1 policies created by other rApps.
        - Subscription request can be based on a given scope to which A1 policies are applied (e.g., a UE, a group of UEs, a slice, a QoS flow, a cell, a Near-RT RIC, a policy type, etc.).
        - Subscription request for A1 policy modification and deletion can be based on policy identifiers, if policy identifiers are known to the rApp.
    - rApp subscribes/unsubscribes notifications for changes in enforcement status of A1 policies. SMO/Non-RT RIC framework notifies rApp of the subscribed events. If the subscription request applies to the A1 policies created by other rApps, rApp needs authorization from the SMO/Non-RT RIC framework to receive notifications for enforcement status changes of A1 policies created by other rApps.
        - Subscription request can be based on a given scope to which A1 policies are applied (e.g., a UE, a group of UEs, a slice, a QoS flow, a cell, a Near-RT RIC, a policy type, etc.).
        - Subscription request can be based on policy identifiers, if identifiers are known to the rApp.

7.4.2 A1 enrichment information related services

A1 enrichment information related services produced by the SMO/Non-RT RIC framework enable:

- Registration/deregistration of EI types.
    - rApp registers EI types of which data it can produce as the source for the EI job results delivered over the A1 interface.
    - rApp deregisters EI types it has registered previously.

NOTE: EI types cannot per definition, see O-RAN.WG2.A1AP [5], be updated. If a type definition is modified, the rApp needs to register a new EI type and may deregister the previous EI type.

7.5 O1-related Services

O1-related services produced by the SMO/Non-RT RIC framework enable rApps:

- To obtain information about alarms.
- To obtain performance information related to the network.
- To obtain the current configuration of the network.
- To provision changes of the configuration of the network.
- To obtain additional information related to the network.

7.6 O2-related services

O2-related services produced by the SMO/Non-RT RIC framework enable rApps:

- To obtain information related to O-Cloud infrastructure management services such as,
    - Infrastructure Inventory.
    - Infrastructure Monitoring.
    - Infrastructure Provisioning.
    - Infrastructure Lifecycle Management.
    - Infrastructure Software Management.
- To obtain information related to O-Cloud deployment management services such as,
    - Deployment Inventory.
    - Deployment Monitoring.
    - Deployment Lifecycle Management.
- To provision changes of the configuration of the O-Cloud.
- To obtain additional information related to the O-Cloud.

NOTE: Details about O-Cloud deployment management services, provision changes of the configuration, additional information related are FFS.

8 Non-RT RIC Function Definitions

8.1 General Descriptions

In the Non-RT RIC architecture in Clause 6, R1 services are provided by a collection of logical functions in Non-RT RIC framework/SMO framework or rApps. These logical functions, as defined in Clause 6, include (but are not limited to):

- R1 service management and exposure functions: These functions produce service management and exposure services.
- R1 termination: It enables the Non-RT RIC framework and rApps to exchange messages to access the R1 services via the R1 interface.
- Data management and exposure functions: These functions produce data management and exposure services.
- AI/ML workflow functions: These functions produce AI/ML workflow services.

NOTE: AI/ML workflow functions will be defined in further releases of the present document.

- A1-related functions: These functions produce A1 related services.
- O1-related functions: These functions produce O1 related services.
- O2-related functions: These functions produce O2 related services.
- A1 termination: It enables the Non-RT RIC framework and the Near-RT RIC to exchange messages over the A1 interface.
- O1 termination: It enables the SMO framework to exchange messages with the Near-RT RIC/E2 nodes over the O1 interface.
- O2 termination: It enables the SMO framework to exchange messages with the O-Cloud over the O2 interface.
- External terminations: These functions enable the SMO/Non-RT RIC framework to exchange messages with external entities over interfaces outside the scope of O-RAN.

8.2 R1 service management and exposure functions

8.2.1 Service registration and discovery functions

The service registration and discovery functions are logical functions that produce the service registration, service deregistration, service discovery, service notification, communication support services and optional bootstrap and heartbeat services specified in Clause 7.2.

Examples of service producers include: service management and exposure services producer, A1-related services producer, etc. Examples of service consumers are rApps. It is noted that rApps can also be service producers and logical functions can also be service consumers.

A service producer or an entity acting on its behalf can send a service registration request to the service registration and discovery functions to register services. A service consumer can either send a service discovery request to the service registration and discovery functions to discover the services and the endpoints on which they can be consumed, or send a service request directly if it receives a service notification with the related information on the service and on the endpoints on which it can be consumed.

8.2.2 Authentication and authorisation functions

The authentication and authorisation functions are logical functions that produce the authentication and authorisation services specified in Clause 7.2.

8.3 Data management and exposure functions

The data management and exposure functions are logical functions that produce the data registration and discovery service, the data request and subscription service, the data offer service and optional data processing services specified in Clause 7.3. They further initiate data collection by consuming the data request and subscription service. Additionally, they produce and consume the data delivery services in order to obtain collected data from data producers and to allow data consumers to obtain data from the data management and exposure functions for consumption.

The data management and exposure functions track data types that are registered. They match data discovery requests against such data types. If the data type is registered, the data management and exposure functions identify a valid Data Producer, which can be a previously registered Data Producer rApp or other Data Producers in the Non-RT RIC framework /SMO (e.g., O1-related functions). The data management and exposure functions optionally provide data processing functionality on collected data (e.g., quantization, normalization, correlation, labelling, etc.).

When Data Consumers use the Subscribe data procedure to initiate continuous data consumption, the data management and exposure functions check the availability of the subscribed data after receiving a data subscription request from a Data Consumer. If the subscribed data are already being collected, the data management and exposure functions deliver the data to the Data Consumer when they become available. If the data are not already being collected and a valid Data Producer is identified, the data management and exposure functions initiate data collection to obtain data from the identified Data Producer. When the collected data become available, the data management and exposure functions deliver the data to the Data Consumer.

When Data Consumers use the Request data procedure to initiate one-time data consumption, the data management and exposure functions check the availability of the required data after receiving a one-time data request from a Data Consumer. If the requested data are available, the data management and exposure functions deliver the data to the Data Consumer. If the data are not available and a valid Data Producer is identified, the data management and exposure functions initiate data collection to obtain data from the identified Data Producer. When the collected data become available, the data management and exposure functions deliver the data to the Data Consumer.

When a Data Producer uses the create Data offer procedure to trigger data collection, the data management and exposure functions set up an endpoint for receiving data availability notifications or push data messages (depending on the delivery method chosen) and confirm the creation of the data offer to the Data Producer. Data collection starts once the data offer creation is confirmed, and proceeds until the data offer is terminated by the Data Producer or by the data management and exposure functions. The data management and exposure functions store the collected data for later consumption.

The use of a data repository for storing data during data collection is optional if data collection was triggered by a data request or data subscription from a Data Consumer. It is required if data collection was triggered by a data offer from a Data Producer. Data can be optionally pre-processed before being stored, and can be optionally post-processed after being read.

NOTE: Instead of storing data in a data repository, data management and exposure functions can broker data from Data Producers to Data Consumers.

8.4 A1-related functions

8.4.1 A1 Policy functions

The logical functions that produce the A1 policy related services which are exposed to rApps via the R1 interface are called the “A1 policy functions". They realize the following functionalities:

- Producing A1 policy related services which are exposed to rApps via the R1 interface.
    - The corresponding service definition can be found in Clause 7.4.1.
- Performing A1 policy conflict mitigation.
    - If more than one policy generation rApps set contradicting A1 policy statements to the same or overlapping policy scope identifiers, A1 policy function decides to reject the creation/modification request of one or more A1 policies or to temporarily suspend the enforcement of one or more A1 policies.
- Interfacing with Near-RT RICs through A1 termination
    - A1 policy function determines the Near-RT RIC for generated A1 policies.
    - A1 policy function sends A1 policy creation, update, and deletion requests to the Near-RT RIC.
    - A1 policy function queries supported A1 policy types of connected Near-RT RICs.
    - A1 policy function queries enforcement status of A1 policies.
    - A1 policy function receives A1 policy feedback from the Near-RT RIC.

8.4.2 A1 enrichment information functions

“A1 enrichment information functions” produce the A1 enrichment information related services which are exposed to rApps via the R1 interface. The corresponding services definition can be found in Clause 7.4.2. “A1 enrichment information functions” also produce the A1-EI service which is exposed to Near-RT RICs via the A1 interface. The corresponding service definition can be found in the A1 interface specification, see references in Clause 9.1. “A1 enrichment information functions” collect data that are to be delivered as EI job results to Near-RT RICs via the data management and exposure functions.

NOTE: The data delivered as EI job results can be produced by rApps or by external enrichment information sources.

“A1 enrichment information functions” realize the following functionalities:

- Handling of EI types registered by rApps via the R1 interface.

- Handling of EI jobs created by the Near-RT RICs via the A1 interface.
- Collecting data that are to be delivered as EI job results to the Near-RT RICs via the A1 interface.
- Interfacing with Near-RT RICs through A1 termination.

8.5 O1-related functions

The logical functions that produce the O1-related services which are exposed to rApps via the R1 interface are called the “O1-related functions”. They realize the following functionalities:

- Producing O1-related services which are exposed to rApps via the R1 interface.
    - The corresponding service definition can be found in Clause 7.5.
- Performing CM conflict mitigation.
- Interfacing with Near-RT RICs and E2 nodes through O1 termination.
    - O1-related functions receive fault notifications and obtain alarm list from Near-RT RICs and E2 nodes.
    - O1-related functions provision configuration changes to Near-RT RICs and E2 nodes.
    - O1-related functions collect performance data from Near-RT RICs and E2 nodes.

8.6 O2-related functions

The O2 interface is an open logical interface within O-RAN architecture. The logical functions that produce the O2-related services which are exposed to rApps via the R1 interface are called the “O2-related functions”. They realize the following functionalities:

- Producing O2-related services which are exposed to rApps via the R1 interface.
    - The corresponding service definition can be found in Clause 7.6.
- Interfacing with O-Cloud (O2ims & O2dms) through O2 termination.
    - O2-related functions receive O-Cloud infrastructure FCAPS (PM, CM, FM) data from O-Cloud IMS.
    - O2-related functions receive O-Cloud deployment FCAPS (PM, CM, FM) data from O-Cloud DMS.
    - O2-related functions provision configuration changes to O-Cloud.
- Performing conflict mitigation for configuration Changes.

9 External Interfaces

9.1 A1 Interface

O-RAN.WG2.A1GAP [4] specifies A1 interface general aspects and principles.

O-RAN.WG2.A1AP [5] specifies A1 interface application protocols.

O-RAN.WG2.A1TP [6] specifies A1 interface transport protocols.

O-RAN.WG2.A1TD [7] specifies data model and data types used in the A1 interface.

9.2 Other Interfaces

NOTE: Other external interfaces include external EI interface, external AI/ML interface, interface for external oversight, which are out of scope of the present specification.

10 Procedures

10.1 Data management and exposure procedures

This clause demonstrates example procedures related to data management and exposure (DME) services produced and consumed by SMO/Non-RT RIC framework. A Data Consumer rApp request/subscribe data from SMO/Non-RT RIC framework, and the requested/subscribed data are collected from another Data Producer rApp or other O-RAN entities.

NOTE: It is not required to use the same type of DME services procedure in data consumption and data collection.

Data consumption denotes the procedures of Data Consumer rApp requesting/subscribing data from SMO/Non-RT RIC framework. Data collection denotes the procedures of SMO/Non-RT RIC framework requesting/subscribing data from Data Producer rApps or other O-RAN entities.

10.1.1 Subscribe data procedures in data consumption and collection

Figure 10.1.1-1 illustrates an example workflow of using DME services to subscribe data and deliver data over the R1 interface. In this example, it is assumed that

- Two Data Consumer rApps subscribe the same data instance (i.e., same data type and scope) from the DME functions.
- The requested data are produced by a Data Producer rApp.
- The DME functions subscribe data instance from the Data Producer rApp.
- Data are delivered using pull or push data delivery. In case of pull delivery, the pull delivery details can be contained in subscribe data response or data availability notification. The Data Consumer makes a pull request based on the received pull delivery details.

When the first Data Consumer rApp initiates the Subscribe data procedure to obtain data, the DME functions first identify the Data Producer for the requested data instance. In this example, one Data Producer rApp which is already registered as being capable of producing the subscribed data are identified as the Data Producer. The DME functions figure out there is no data subscription for the requested data instance yet, and the DME functions subscribe data from Data Producer rApp. In case of pull delivery, the Data Producer rApp notifies the data availability to the DME functions and provides pull data details in the notification. The DME functions then pull data from the Data Producer rApp. In case of push delivery, the Data Producer rApp pushes the data to the DME functions. The DME functions may optionally process the data or store the data in the data repository for future usage. In case of pull delivery, the DME functions further notify the data availability to the first Data Consumer rApp. The rApp then pulls the data from the DME functions. In case of push delivery, the DME functions push the data to the Data Consumer rApp.

When the second Data Consumer rApp sends a data subscription request to the DME functions, the DME functions find out that the data subscription for the requested data instance is already established. Therefore, the DME functions do not set up additional data subscription with the Data Producer rApp. Once new data are available, the following steps are executed: In case of pull delivery, the Data Producer notifies the data availability to the DME functions, and these pull the data from the Data Producer. In case of push delivery, the Data Producer pushes the data to the DME functions. The DME functions may optionally process the data and store the data in the data repository for future usage. In case of pull delivery, the DME functions further notify the data availability to both Data Consumer rApps which then pull the data from the DME functions. In case of push delivery, the DME functions push the data to both Data Consumer rApps.

The DME functions can use different procedures to deliver data to Data Consumer rApps even if both Data Consumer rApps subscribe the same data instance. For example, the DME functions can notify data availability to the one Data Consumer rApp allowing the rApp to pull data, meanwhile the DME functions push data to another Data Consumer.

NOTE 1: Push content can be (1) notifications that carry a data payload or (2) streams of data.

NOTE 2: Notifications and push data delivery may require the set-up of a dedicated communication channel that allows the content to be pushed. Such dedicated communication channels may map to, e.g., Kafka topic, WebSocket connection, gRPC connection, etc. Alternatively, individual HTTP requests may be used as push messages.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
autonumber

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "Data Consumer rApp #1" as app1
    participant "Data Consumer rApp #2" as app2
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "Data Management and Exposure functions" as dme
    database "Data repository" as db
  endbox

  box #ivory
    participant "Data Producer rApp" as src
  endbox
endbox

group Data Consumer rApp #1 subscribes data
app1 -> dme : <<R1>> Subscribe data request
app1 <- dme : <<R1>> Subscribe data response
end

dme --> dme : Find out that data instance\nis not being collected;\nidentify the Data
Producer

group DME functions subscribes data from Data Producer rApp
src <- dme : <<R1>> Subscribe data request
src -> dme : <<R1>> Subscribe data response
end
```

```

loop data available
alt pull delivery
     src -> dme : <<R1>> Notify data availability (pull delivery details)
     src <- dme : <<R1>> Pull data request
     src -> dme : <<R1>> Pull data response (with data payload)
else push delivery
     src -> dme : <<R1>> Push data message (with data payload)
end
opt
dme --> dme : Processing data
end
opt
dme <--> db: Storing data
end

alt pull delivery
     app1 <- dme : <<R1>> Notify data availability (pull delivery details)
     app1 -> dme : <<R1>> Pull data request
     app1 <- dme : <<R1>> Pull data response (with data payload)
else push delivery
     app1 <- dme : <<R1>> Push data message (with data payload)
end
end

group Data Consumer rApp #2 subscribes data
app2 -> dme : <<R1>> Subscribe data request
app2 <- dme : <<R1>> Subscribe data response
end

dme --> dme : Find out that data instance\nis already being collected

loop data available
alt pull delivery
     src -> dme : <<R1>> Notify data availability (pull delivery details)
     src <- dme : <<R1>> Pull data request
     src -> dme : <<R1>> Pull data response (with data payload)
else push delivery
     src -> dme : <<R1>> Push data message (with data payload)
end
opt
dme --> dme : Processing data
end
opt
dme <--> db: Storing data
end

alt pull delivery
     app1 <- dme : <<R1>> Notify delivery details (pull delivery details)
     app1 -> dme : <<R1>> Pull data request
     app1 <- dme : <<R1>> Pull data response (with data payload)
     app2 <- dme : <<R1>> Notify data availability (pull delivery details)
     app2 -> dme : <<R1>> Pull data request
     app2 <- dme : <<R1>> Pull data response (with data payload)
else push delivery
     app1 <- dme : <<R1>> Push data message (with data payload)
     app2 <- dme : <<R1>> Push data message (with data payload)
end
end
@enduml
```

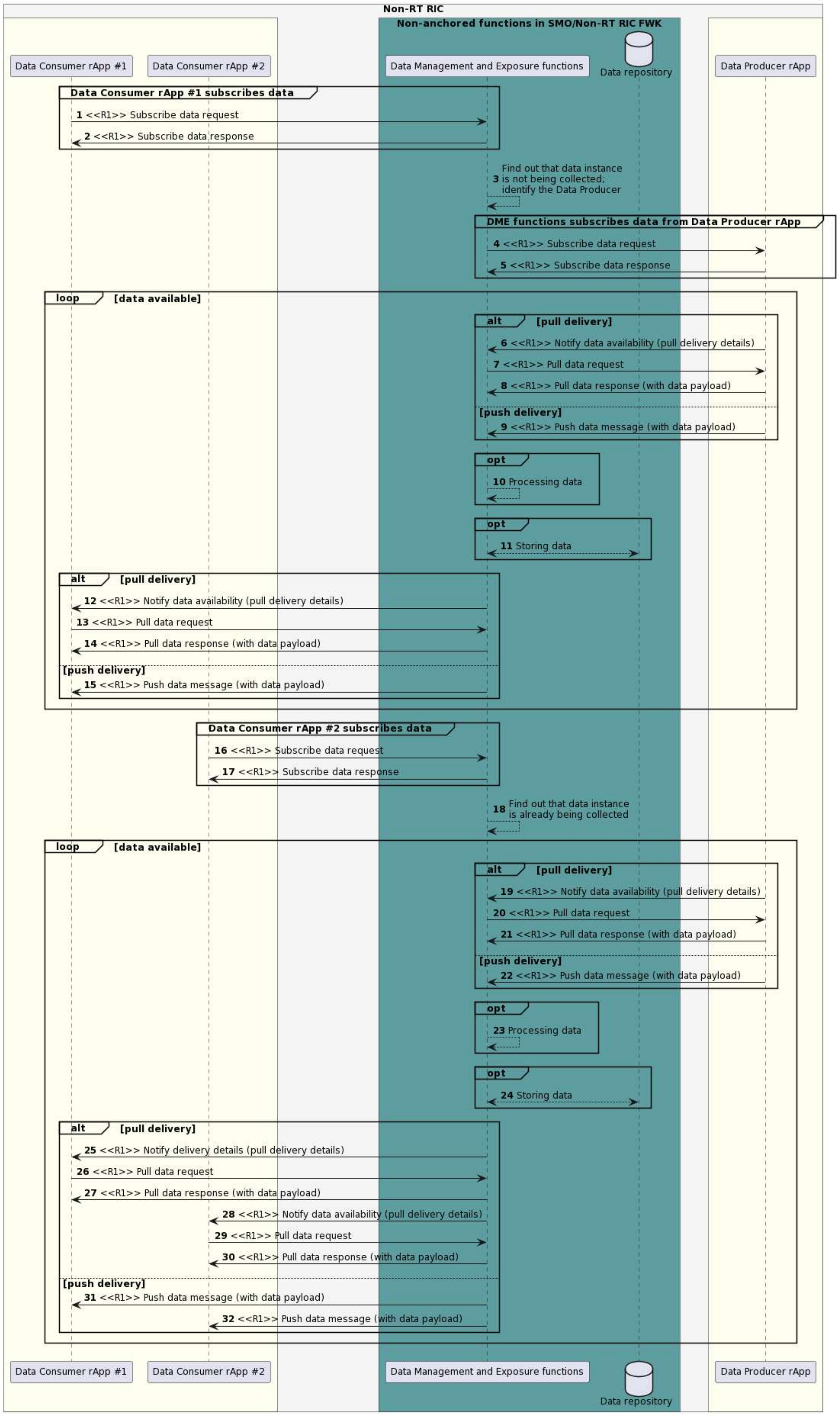
1 
Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
Data Consumer rApp #1
Data Consumer rApp #2
Data Management and Exposure functions
Data repository
Data Producer rApp
Data Consumer rApp #1 subscribes data
1 <<R1>> Subscribe data request
2 <<R1>> Subscribe data response
3 Find out that data instance is not being collected; identify the Data Producer
DME functions subscribes data from Data Producer rApp
4 <<R1>> Subscribe data request
5 <<R1>> Subscribe data response
loop [data available]
alt [pull delivery]
6 <<R1>> Notify data availability (pull delivery details)
7 <<R1>> Pull data request
8 <<R1>> Pull data response (with data payload)
[push delivery]
9 <<R1>> Push data message (with data payload)
opt
10 Processing data
opt
11 Storing data
alt [pull delivery]
12 <<R1>> Notify data availability (pull delivery details)
13 <<R1>> Pull data request
14 <<R1>> Pull data response (with data payload)
[push delivery]
15 <<R1>> Push data message (with data payload)
Data Consumer rApp #2 subscribes data
16 <<R1>> Subscribe data request
17 <<R1>> Subscribe data response
18 Find out that data instance is already being collected
loop [data available]
alt [pull delivery]
19 <<R1>> Notify data availability (pull delivery details)
20 <<R1>> Pull data request
21 <<R1>> Pull data response (with data payload)
[push delivery]
22 <<R1>> Push data message (with data payload)
opt
23 Processing data
opt
24 Storing data
alt [pull delivery]
25 <<R1>> Notify delivery details (pull delivery details)
26 <<R1>> Pull data request
27 <<R1>> Pull data response (with data payload)
28 <<R1>> Notify data availability (pull delivery details)
29 <<R1>> Pull data request
30 <<R1>> Pull data response (with data payload)
[push delivery]
31 <<R1>> Push data message (with data payload)
32 <<R1>> Push data message (with data payload)
Data Consumer rApp #1
Data Consumer rApp #2
Data Management and Exposure functions
Data repository
Data Producer rApp

Figure 10.1.1-1: Example workflow using Subscribe data and Notify data availability procedures

Figure 10.1.1-2 illustrates an example workflow of using Unsubscribe data procedures to terminate data subscription in data consumption data and data collection. In the example, it is assumed that the two Data Consumer rApps in the figure are the only two Data Consumers which have subscribed the data instance produced by the Data Producer rApp.

When the second Data Consumer rApp initiates the Unsubscribe data procedure, the DME functions find out that the produced data are also subscribed by the first Data Consumer rApp. Therefore, the DME functions can't unsubscribe from the Data Producer rApp.

When the first Data Consumer rApp initiates the Unsubscribe data procedure, the DME functions figure out that the data subscription is no longer needed. Hence, in this example, the DME functions identify the Data Producer rApp and send an unsubscription request to cancel the data subscription.

NOTE: The DME functions are not required to unsubscribe data from Data Producers even if there are no Data Consumers requesting or subscribing these data. The DME functions may collect data from Data Producers for its own purposes.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
autonumber

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "Data Consumer rApp #1" as app1
    participant "Data Consumer rApp #2" as app2
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "Data Management and Exposure functions" as dme
    database "Data repository" as db
  endbox

  box #ivory
    participant "Data Producer rApp" as src
  endbox
endbox

group Data Consumer rApp #2 unsubscribes data
app2 -> dme : <<R1>> Unsubscribe data  request
app2 <- dme : <<R1>> Unsubscribe data response
end

dme --> dme : Find out that data are subscribed \n by other Data Consumers

group Data Consumer rApp #1 unsubscribes data
app1 -> dme : <<R1>> Unsubscribe data request
app1 <- dme : <<R1>> Unsubscribe data response
end

dme --> dme : Find out that data subscription \n is no longer required;\nidentify the
Data Producer

opt DME unsubscribes data from Data Producer rApp
```

```
src <- dme : <<R1>> Unsubscribe data request
src -> dme : <<R1>> Unsubscribe data response
end

@enduml
```

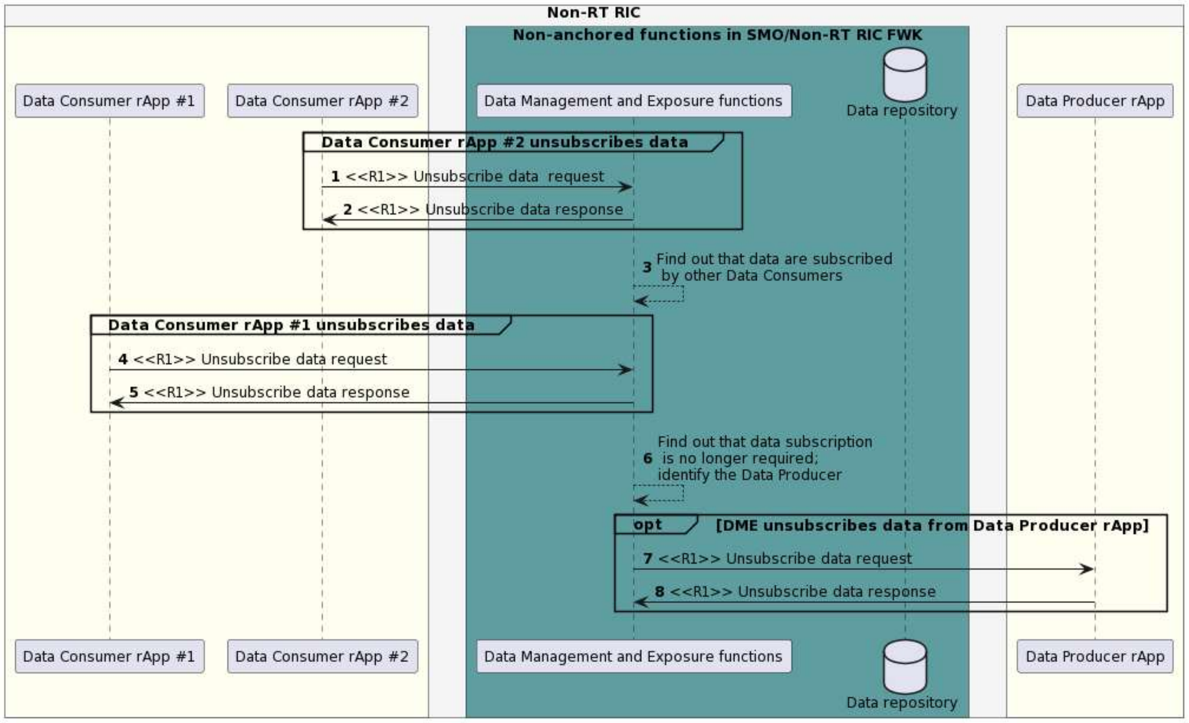

Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
Data Consumer rApp #1
Data Consumer rApp #2
Data Management and Exposure functions
Data repository
Data Producer rApp
Data Consumer rApp #2 unsubscribes data
1 <<R1>> Unsubscribe data request
2 <<R1>> Unsubscribe data response
3 Find out that data are subscribed by other Data Consumers
Data Consumer rApp #1 unsubscribes data
4 <<R1>> Unsubscribe data request
5 <<R1>> Unsubscribe data response
6 Find out that data subscription is no longer required; identify the Data Producer
opt [DME unsubscribes data from Data Producer rApp]
7 <<R1>> Unsubscribe data request
8 <<R1>> Unsubscribe data response

Figure 10.1.1-2: Example workflow using Unsubscribe data procedure

10.1.2 Request data procedure in data consumption and data collection

Figure 10.1.2-1 illustrates an example workflow of using DME services to request and deliver data over the R1 interface. In this example, it is assumed that

- Two Data Consumer rApps request the same data instance (i.e., same data type and scope) using Request data procedure.
- The requested data are produced by a Data Producer rApp.
- The DME functions request data from the Data Producer rApp using Request data procedure.
- Data are delivered using pull delivery. The pull delivery details are contained in the request data response. The Data Consumer makes a pull request based on the received pull delivery details.

The pull delivery details provide Data Consumer information of how to formulate a pull data request for the delivery of requested data (e.g., a URI from which to fetch the data).

When the first Data Consumer rApp initiates the Request data procedure to obtain data, the DME functions first check whether the requested data are available in the data repository. In this example, it figures out that data are not available. Then the DME functions identify the Data Producer rApp as the Data Producer, e.g., based on data type registration records. The DME functions send data request to the Data Producer rApp, and the data are delivered from the Data Producer rApp using pull delivery. The DME functions then send data request response to the first Data Consumer rApp carrying pull delivery details. The first Data

Consumer rApp pulls data from the DME functions. The DME functions may optionally process data and store the data in the data repository for future usage.

When the second Data Consumer rApp sends data request to the DME functions, the DME functions notice that the requested data are already available in the data repository. Then the DME functions retrieve requested data from data repository and responds the second Data Consumer rApp with pull delivery details. The data are pulled by the second Data Consumer rApp.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
autonumber

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "Data Consumer rApp #1" as app1
    participant "Data Consumer rApp #2" as app2
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "Data Management and Exposure functions" as dme
    database "Data repository" as db
  endbox

  box #ivory
    participant "Data Producer rApp" as src
  endbox
endbox
group Data Consumer rApp #1 requests data
app1 -> dme : <<R1>> Request data request
dme --> dme : Find out that requested data \n are not available in the data repository
app1 <- dme : <<R1>> Request data response (pull delivery details)

group Obtain data not available in the repository
src <- dme : <<R1>> Request data request
src -> dme : <<R1>> Request data response (pull delivery details)
src <- dme : <<R1>> Pull data request
src -> dme : <<R1>> Pull data response (with data payload)
opt
dme --> dme : Processing data
end
opt
dme <--> db: Storing data
end
end

app1 -> dme : <<R1>> Pull data request
app1 <- dme : <<R1>> Pull data response (with data payload)
end

group Data Consumer rApp #2 requests data
app2 -> dme : <<R1>> Request data request
dme --> dme : Find out that requested data \n are available in the data repository
app2 <- dme : <<R1>> Request data response (pull delivery details)
group Obtain data that are available in the repository
dme <--> db: Retrieving data
end
app2 -> dme : <<R1>> Pull data request
app2 <- dme : <<R1>> Pull data response (with data payload)
end
```

```
@enduml
```

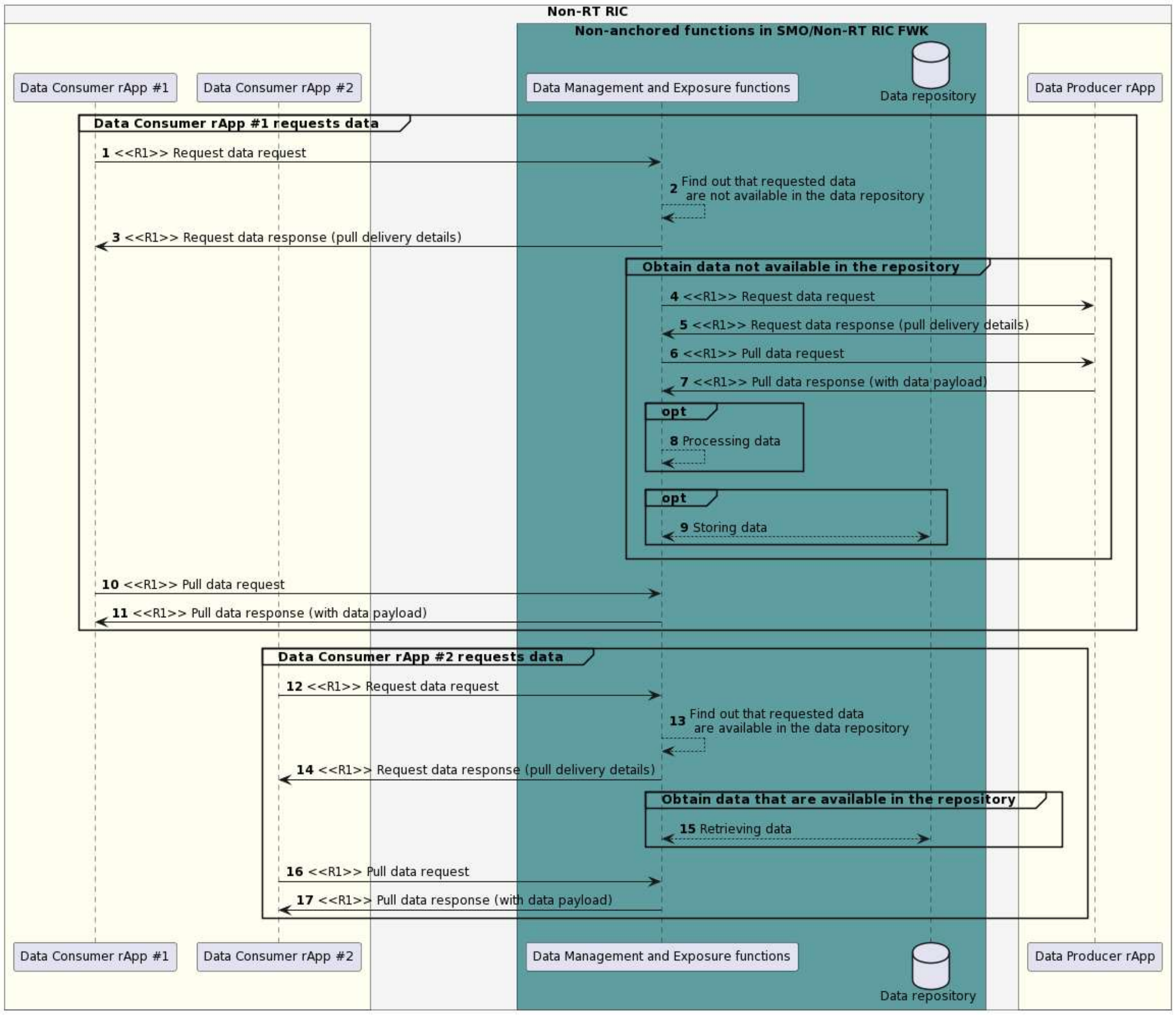

Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
Data Consumer rApp #1
Data Consumer rApp #2
Data Management and Exposure functions
Data repository
Data Producer rApp
Data Consumer rApp #1 requests data
1 <<R1>> Request data request
2 Find out that requested data are not available in the data repository
3 <<R1>> Request data response (pull delivery details)
Obtain data not available in the repository
4 <<R1>> Request data request
5 <<R1>> Request data response (pull delivery details)
6 <<R1>> Pull data request
7 <<R1>> Pull data response (with data payload)
opt
8 Processing data
opt
9 Storing data
10 <<R1>> Pull data request
11 <<R1>> Pull data response (with data payload)
Data Consumer rApp #2 requests data
12 <<R1>> Request data request
13 Find out that requested data are available in the data repository
14 <<R1>> Request data response (pull delivery details)
Obtain data that are available in the repository
15 Retrieving data
16 <<R1>> Pull data request
17 <<R1>> Pull data response (with data payload)
Data Consumer rApp #1
Data Consumer rApp #2
Data Management and Exposure functions
Data repository
Data Producer rApp

Figure 10.1.2-1: Example workflow using Request data procedure

10.1.3 O1 related data collection procedures based on data subscription

Figure 10.1.3-1 illustrates an example workflow of how data management and exposure (DME) functions in the SMO/Non-RT RIC framework collect O1 related data based on the data subscription from a Data Consumer rApp.

In this example, when the Data Consumer rApp initiates the Subscribe data procedure to obtain O1 related data, the DME functions acknowledge the data subscription, and check whether subscribed data are being collected already or not. If the subscribed data are not being collected, the DME functions initiates Data collection procedure to collect from Radio nodes/Near-RT RIC via the O1 related functions. This procedure has been specified in clause 2.2.2. “O-RAN Measurement Data Collection” in O-RAN.WG10.OAM-Architecture [8]. When the DME functions receive the collected data from the O1 related functions, it notifies the Data Consumer rApp, and the subscribed data are delivered using pull or push data services.

NOTE: Data delivery procedures are defined in clause 3.2.5 of O-RAN.WG2.R1GAP [9].

```
@startuml
```

```
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
skinparam sequenceReferenceHeaderBackgroundColor Transparent
autonumber

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "Data Consumer rApp" as dc
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "Data Management and Exposure functions" as dme
    participant "O1 related functions" as OAM
  endbox
endbox
participant “Radio node/Near-RT RIC” as dp
autonumber

dc -> dme : <<R1>> Subscribe data request
dme ->dc : <<R1>> Subscribe data response

dme --> dme : Check if the subscribed data are being collected already

group if subscribed data are not being collected yet
ref over dp, dme
As specified in Section 2.2.2. “O-RAN Measurement Data Collection” in
O-RAN.WG10.OAM-Architecture
end
end

loop data available
OAM --> dme : Implementation-specific data delivery
alt pull delivery
dme -> dc: <<R1>> Notify data availability (pull delivery details)
dc -> dme: <<R1>> Pull data request
dme -> dc: <<R1>> Pull data response (data payload)
else push delivery
dme -> dc : <<R1>>Push data message (data payload)
end
end

@enduml
```

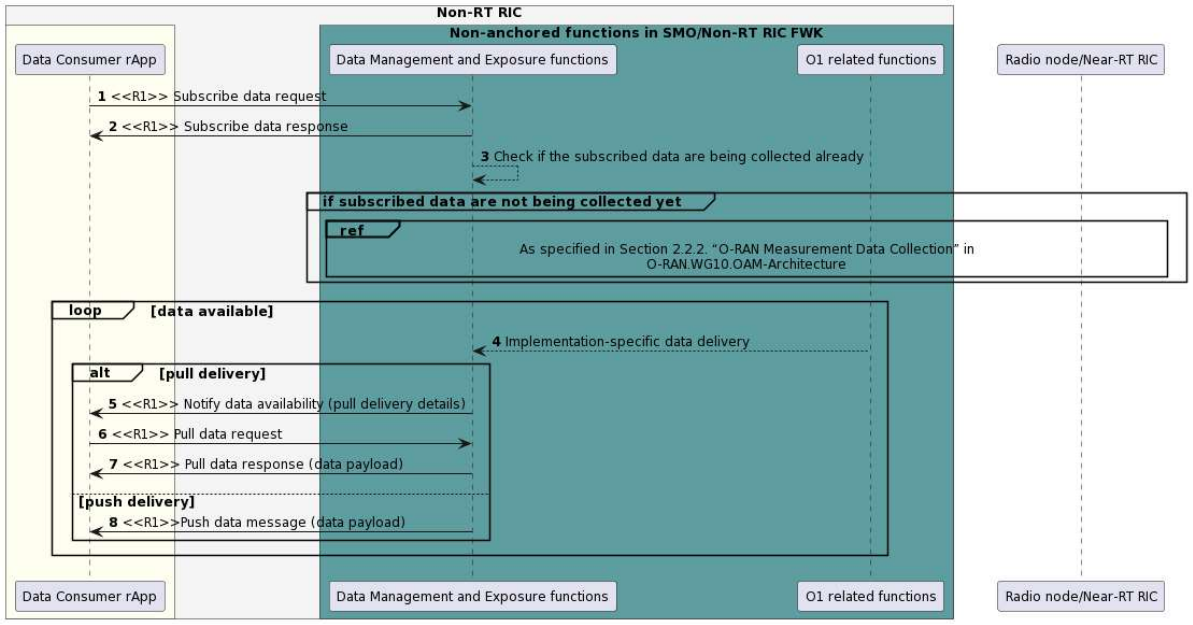

Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
Data Consumer rApp
Data Management and Exposure functions
O1 related functions
Radio node/Near-RT RIC
1 <<R1>> Subscribe data request
2 <<R1>> Subscribe data response
3 Check if the subscribed data are being collected already
if subscribed data are not being collected yet
ref
As specified in Section 2.2.2. "O-RAN Measurement Data Collection" in O-RAN.WG10.OAM-Architecture
loop [data available]
4 Implementation-specific data delivery
alt [pull delivery]
5 <<R1>> Notify data availability (pull delivery details)
6 <<R1>> Pull data request
7 <<R1>> Pull data response (data payload)
[push delivery]
8 <<R1>>Push data message (data payload)

Figure 10.1.3-1: Example workflow for O1 related data collection procedure via DME

10.1.4 O2 related data collection procedures based on data subscription

Figure 10.1.4-1 illustrates an example workflow of how data management and exposure (DME) functions in the SMO/Non-RT RIC framework collect O2 related data based on the data subscription from a Data Consumer rApp.

In this example, when the Data Consumer rApp initiates the Subscribe data procedure to obtain O2 related data, the DME functions acknowledge the data subscription, and check whether subscribed data are being collected already or not. If the subscribed data are not being collected, the DME functions initiate Data collection procedure to collect from DMS via the O2 related functions. This procedure has been specified in clause 2.2.2. "O-RAN Measurement Data Collection" in O-RAN.WG10.OAM-Architecture [8]. When the DME functions receive the collected data from the O2 related functions, it notifies the Data Consumer rApp, and the subscribed data are delivered using pull or push data services.

NOTE: Data delivery procedures are defined in clause 3.2.5 of O-RAN.WG2.R1GAP [9].

```
@startuml
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
skinparam sequenceReferenceHeaderBackgroundColor Transparent

autonumber

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "Data Consumer rApp" as dc
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
```

```
    participant "Data Management and Exposure functions" as dme
    participant "O2 related functions" as OAM
  endbox
endbox
participant “DMS” as dp
autonumber

dc -> dme : <<R1>> Subscribe data request
dme ->dc : <<R1>> Subscribe data response

dme --> dme : Check if the subscribed data are being collected already

group if subscribed data are not being collected yet
ref over dp, dme
As specified in clause 2.2.2. “O-RAN Measurement Data Collection” in
O-RAN.WG10.OAM-Architecture
End
End

loop data available
OAM --> dme : Implementation-specific data delivery
alt pull delivery
dme -> dc: <<R1>> Notify data availability (pull delivery details)
dc -> dme: <<R1>> Pull data request
dme -> dc: <<R1>> Pull data response (data payload)
else push delivery
dme -> dc : <<R1>> Push data message (data payload)
end
end

@enduml
```

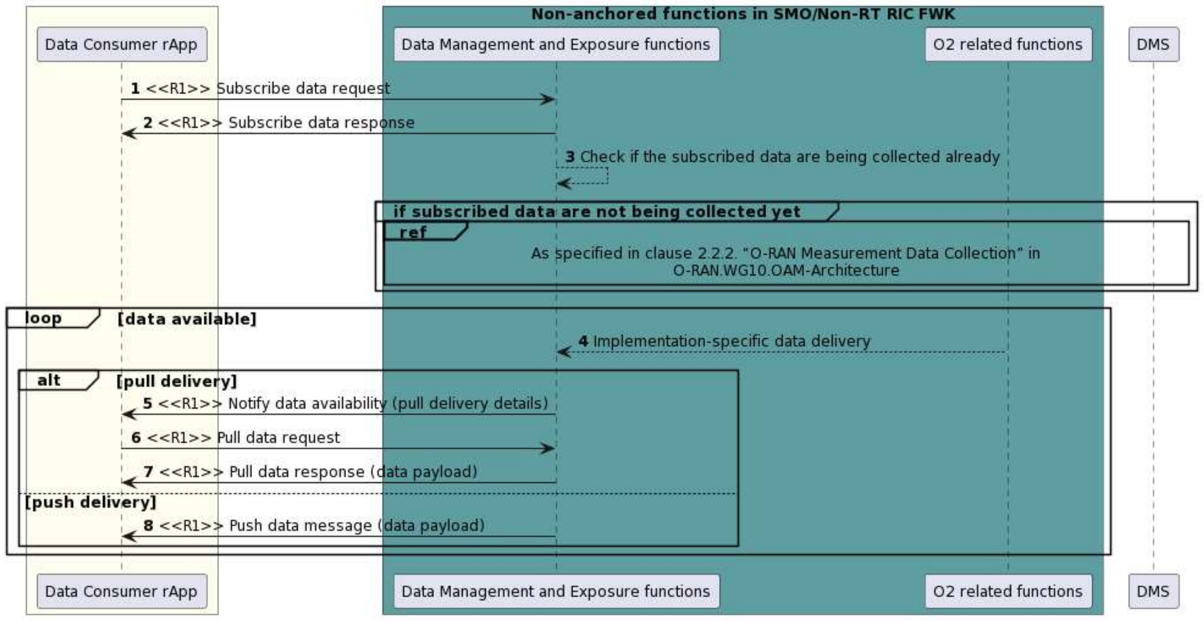

Non-anchored functions in SMO/Non-RT RIC FWK
Data Consumer rApp
Data Management and Exposure functions
O2 related functions
DMS
1 <<R1>> Subscribe data request
2 <<R1>> Subscribe data response
3 Check if the subscribed data are being collected already
if subscribed data are not being collected yet
ref
As specified in clause 2.2.2. "O-RAN Measurement Data Collection" in O-RAN.WG10.OAM-Architecture
loop [data available]
4 Implementation-specific data delivery
alt [pull delivery]
5 <<R1>> Notify data availability (pull delivery details)
6 <<R1>> Pull data request
7 <<R1>> Pull data response (data payload)
[push delivery]
8 <<R1>> Push data message (data payload)

Figure 10.1.4-1: Example workflow for O2 related data collection procedure via DME

10.1.5 Data offer procedures

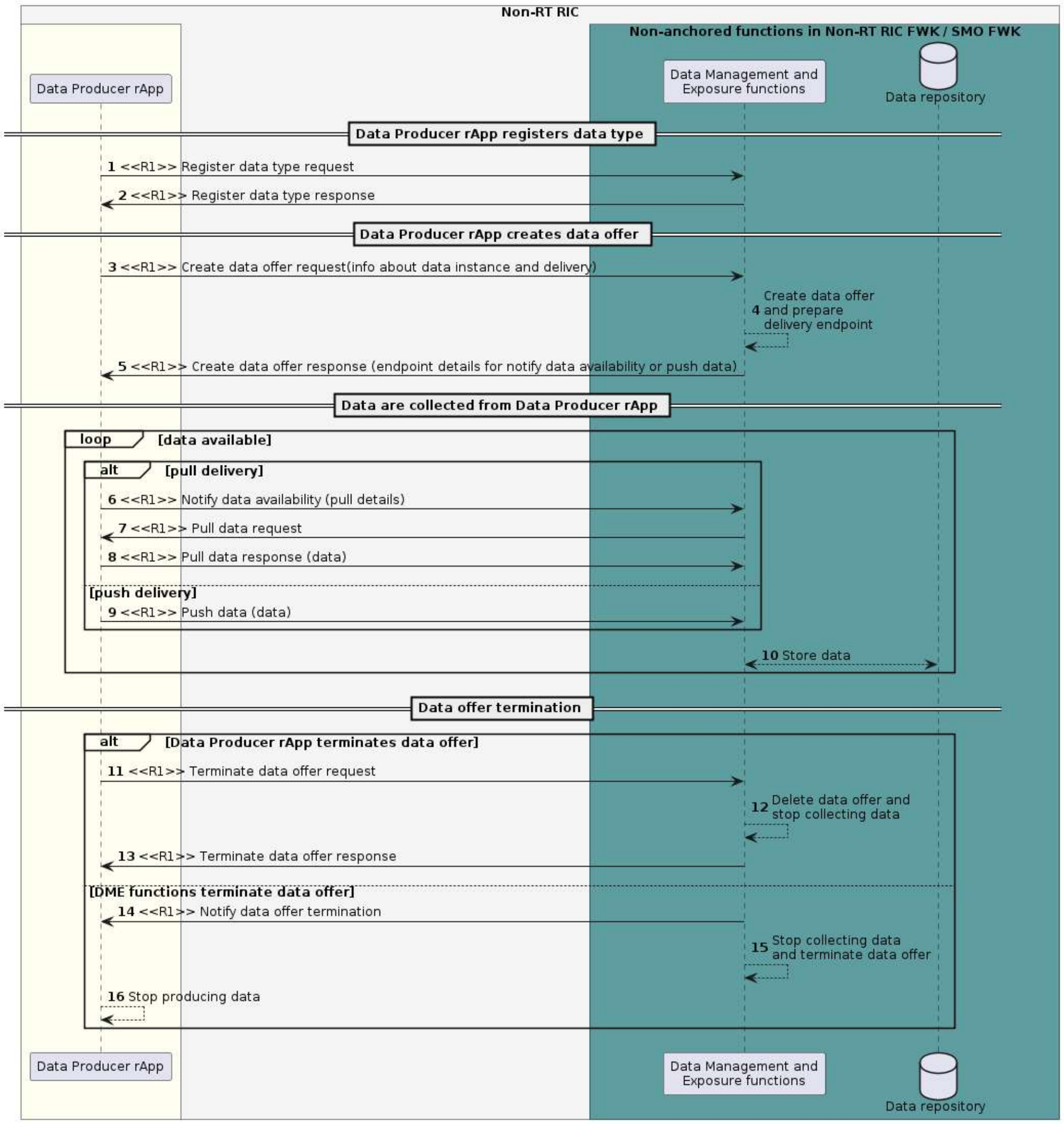
Figure 10.1.5-1 illustrates an example workflow of using the DME services by a Data Producer to trigger the DME functions to collect a data instance produced by that Data Producer and to store it for later consumption.

The Data Producer rApp first needs to register a data type for the data instance it intends to produce. The rApp then initiates the Create data offer procedure to trigger the collection of a data instance of that data type which it intends to produce. The parameters of the data instance and the supported delivery services are indicated in the request. After receiving the request, the DME functions first check whether the Data Producer has registered the data type for which it intends to produce an instance. Upon success, the DME functions set up an endpoint for the reception of data availability notifications or push data messages, depending on the delivery service chosen, and communicate this information to the Data Producer. Data collection can start immediately after the data offer creation has been acknowledged by the DME functions.

When data are available, the Data Producer sends a data availability notification to the DME functions and these pull the data, or the Data Producer pushes the data to the DME functions. Then, the DME functions store these data for subsequent consumption.

When the Data Producer intends to stop data production, it terminates the data offer by using the Terminate data offer procedure. Data collection ends upon successful execution of that procedure. When the data management and exposure functions intend to stop receiving produced data, they send a data offer termination notification to the Data Producer. Subsequently, possibly after a delay to allow the Data Producer to react to the notification, the data management and exposure functions stop data collection and terminate the data offer. Upon receiving this notification, the Data Producer stops data production.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
skinparam lifelineStrategy solid
autonumber

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "Data Producer rApp" as rapp
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "Data Management and\nExposure functions" as dme
    database "Data repository" as db
  endbox
endbox

group Data Producer rApp registers data type
     rapp -> dme: <<R1>> Register data type request
     rapp <- dme: <<R1>> Register data type response
end

group Data Producer rApp creates data offer
     rapp -> dme: <<R1>> Create data offer request(info about data instance and
delivery)
     dme --> dme: Create data offer\nand prepare\ndelivery endpoint
     dme -> rapp: <<R1>> Create data offer response (endpoint details for notify data
availability\nor push data)
end

group Data are collected from Data Producer rApp
     loop data available
          alt pull delivery
               rapp -> dme: <<R1>> Notify data availability (pull details)
               dme -> rapp: <<R1>> Pull data request
               rapp -> dme: <<R1>> Pull data response (data)
          else push delivery
               rapp -> dme: <<R1>> Push data (data)
          end
          dme <--> db: Store data
     end
end

group Data offer termination
     alt Data Producer rApp terminates data offer
          rapp -> dme: <<R1>> Terminate data offer request
          dme --> dme: Delete data offer and\nstop collecting data
          dme -> rapp: <<R1>> Terminate data offer response
     else DME functions terminate data offer
          dme -> rapp: <<R1>> Notify data offer termination
          dme --> dme: Stop collecting data\nand terminate data offer
          rapp --> rapp: Stop producing data
     end
end

@enduml
```



Non-RT RIC
Non-anchored functions in Non-RT RIC FWK / SMO FWK
Data Producer rApp
Data Management and Exposure functions
Data repository
Data Producer rApp registers data type
1 <<R1>> Register data type request
2 <<R1>> Register data type response
Data Producer rApp creates data offer
3 <<R1>> Create data offer request(info about data instance and delivery)
4 Create data offer and prepare delivery endpoint
5 <<R1>> Create data offer response (endpoint details for notify data availability or push data)
Data are collected from Data Producer rApp
loop [data available]
alt [pull delivery]
6 <<R1>> Notify data availability (pull details)
7 <<R1>> Pull data request
8 <<R1>> Pull data response (data)
[push delivery]
9 <<R1>> Push data (data)
10 Store data
Data offer termination
alt [Data Producer rApp terminates data offer]
11 <<R1>> Terminate data offer request
12 Delete data offer and stop collecting data
13 <<R1>> Terminate data offer response
[DME functions terminate data offer]
14 <<R1>> Notify data offer termination
15 Stop collecting data and terminate data offer
16 Stop producing data

Figure 10.1.5-1: Example workflow using Data offer Procedures to trigger the collection and storage of data for later consumption

10.2 A1-related procedures

10.2.1 EI results collection from rApps

Figure 10.2.1-1 illustrates an example workflow of collecting data from rApps (over the R1 interface) that is delivered as EI job results to the Near-RT RICs (over the A1 interface). EI job results refer to the data being transferred from the Non-RT RIC to the Near-RT RICs, while EI source data refers to the data produced by the rApp that acts as the source for the data provided as EI job results. Correspondingly, EI type refers to the data type that is made available to the Near-RT RICs while the corresponding R1 data type registered by the rApp is referred to as EI source data type. Moreover, EI job refers to a request for EI by the Near-RT RIC while EI source data subscription refers to the corresponding subscription to the rApp that acts the source for the data.

NOTE 1: EI source data type refers to any data type that is registered with the DME and the corresponding data are used as the source of EI job results.

The rApp which generates the enrichment information first registers the EI source data type in the data management and exposure (DME) functions. After successful R1 data type registration, the rApp registers the EI type for which it can produce enrichment information to the A1 enrichment information functions. In the registration request, the rApp should include the EI source data type registered in the DME functions. The EI type is to be exposed via the A1 interface to the Near-RT RIC. The EI source data type allows the A1 enrichment information functions to subscribe EI source data from DME functions.

The A1 enrichment information functions query DME functions about the EI source data type. If the query is successful, the EI type registered by the rApp is made available over the A1 interface. The Near-RT RIC queries available EI type identifiers in the Non-RT RIC. The Near-RT RIC queries EI type and creates an EI job via the A1 interface. After receiving the EI job creation request from the Near-RT RIC, the A1 enrichment information functions subscribe EI source data from DME functions. The DME functions further subscribe EI source data from the rApp over the R1 interface.

Each time new EI source data are available, the rApp delivers EI source data to the DME functions using pull or push delivery. The DME functions deliver EI source data to the A1 enrichment information functions. The A1 enrichment formation functions establish the connection with Near-RT RIC and push the EI job results via the A1 interface.

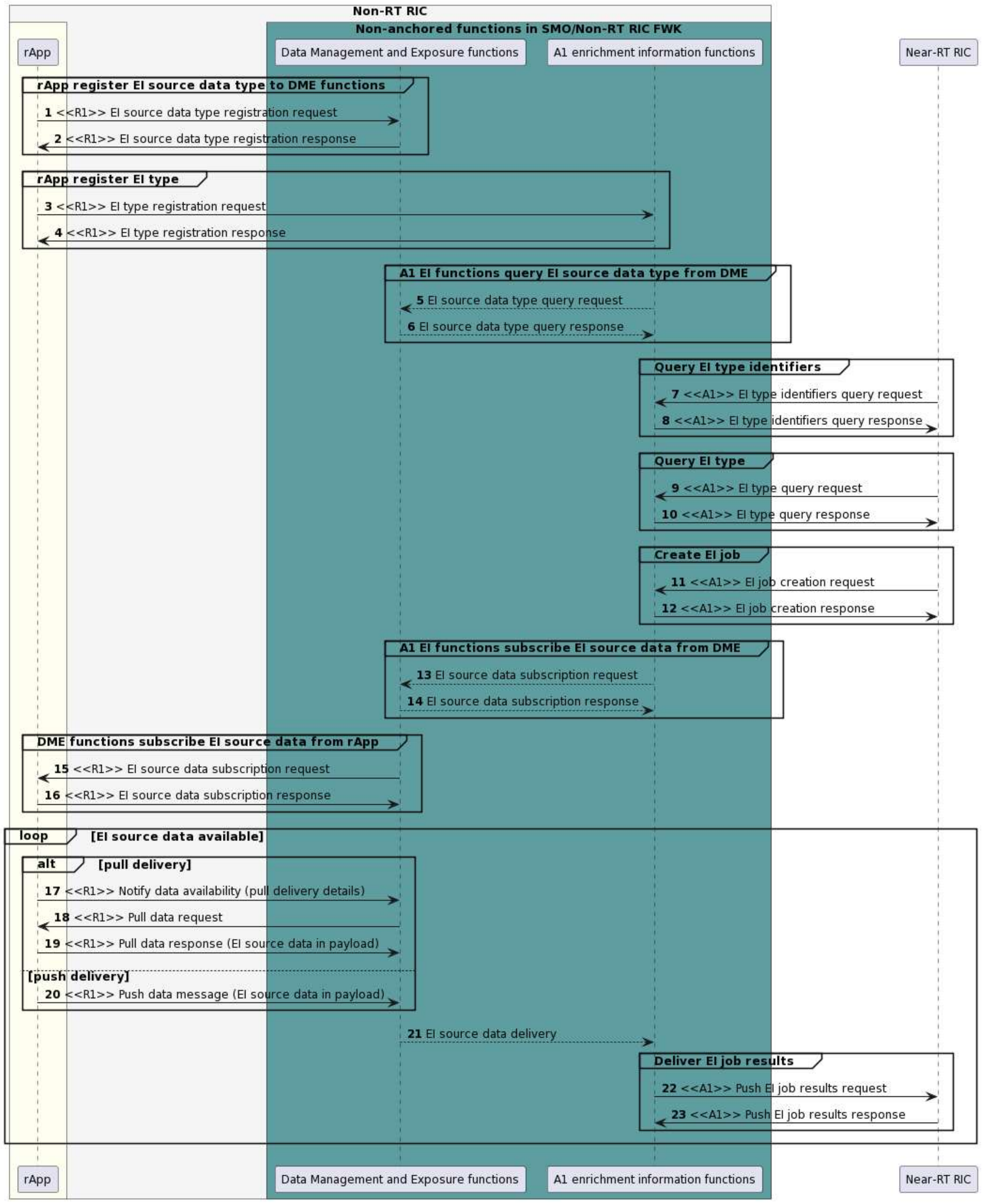
NOTE 2: Non-RT RIC framework internal interactions are not specified.

NOTE 3: As an alternative to using the Data subscription procedure, the Data request procedure can be used to collect EI source data from the rApp.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
autonumber

box "Non-RT RIC" #whitesmoke
```

```
  box #ivory
    participant "rApp" as app
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "Data Management and Exposure functions" as dme
    participant "A1 enrichment information functions" as eif
  endbox
endbox

Participant "Near-RT RIC" as ran

group rApp register EI source data type to DME functions
app -> dme : <<R1>> EI source data type registration request
app <- dme : <<R1>> EI source data type registration response
end

group rApp register EI type
app -> eif : <<R1>> EI type registration request
app <- eif : <<R1>> EI type registration response
end

group A1 EI functions query EI source data type from DME
eif --> dme : EI source data type query request
eif <-- dme : EI source data type query response
end

group Query EI type identifiers
ran -> eif : <<A1>> EI type identifiers query request
ran <- eif : <<A1>> EI type identifiers query response
end

group Query EI type
ran -> eif : <<A1>> EI type query request
ran <- eif : <<A1>> EI type query response
end

group Create EI job
ran -> eif : <<A1>> EI job creation request
ran <- eif : <<A1>> EI job creation response
end

group A1 EI functions subscribe EI source data from DME
eif --> dme : EI source data subscription request
eif <-- dme : EI source data subscription response
end

group DME functions subscribe EI source data from rApp
dme -> app : <<R1>> EI source data subscription request
dme <- app : <<R1>> EI source data subscription response
end

loop EI source data available
alt pull delivery
app -> dme: <<R1>> Notify data availability (pull delivery details)
app <- dme: <<R1>> Pull data request
app -> dme: <<R1>> Pull data response (EI source data in payload)
else push delivery
app -> dme : <<R1>> Push data message (EI source data in payload)
end


dme --> eif : EI source data delivery
group Deliver EI job results
eif -> ran : <<A1>> Push EI job results request
eif <- ran : <<A1>> Push EI job results response
end
```

```
end
@enduml
```



Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
rApp
Data Management and Exposure functions
A1 enrichment information functions
Near-RT RIC
rApp register EI source data type to DME functions
1 <<R1>> EI source data type registration request
2 <<R1>> EI source data type registration response
rApp register EI type
3 <<R1>> EI type registration request
4 <<R1>> EI type registration response
A1 EI functions query EI source data type from DME
5 EI source data type query request
6 EI source data type query response
Query EI type identifiers
7 <<A1>> EI type identifiers query request
8 <<A1>> EI type identifiers query response
Query EI type
9 <<A1>> EI type query request
10 <<A1>> EI type query response
Create EI job
11 <<A1>> EI job creation request
12 <<A1>> EI job creation response
A1 EI functions subscribe EI source data from DME
13 EI source data subscription request
14 EI source data subscription response
DME functions subscribe EI source data from rApp
15 <<R1>> EI source data subscription request
16 <<R1>> EI source data subscription response
loop [EI source data available]
alt [pull delivery]
17 <<R1>> Notify data availability (pull delivery details)
18 <<R1>> Pull data request
19 <<R1>> Pull data response (EI source data in payload)
[push delivery]
20 <<R1>> Push data message (EI source data in payload)
21 EI source data delivery
Deliver EI job results
22 <<A1>> Push EI job results request
23 <<A1>> Push EI job results response

Figure 10.2.1-1: Example workflow for EI results collection procedures

NOTE: A procedure for EI source data collection from external sources is FFS.

10.3 O1-related procedures

10.3.1 Alarm query procedure

Figure 10.3.1-1 illustrates an example workflow of querying alarm information. Alarm information is typically received by the SMO via O1. It is consolidated by the Non-RT RIC / SMO framework and can be queried by rApps via the R1 interface.

NOTE: M-Plane alarms and slice alarms are FFS.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
skinparam sequenceReferenceHeaderBackgroundColor Transparent
skinparam lifelineStrategy solid

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "rApp" as rapp
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "O1-related functions" as o1rf
  endbox
endbox
participant "Radio nodes / Near-RT RIC " as me

group SMO/Non-RT RIC FW consolidates alarm information
      alt Using alarm notifications
            ref over o1rf, me
                <b>1</b> As specified in Section 2.2.1.3 "Fault Notification - Procedures"
                in O-RAN.WG10.O1-Interface
            end
      else Using alarm list
            ref over o1rf, me
              <b>2</b> As specified in Section 2.2.2.3 "Fault Supervision Control -
              Procedures"
              in O-RAN.WG10.O1-Interface
            end
      end

      autonumber 3
      o1rf --> o1rf: Consolidate alarm information
end

rapp -> o1rf: <<R1>> Query alarm information request(rApp identifier, filter criteria)
rapp <- o1rf: <<R1>> Query alarm information response(alarm information)

@enduml
```

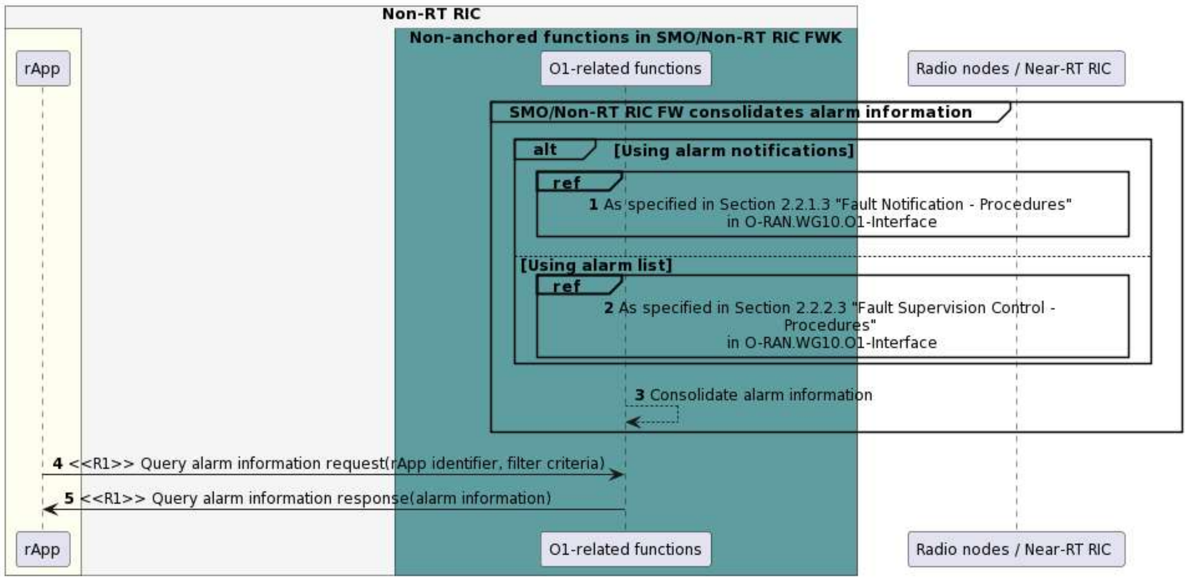

Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
rApp
O1-related functions
Radio nodes / Near-RT RIC
SMO/Non-RT RIC FW consolidates alarm information
alt [Using alarm notifications]
ref
1 As specified in Section 2.2.1.3 "Fault Notification - Procedures" in O-RAN.WG10.O1-Interface
[Using alarm list]
ref
2 As specified in Section 2.2.2.3 "Fault Supervision Control - Procedures" in O-RAN.WG10.O1-Interface
3 Consolidate alarm information
4 <<R1>> Query alarm information request(rApp identifier, filter criteria)
5 <<R1>> Query alarm information response(alarm information)
rApp
O1-related functions
Radio nodes / Near-RT RIC

Figure 10.3.1-1: Example workflow of alarm query

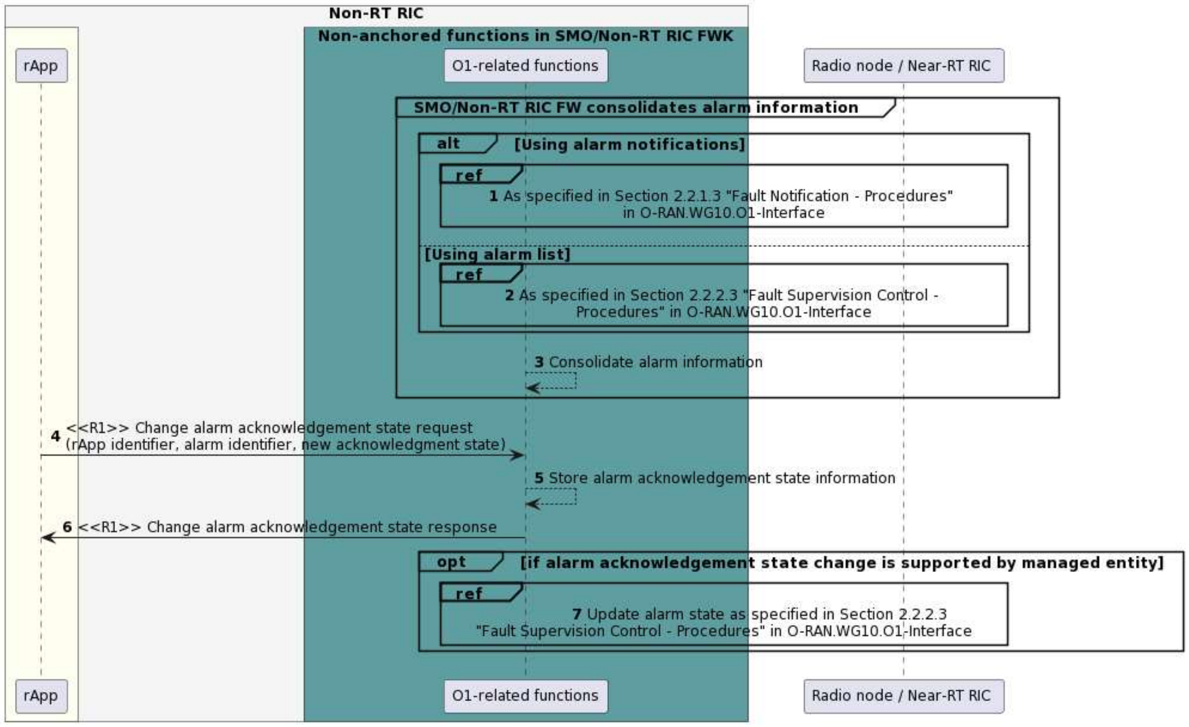
10.3.2 Change alarm acknowledgement state procedure

Figure 10.3.2-1 illustrates an example workflow of changing the acknowledgement state of an alarm. Alarm information is typically received by the SMO via O1. It is consolidated by the Non-RT RIC / SMO framework. The rApps can change the acknowledgement state of the alarms that are known by the SMO/Non-RT RIC framework. Alarm acknowledgment state information is usually managed by the SMO/Non-RT RIC framework and is only passed to the radio nodes / Near-RT RIC if alarm acknowledgement / unacknowledgement is supported by these entities.

NOTE: M-Plane alarms and slice alarms are FFS.

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
skinparam sequenceReferenceHeaderBackgroundColor Transparent
skinparam lifelineStrategy solid

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "rApp" as rapp
  endbox

  box "Non-anchored functions in SMO/Non-RT RIC FWK" #cadetBlue
    participant "O1-related functions" as o1rf
  endbox
endbox
participant "Radio node / Near-RT RIC " as me

group SMO/Non-RT RIC FW consolidates alarm information
     alt Using alarm notifications
          ref over o1rf, me
              <b>1</b> As specified in Section 2.2.1.3 "Fault Notification - Procedures"
              in O-RAN.WG10.O1-Interface
          end
```

```
      else Using alarm list
           ref over o1rf, me
             <b>2</b> As specified in Section 2.2.2.3 "Fault Supervision Control -
             Procedures" in O-RAN.WG10.O1-Interface
           end
      end

      autonumber 3
      o1rf --> o1rf: Consolidate alarm information
end

rapp -> o1rf: <<R1>> Change alarm acknowledgement state request\n(rApp identifier, alarm
identifier, new acknowledgment state)
o1rf --> o1rf: Store alarm acknowledgement state information
rapp <- o1rf: <<R1>> Change alarm acknowledgement state response

opt if alarm acknowledgement state change is supported by managed entity
     ref over o1rf, me
          <b>7</b> Update alarm state as specified in Section 2.2.2.3
          "Fault Supervision Control - Procedures" in O-RAN.WG10.O1-Interface
     end
end

@enduml
```



Non-RT RIC
Non-anchored functions in SMO/Non-RT RIC FWK
rApp
O1-related functions
Radio node / Near-RT RIC
SMO/Non-RT RIC FW consolidates alarm information
alt [Using alarm notifications]
ref
1 As specified in Section 2.2.1.3 "Fault Notification - Procedures" in O-RAN.WG10.O1-Interface
[Using alarm list]
ref
2 As specified in Section 2.2.2.3 "Fault Supervision Control - Procedures" in O-RAN.WG10.O1-Interface
3 Consolidate alarm information
4 <<R1>> Change alarm acknowledgement state request (rApp identifier, alarm identifier, new acknowledgment state)
5 Store alarm acknowledgement state information
6 <<R1>> Change alarm acknowledgement state response
opt [if alarm acknowledgement state change is supported by managed entity]
ref
7 Update alarm state as specified in Section 2.2.2.3 "Fault Supervision Control - Procedures" in O-RAN.WG10.O1-Interface
rApp
O1-related functions
Radio node / Near-RT RIC

Figure 10.3.2-1: Example workflow of alarm acknowledgement state change

Revision History

Date	Revision	Description
2022.07.30	02.01	Editorial correction based on the review comments
2022.07.21	02.00.03	Implemented CRs: NOK-2022-06-21-WG2-CR-0046-NonRT_RIC_Arch-Alarm Query_v03.docx INT-2022.06.16-WG2-CR-00038-rApp creates policy without Near-RT RIC identifier-v03.docx INT-2022.07.06-WG2-CR-00040-remove void clauses-v01.docx INT-2022.07.06-WG2-CR-00041-align plantuml figures-v01.docx NOK-2022-06-21-WG2-CR-0045-NonRT_RIC_Arch-Alarm Ack_v03.docx NOK.AO-2022-07-12-WG2-CR-0052-NonRT_RIC_ARCH-fix containment relationships_v03.docx Remove “skin rose” in PlantUML markup text Adjusted according to updated template “O-RAN.WGn.descriptor.0-v01.00-TS-template.docx”. Implemented changes to the cover page and foot note. Removed Annex ZZZ.
2022.06.30	02.00.02	A new baseline according to O-RAN.TSC.Drafting-Rules.0-v01.00 based on CR CMCC.AO-2022.06.21-WG2-CR-0008-Non-RT RIC ARCH TS drafting rules update-v02.zip
2022.05.26	02.00.01	Implemented CRs: NOK-2022-03-29-WG2-CR-0030-Non_RT_RIC_Arch-DME alignment review comments NOK02 NOK03_v03.docx NOK-2022-04-12-WG2-CR-0036-NoRT_RIC_Arch-Resolving remaining review comments on services + functions_v03.docx NOK-2022-04-13-WG2-CR-0037-NoRT_RIC_Arch-Resolving remaining review comments on procedures_v02.docx NOK-2022-04-13-WG2-CR-0038-NoRT_RIC_Arch-Resolving remaining review comments on EI_v01.docx NOK.AO-2022-04-13-WG2-CR-0039-NonRT_RIC_Arch-DME Data Offer_v05.docx
2022.03.30	02.00	v02.00 Publication

History

Date	Revision	Description
2022.07.30	02.01	Published as Final version 02.01
2022.03.30	02.00	v02.00 Publication
2021.07.16	01.00	v01.00 Publication