O-RAN.WG2.R1UCR-R003-v03.00
Technical Specification

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

R1 interface: Use Cases and Requirements

O-RAN ALLIANCE e.V., Buschkauler Weg 27, 53347 Alfter, Germany
Register of Associations, Bonn VR 11238, VAT ID DE321720189

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 release date and an increase in version number as follows:

Release xx.yy.zz

where:

xx the first two-digit value is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc. (the initial approved document shall have xx=01).

yy the second two-digit value is incremented when editorial only changes have been incorporated in the document.

zz the third two-digit value is included only in working versions of the document indicating incremental changes during the editing process; externally published documents never have this third two-digit value included.

The present document specifies the R1 Use cases and Requirements. It is part of a TS-family covering the WG2: R1 Interface Specifications.

2 References

2.1 Normative references

References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific. For a specific reference, only the cited version applies.For a non-specific reference, the latest version of that referenced document (including any amendments) applies.

[1] O-RAN WG2: "R1 General Aspects and Principles" ("R1GAP")

[2] O-RAN WG2: " Use Cases and Requirements"("UCR")

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 Definitions of terms, symbols and abbreviations

3.1 Terms

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

Non-RT RIC: O-RAN non-real-time RAN Intelligent Controller: a logical function in the SMO framework 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. The Non-RT RIC is comprised of the Non-RT RIC framework and Non-RT RIC applications (rApps).

Non-RT RIC framework: Functionality internal to the SMO framework that logically terminates the A1 interface and provides the R1services to rApps through the R1 interface.

rApp: Non-RT RIC application: an application designed to consume and/or produce R1 Services.

NOTE: rApps can leverage the functionality provided by the SMO/Non-RT RIC framework to deliver value added services related to intelligent RAN optimization and operation.

3.2 Symbols

Void

3.3 Abbreviations

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

API	Application programming Interface
AuthN	Authentication
AuthZ	Authorization
DME	Data Management and Exposure
FWK	Framework
FM	Fault management
ID	Identifier
Non-RT RIC	Non-real-time RAN Intelligent Controller
PM	Performance management
SME	Service Management and Exposure
SMO	Service Management and Orchestration

4 General

The R1 interface is defined between the rApps and the Non-RT RIC framework, as defined in R1GAP [1].

In the use case sequence flows, solid lines are used for the message interactions which are defined in O-RAN specifications. Dashed lines are used to represent the messages interactions that are not specified in O-RAN specifications or can be implementation specific interactions.

5 Requirements

5.1 General

The general R1 interface requirements are specified in UCR [2].

5.2 SME services functional requirements.

This clause specifies the SME services functional requirements.

The functional requirements are specified in table 5.2-1

Table 5.2-1: SME services functional requirements

REQ	Description	Note
REQ-R1-SME-rAppreg-FUN1	The SME services shall support registration of an rApp.	
REQ-R1-SME-rAppreg-FUN2	The SME services shall support authentication of an rApp.	
REQ-R1-SME-Servicemag-FUN1	The SME services shall support registration of a service.	
REQ-R1-SME-Servicemag-FUN2	The SME services shall support update of a service registration.	
REQ-R1-SME-Servicemag-FUN3	The SME services shall support deregistration of a registered service.	
REQ-R1-SME-Service discovery - FUN1	The SME services shall support discovery of services.	
REQ-R1-SME-Servicesub-FUN1	The SME services shall support subscription to notifications of service availability changes.	
REQ-R1-SME-Servicesub-FUN2	The SME services shall support sending notifications for the service availability changes.	
REQ-R1-SME-Servicesub-FUN3	The SME services shall support unsubscription to notifications of service availability changes.	

5.3 DME services functional requirements

This cause specifies the DME services functional requirements.

The functional requirements are specified in table 5.3-1.

Table 5.3-2: DME services functional requirements

REQ	Description	Note
REQ-R1-DME-Regdatatype-FUN1	The DME services shall support registration and deregistration of a data types.	
REQ-R1-DME-Discoverydatatype-FUN1	The DME services shall support querying available data types.	
REQ-R1-DME-Subscribedatatypechanges-FUN1	The DME services shall support subscription to notifications regarding changes in the availability of data types.	
REQ-R1-DME-Notifydatatypechanges-FUN1	The DME services shall support sending notifications about changes in the availability of data types.	
REQ-R1-DME-Unsubscribedatatypechanges-FUN1	The DME services shall support to unsubscribe from notifications regarding changes in the availability of data types.	
REQ-R1-DME-Dataoffer -FUN1	The DME services shall support the creation of data offer.	
REQ-R1-DME-Dataoffer -FUN2	The DME services shall support the termination of data offers.	
REQ-R1-DME-Data subscription-FUN1	The DME services shall support the creation of data subscriptions.	
REQ-R1-DME-Data subscription-FUN2	The DME services shall support the termination of data subscriptions.	
REQ-R1-DME-Data request-FUN1	The DME services shall support the creation of data requests.	
REQ-R1-DME-Data request-FUN2	The DME services shall support the cancellation of data requests.	
REQ-R1-DME-Data delivery-FUN1	The DME services shall support the delivery of subscribed or requested data.	

5.4 RAN OAM-related services functional requirements

This clause specifies the FM and PM services functional requirements.

The functional requirements are specified in table 5.4.1-1.

Table 5.4.1-3: FM and PM services functional requirements

REQ	Description	Note
REQ-R1-OAM-FMservice-FUN1	The FM service shall support querying alarm information related to the O-RAN managed entities.	
REQ-R1-OAM-FMservice-FUN2	The FM service shall support changing the acknowledgement state of alarms related to the O-RAN managed entities.	

REQ-R1-OAM-PMservice-FUN1	The PM service shall support querying performance information related to the O-RAN managed entities.	

NOTE: The querying of performance information, and how this relates to the Data management and exposure services, is FFS.

6 Use cases for Service Management and Exposure

6.1 SME use case 1: Registration and Bootstrap

6.1.1 Overview

This use case provides the description and requirements for registering an rApp with the SMO/Non-RT RIC framework.

6.2.1 Background and goal of the use case

The rApp registration procedure registers the rApp to the SMO/Non-RT RIC framework. It may use the bootstrap service to discover the rApp registration API. To register an rApp, an rApp may pass relevant info that may include the rApp name, vendor, software version, and other information needed by the SMO/Non-RT RIC framework.

6.3.1 Entities/resources involved in the use case

1) SME functions in the role of SME services Producer
    a. support functionality that allows the rApp to retrieve the bootstrap information that enables it to discover the SME services,
    b. support functionality to authenticate the rApp,
    c. assigns the rApp with an ID.
2) rApp
    a. retrieves the bootstrap information that enables it to discover the SME services,
    b. initiates the procedure of rApp registration.

6.4.1 Solutions

6.4.1.1 rApp registration and bootstrap

Table 6.4.1.1-4: rApp registration and bootstrap use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	1) Find the SME service endpoints (Optional), 2) Register the rApp to the SME services Producer in SMO/Non-RT RIC framework.	
Actors and Roles	- rApp in the role of Service Producer and/or Service Consumer - SME functions in the role of SME services Producer.	
Assumptions	n/a	
Preconditions	The rApp is instantiated.	
Begins when	The rApp is ready for registration.	
Step 1 (O)	The rApp will initiate a bootstrap request to find the SME endpoints.	
Step 2 (O)	The SME functions respond with the details of the service endpoints.	
Step 3 (M)	The rApp sends a registration request to the SME functions in the SMO/Non-RT RIC framework passing relevant information needed.	
Step 4 (M)	The SME functions process the rApp registration request that may include - performing authentication, - if rApp is authenticated, rAppId will be assigned to rApp.	
Step 5 (M)	Response to the registration request along with the rAppId.	
Ends when	The rApp is allocated with an rAppId.	
Exceptions	n/a	
Post Conditions	The rApp is registered with the SMO/Non-RT RIC framework and an rAppId is allocated to the rApp.	
Traceability	REQ-R1-SME-rAppreg-FUN1, REQ-R1-SME-rAppreg-FUN2.	

```
@startuml
'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
participant  "SME functions"  as SEF
endbox
==bootstrap ==
opt
rApp -> SEF :<<R1>>Bootstrap request
activate SEF
SEF -> rApp :<<R1>>Bootstrap response
Deactivate SEF
end
==rApp registration ==
rApp -> SEF:<<R1>>Registration request
activate SEF
```



```
SEF--> SEF : AuthN
note right
registration processing
assign rAppId
end note
SEF -> rApp :<<R1>>Registration response (rAppId)
Deactivate SEF
@enduml
```

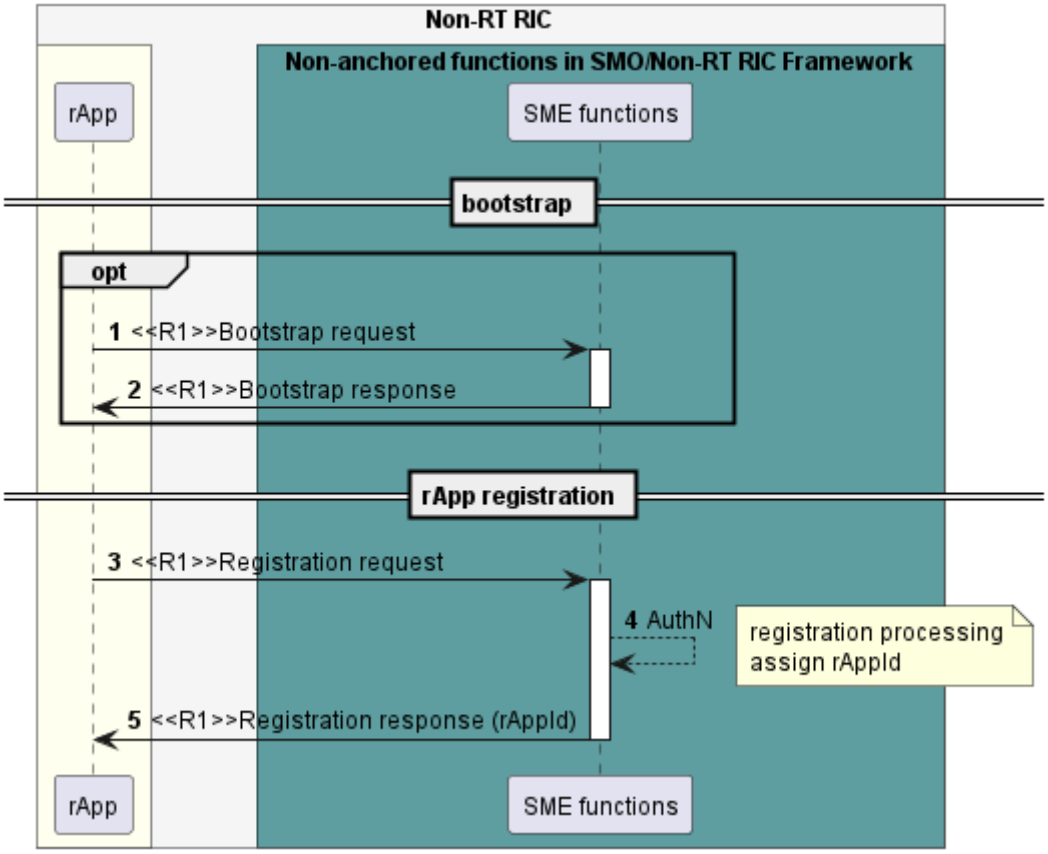

Figure 6.4.1.1-5: rApp registration and bootstrap use case flow diagram

6.5.1 Required data

For registering, an rApp needs to provide the security credentials along with e.g., rApp name, vendor, software version, certificates etc. and the role an rApp is trying to register (Service Producer and/or Service Consumer).

The response to the rApp registration includes the status of registration i.e., success or failure.

For successful rApp registration, the response will also include the rAppId.

For unsuccessful rApp registration, the response will also include the error code along with relevant information.

The rApp may obtain information about the SME services endpoints from the producer of bootstrap service.

6.2 SME Use case 2: Manage Service Registration

6.2.1 Overview

This use case allows an rApp as Service Producer to register each of its produced services, update the service information of each of its registered services and deregister each of its registered service as specified in R1GAP[[1]].

6.2.2 Background and goal of the use case

An rApp as a Service Producer can register each of the new services it producers. Other rApps that are Service Consumers can discover the registered services.

After an rApp as a Service Producer has successfully registered the each of the services it produces, it can update and/or deregister each of the registered services.

6.2.3 Entities/resources involved in the use case

1) SME functions in the role of SME services Producer
    a. support functionality to register a service produced by an rApp,
    b. support authorization of an rApp to determine whether an rApp can register a service, update a registered service, and deregister a registered service,
    c. support validation of the new service produced by an rApp that includes determining whether there are conflicts with services registered earlier,
    d. provide the response of success or failure result to Register service request, Update service registration request and Deregister service request.
2) rApp
    a. supports to initiate the procedure to register a service it produces, update a registered service, and deregister a registered service.

6.2.4 Solutions

6.2.4.1 Register Service

Table 6.2.4.1-6: Register service use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp successfully registers a service.	
Actors and Roles	- rApp in the role of Service Producer that requests service registration. - SME functions in the role of SME services Producer that handles the service registration.	
Assumptions	n/a	
Preconditions	The rApp is deployed and authorized to register a service, and optionally has the bootstrap info.	
Begins when	The rApp determines the need to register a service.	
Step 1 (M)	The rApp requests the SME functions to register a service by providing the rAppId and service profile.	
Step 2 (M)	The SME functions check whether the rApp is authorized to register a service.	
Step 3 (M)	The SME functions validate the information about the new service produced by the rApp, that includes determining whether there are conflicts with services registered earlier.	
Step 4 (M)	The SME functions register the service and make the service endpoint available for discovery.	
Step 5 (M)	The SME functions respond to the rApp with a success result along with a service identifier.	
Ends when	The rApp was able to register a service.	
Exceptions	n/a	
Post Conditions	The rApp will be able to update and de-register the registered service.	
Traceability	REQ-R1-SME-Servicemag-FUN1.	

```
@startuml
'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  "SME functions"  as SME
endbox
rApp->SME:<<R1>>Register service request (rAppId,service profile)
activate SME
  SME --> SME : AuthZ
  note right
  check authorization to
  register a service
  end note
  SME --> SME : Validate
  SME --> SME : register service
```



```
  SME ->rApp:<<R1>>Register service response (service identifier)
deactivate SME
@enduml
```

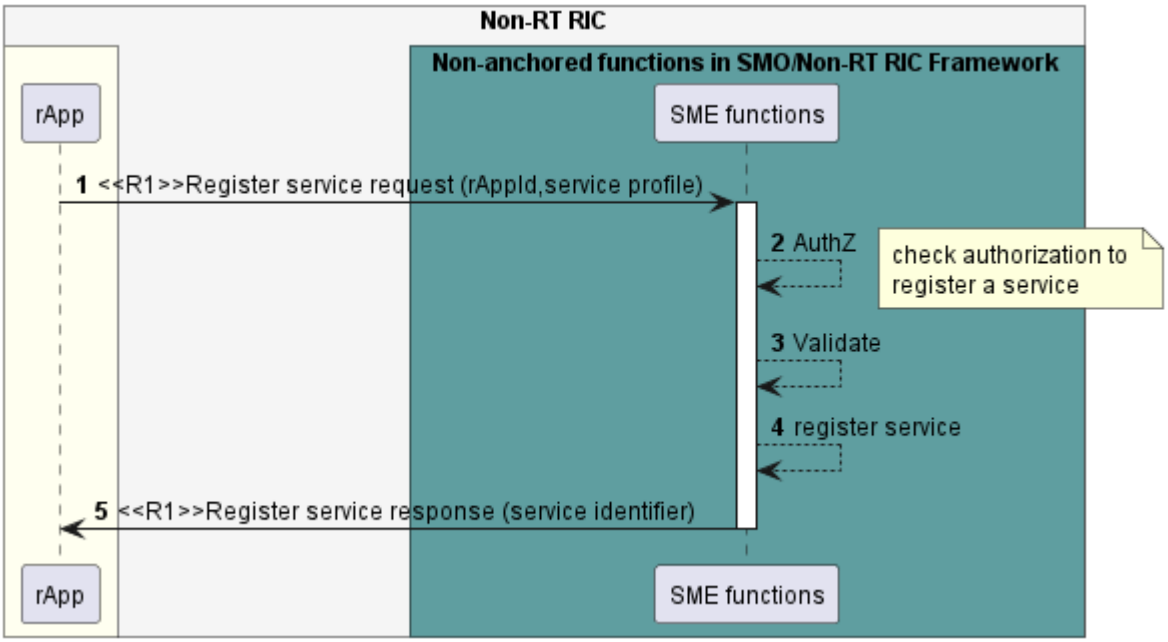

Figure 6.2.4.1-7: Register service use case flow diagram

6.2.4.2 Update service registration

Table 6.2.4.2-8: Update service registration use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp successfully updates a service registration.	
Actors and Roles	- rApp in the role of Service Producer that requests updating the registered information of a service. - SME functions in the role of SME services Producer in SMO/Non-RT RIC framework that handles the update to registered service.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to update service, - The rApp has registered a service.	
Begins when	The rApp determines the need to update the registered information of a service.	
Step 1 (M)	The rApp requests the SME functions to update the service information of registered service by providing rAppId, service identifier and delta service profile.	
Step 2 (M)	The SME functions check whether the rApp is authorized to update a registered service.	
Step 3 (M)	The SME functions update the service information and make the updated service endpoint available for discovery.	
Step 4 (M)	The SME functions respond to rApp with successful result.	
Ends when	The rApp was able to update service registration.	
Exceptions	n/a	
Post Conditions	The rApp will be able to update and de-register the registered service.	
Traceability	REQ-R1-SME-Servicemag-FUN2.	

```
@startuml
```



```
'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  "SME functions"  as SME
endbox
rApp -> SME : <<R1>>Update service registration request (rAppId,service identifier,
service profile)
activate SME
  SME --> SME : AuthZ
note right
check authorization to
update a registered service
end note
  SME --> SME : service update
  SME -> rApp :<<R1>>Update service registration response
deactivate SME
@enduml
```

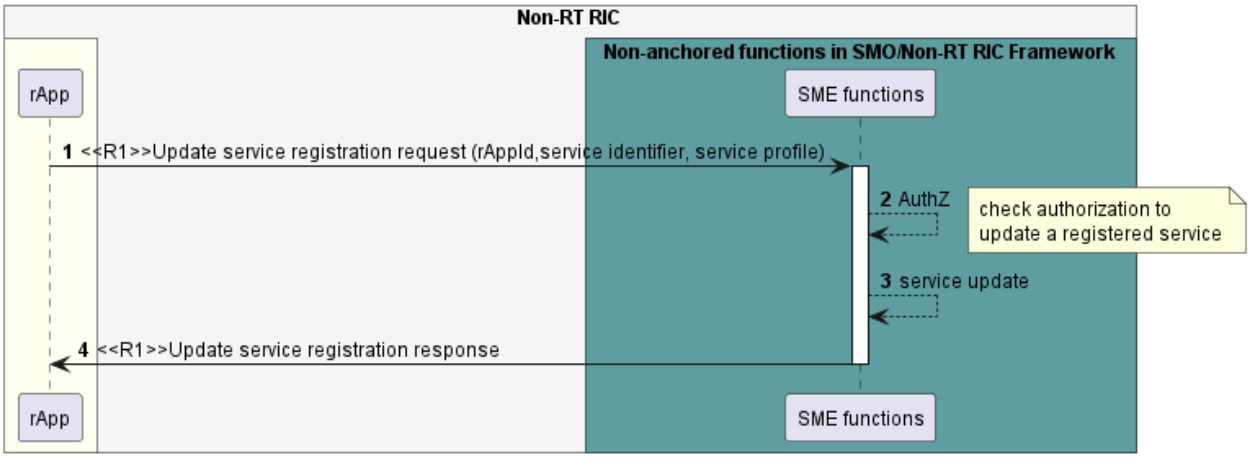

Figure 6.2.4.2-9: Update service registration use case flow diagram

6.2.4.3 Deregister Service

Table 6.2.4.3-10: Deregister service use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp successfully deregister a service.	
Actors and Roles	- rApp in the role of Service Producer that requests service deregistration. - SME functions in the role of SME services Producer that handles the service deregistration.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to deregister a service. - The rApp has previously registered the service that is requested to be deregistered.	
Begins when	The rApp determines the need to deregister a registered service.	

Step 1 (M)	The rApp requests the SME functions to deregister a service by providing the rAppId and service identifier.	
Step 2 (M)	The SME functions check if rApp is authorized to deregister a service.	
Step 3 (M)	The SME functions remove the registered service and remove the service endpoint from the set of end-points that can be discovered.	
Step 4 (M)	The SME functions respond to rApp with a successful result.	
Ends when	The rApp was able to deregister a service.	
Exceptions	n/a	
Post Conditions	The rApp has deregistered a registered service; service endpoint will be no longer available for service discovery.	
Traceability	REQ-R1-SME-Servicemag-FUN3.	

```
@startuml
'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  "SME functions "  as SME
endbox
rApp -> SME : <<R1>>Deregister service request (rAppId,service identifier)
activate SME
SME --> SME : AuthZ
note right
check authorization to
deregister a registered service
end note
  SME --> SME : Remove service profile
  SME -> rApp :<<R1>>Deregister service response
deactivate SME
@enduml
```

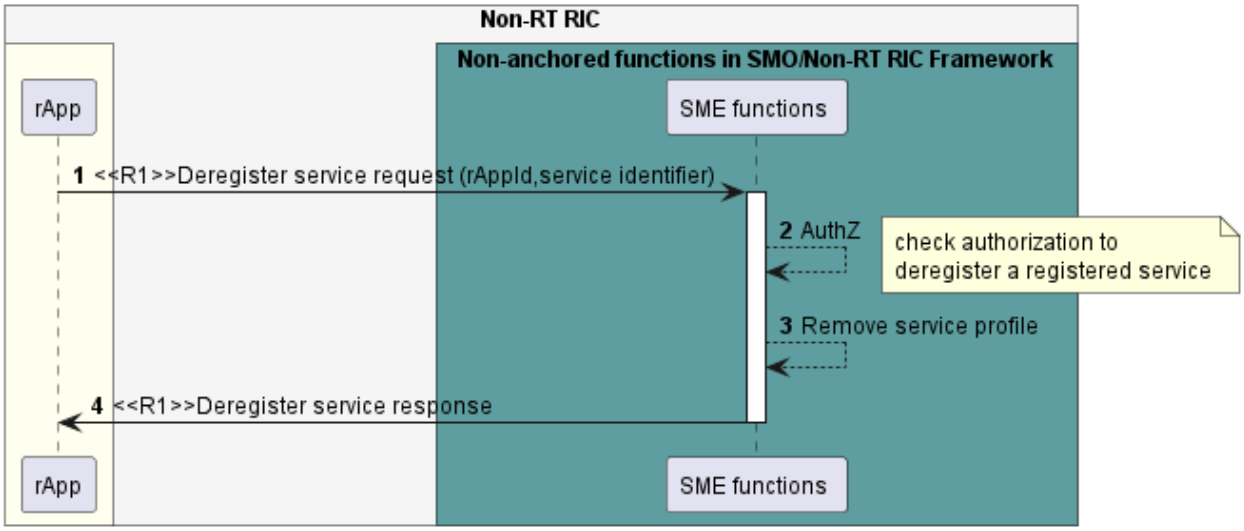

Figure 6.2.4.3-11: Deregister service use case flow diagram

6.2.5 Required data

The service profile information includes e.g., the name of the service, service type, communication mechanism, interface endpoint details (such as IP address, port number, URI, protocol), version number, and data format.

For registering a service, the rApp needs to provide the rAppId and the service profile. On successful registration, the SME services Producer assigns a service identifier for the newly registered service and returns it to the rApp.

For updating a registered service, the rApp needs to provide the rAppId, the service identifier and the delta information that needs to be changed in the service profile.

For de-registering a registered service, the rApp needs to provide the rAppId and the service identifier.

NOTE: It is up to the design of the authorization mechanism whether the rAppId will be passed as a separate piece of information or will be embedded in or implied by the authorization information.

6.3 SME use case 3: Discovery of services

6.3.1 Overview

This use case allows an rApp as a Service Consumer to discover the available services.

6.3.2 Background and goal of the use case

The discovery of services is specified as part of Service management and exposure services in R1GAP[2].

An rApp in the role of Service Consumer will be able to discover the available services.

6.3.3 Entities/resources involved in the use case

1) SME functions in the role of SME services Producer
    a. support functionality to discover the available services,
    b. support authorization of an rApp to determine which services an rApp can discover,
    c. retrieve the stored service information and performs filtering of available services based on selection criteria that may be provided by the rApp,
    d. provide response of success or failure to Discover service request.
2) rApp
    a. supports initiating the procedure to discover available services, optionally providing selection criteria.

6.3.4 Solutions

6.3.4.1 Discover the services

Table 6.3.4.1-12: Discover service use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp discovers the available services.	
Actors and Roles	- rApp in the role of Service Consumer that requests to discover the available services. - SME functions in the role of SME services Producer that handles the discovery of available services.	
Assumptions	n/a	
Preconditions	The rApp is authorized to discover services.	
Begins when	The rApp determines the need to discover the available services.	
Step 1 (M)	The rApp requests the SME functions to discover available services based on the rAppId and optional selection criteria.	
Step 2 (M)	The SME functions check which services the rApp is authorized to discover.	
Step 3 (M)	The SME functions respond with the set of available services and their service identifiers. The list contains only those available services that match the selection criteria if those were provided by the rApp, or all available services otherwise.	
Ends when	The rApp was able to discover the available services.	
Exceptions	n/a	
Post Conditions	The rApp is able to request access to the available services.	
Traceability	REQ-R1-SME-Service Discovery -FUN1.	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  " SME functions"   as SME
endbox
rApp -> SME : <<R1>>Discover service request(rAppId)
activate SME
SME --> SME: AuthZ
note right
check authorization to
discover available services
end note
  SME -> rApp :<<R1>> Discover service response(list of services)
deactivate SME
@enduml
```


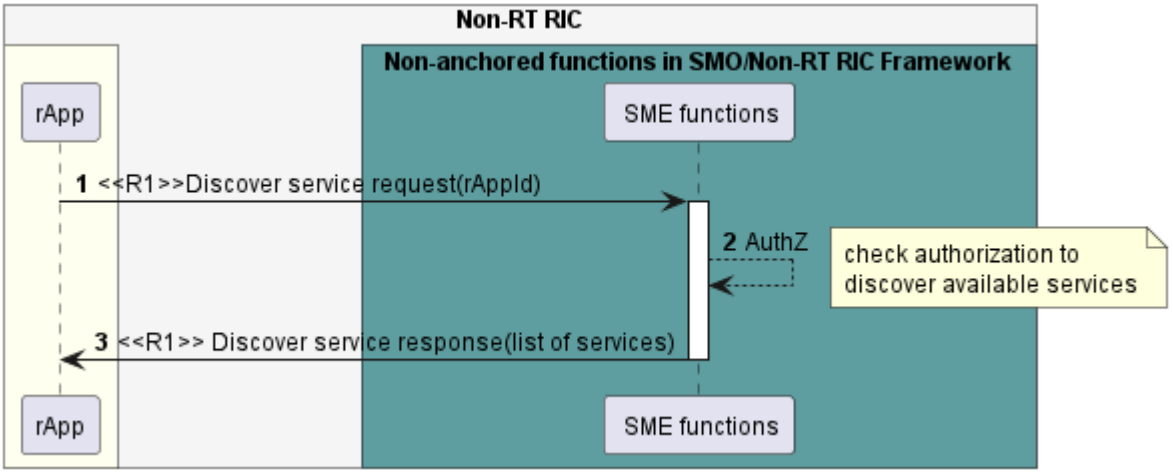

Figure 6.3.4.1-13: Discover service use case flow diagram

6.3.5 Required data

For discovering a service, the rApp needs to provide the rAppId and optional selection criteria such as e.g., name of the service, service type, and capabilities.

The SME functions respond with information about a set of services which includes e.g., endpoint information and service identifiers.

NOTE: It is FFS whether an rApp may be allowed to discover services for which it is currently not authorized but might be able to obtain authorization later.

6.4 SME use case 4: Subscribe service availability

6.4.1 Overview

This use case allows an rApp as Service Consumer to subscribe to notifications and unsubscribe from notifications regarding changes in service availability.

6.4.2 Background and goal of the use case

The subscribe service availability and unsubscribe service availability procedures are defined as part of Service management and exposure services in R1GAP [2].

The rApp in the role of Service Consumer can subscribe to notifications and unsubscribe from notifications regarding changes in the availability of services.

6.4.3 Entities/resources involved in the use case

1) SME functions in the role of SME services Producer

   a) Support functionality to subscribe to and unsubscribe from service availability notifications by an rApp,

   b) Support notifying the rApp regarding the changes in the availability of services,

   c) Provide response of success or failure to subscribe service availability request and unsubscribe service availability request.

2) rApp

a) Supports functionality to initiate requests to subscribe to and unsubscribe from service availability notifications,

b) Supports functionality to accept notifications of the changes in service availability from the SME services Producer.

6.4.4 Solutions

6.4.4.1 Subscribe to service availability notifications.

Table 6.4.4.1-14: Subscribe to service availability notifications

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp subscribes to notifications about changes of the availability of services.	
Actors and Roles	- rApp in the role of Service Consumer that requests subscription to service availability notifications. - SME functions in the role of SME services Producer entity that handles the subscription requests.	
Assumptions	n/a	
Preconditions	The rApp is authorized to subscribe to notifications about changes in service availability.	
Begins when	The rApp determines to subscribe to notifications, regarding changes in the availability of services.	
Step 1 (M)	The rApp requests the SME functions to subscribe to notifications regarding changes in the available services with rAppId and optional service identifiers.	
Step 2 (M)	The SME functions check whether the rApp is authorized to send a subscription request.	
Step 3 (M)	The SME functions respond with subscription identifier.	
Ends when	The rApp was able to subscribe to notifications.	
Exceptions	n/a	
Post Conditions	1. The rApp can receive notifications when there are any changes in the services availability. 2. The rApp can unsubscribe from service availability notifications.	
Traceability	REQ-R1-SME-Servicesub-FUN1.	

```
@startuml
'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  "SME functions "  as SME
endbox
autonumber
rApp -> SME : <<R1>> Subscribe service availability request(rAppId)
activate SME
 SME --> SME : AuthZ
```



```
note right
check authorization to
send a subscription request
end note
   SME -> rApp : <<R1>> Subscribe service availability response (subscription identifier)
deactivate SME
@enduml
```

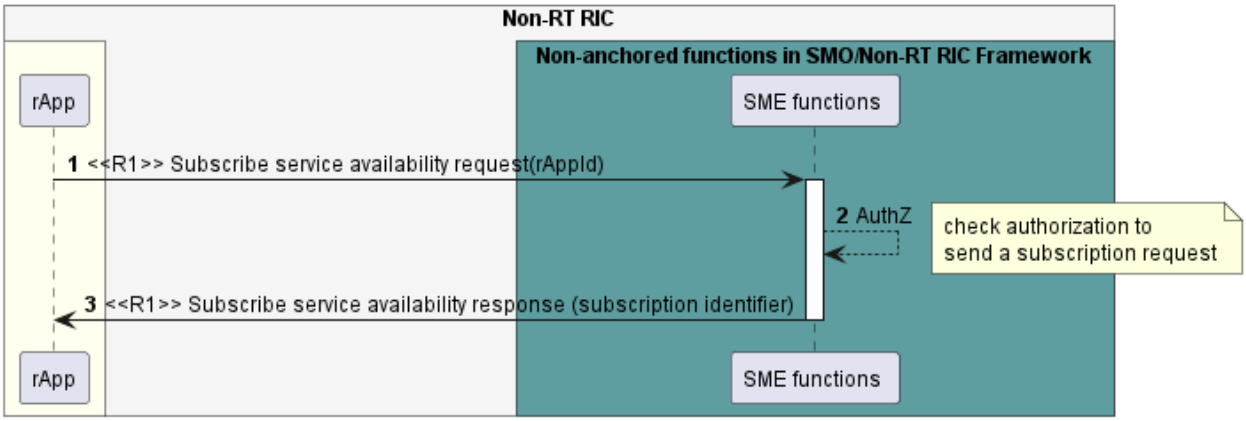

Figure 6.4.4.1-15: Subscribe to of services availability notifications use case flow diagram

6.4.4.2 Notify service availability changes

Table 6.4.4.2-16: Notify service availability changes use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp receives notifications about changes in the available services.	
Actors and Roles	- rApp in the role of Service Consumer that receives notifications on service availability changes. - SME functions in the role of SME services Producer that sends notifications on service availability changes.	
Assumptions	n/a	
Pre conditions	The rApp has subscribed to notifications on service availability changes.	
Begins when	The SME functions determine to send notifications, regarding changes in the available service/s to subscribed rApp.	
Step 1 (M)	The SME functions detect changes in service availability (due to service registration, service update, service deregistration and heartbeat failure).	
Step 2 (M)	The SME functions send notifications regarding the changes to be subscribed rApp with subscription identifier and information about service changes.	
Step 3(M)	The rApp processes the changes.	
Ends when	The rApp was able to process the changes in the available services.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-SME-Servicesub-FUN2.	

```
@startuml
'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
```



```
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  "SME functions "  as SME
endbox
autonumber
SME --> SME : Trigger changes in the service/s
SME -> rApp : <<R1>> notify changes \n(subscription identifier, service identifier/s)
rApp --> rApp : Process the changes
@enduml
```

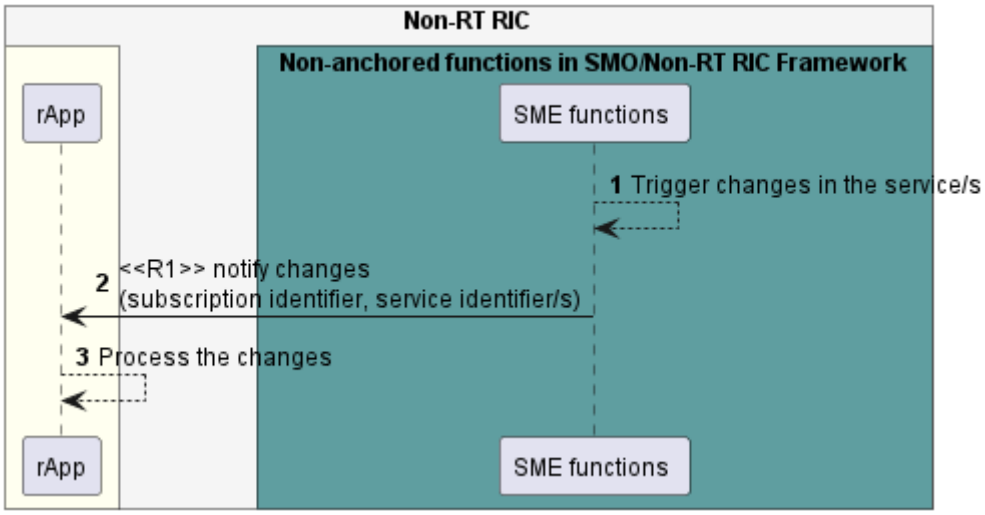

Figure 6.4.4.2-17: Notify service availability changes use case flow diagram

6.4.4.3 Unsubscribe from service availability notifications

Table 6.4.4.3-18: Unsubscribe from service availability notifications

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp unsubscribes from service availability notifications.	
Actors and Roles	- rApp in the role of Service Consumer that requests unsubscribe from service availability notifications. - SME functions in the role of SME services Producer that process the unsubscribe service availability request.	
Assumptions	n/a	
Pre conditions	- The rApp is authorized to unsubscribe from service availability notifications. - The rApp has subscribed to notification changes.	
Begins when	The rApp determines to unsubscribe from service availability notifications.	
Step 1 (M)	The rApp requests the SME functions to unsubscribe from service availability notifications with rAppId and subscription identifier.	
Step 2 (M)	The SME functions check whether the rApp is authorized to unsubscribe.	
Step 3 (M)	The SME functions send a response.	
Ends when	The rApp was able to unsubscribe from service availability notifications.	
Exceptions	n/a	
Post Conditions	The rApp is not subscribed to service availability notifications.	
Traceability	REQ-R1-SME-Servicesub-FUN3.	

```
@startuml
'https://plantuml.com/sequence-diagram
```



```
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant rApp as rApp
endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant  "SME functions "  as SME
endbox
autonumber
rApp -> SME : <<R1>> Unsubscribe from service availability notifications
request(subscription identifier)
activate SME
   SME -> SME : AuthZ
note right
check authorization to
unsubscribe
end note
   SME -> rApp : <<R1>> Unsubscribe from service availability notifications response
deactivate SME
@enduml
```

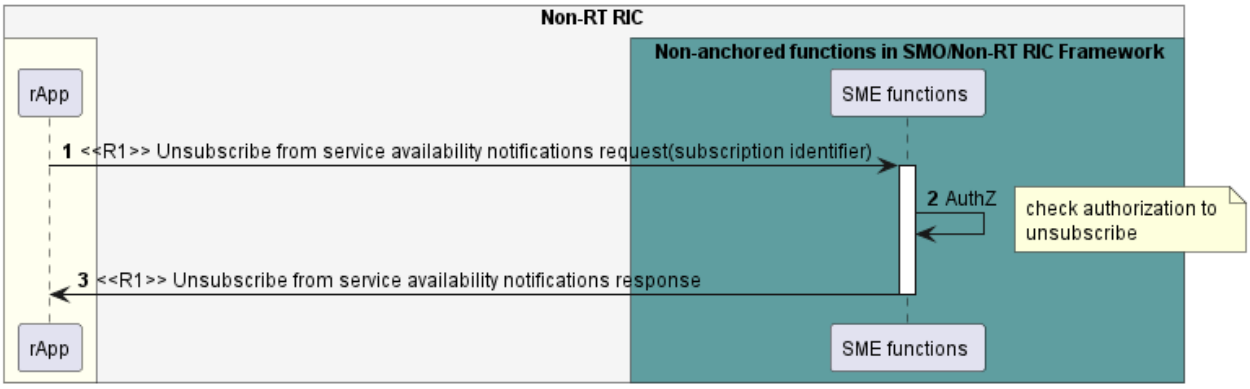

Figure 6.4.4.3-19: Unsubscribe service availability use case flow diagram

6.4.5 Required data

To receive notifications regarding changes in service availability, the rApp as Service Consumer subscribes to notifications by providing rAppId and optional service identifier/s. When the SME functions have successfully processed the request, they responds by providing a subscription identifier.

The SME functions send notifications to the subscribed rApp when there are changes in the availability of services by providing subscription identifier, service identifiers and information about the changes.

For unsubscribing from notifications, a subscribed rApp needs to provide subscription identifier.

7 Use cases for Data Management and Exposure Services

7.1 DME use case 1: Data registration and deregistration

7.1.1 Overview

This use case defines how an rApp registers and deregisters as producer of data for a data type.

7.1.2 Background and goal of the use case

The Register data type and Deregister data type procedures are defined as part of Data management and exposure services in R1GAP[[1]].

7.1.3 Entities/resources involved in the use case

1) Data management and exposure functions
    a) Supports functionality to allow an rApp to register and deregister as producer of data for a data type,
    b) Supports functionality to allow an rApp to register the constraints for how it can produce and deliver data for the registered data type,
    c) Supports validation of data type information (e.g., schema validation).
2) rApp
    a) Initiates the procedure to register and deregister as producer of data for a data type.

7.1.4 Solutions

7.1.4.1 Register data type

Table 7.1.4.1-1: Data type registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp registers as producer of data for a data type.	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the Data management and exposure services. - The rApp not registered as producer of data for the data type that they intend to register in Data management and exposure functions.	
Begins when	The rApp determines the need to register as a producer of data for a data type.	
Step 1 (M)	The rApp requests the Data management and exposure functions to register as producer of data for a data type by providing the rAppId and data type information.	
Step 2 (M)	The Data management and exposure functions check with SME functions whether the rApp is authorized to register as producer of data.	
Step 3 (M)	The Data management and exposure functions validate the data type information.	
Step 4(M)	The Data management and exposure functions register the rApp as data producer of the data type. Further, in case this rApp is the first producer of data that registers for the data type, the Data management and exposure functions add the data type to the set of discoverable data types.	
Step 5(M)	The Data management and exposure functions respond to rApp with successful data type registration along with data type registration identifier.	
Ends when	The rApp was able to register as producer of data for a data type.	
Exceptions	n/a	
Post Conditions	The data type is discoverable.	
Traceability	REQ-R1-DME-Regdatatype-FUN1	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant rApp as rApp
   endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
    endbox
rApp -> DME :<<R1>>Register data type request (rAppId,\n{data type information})
activate DME
DME --> DME :AuthZ
```



```
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME :Validate
note right
validation of data type information
end note
DME --> DME :Register data type
DME -> rApp :<<R1>>Register data type response (data type registration identifier)
deactivate DME
@enduml
```

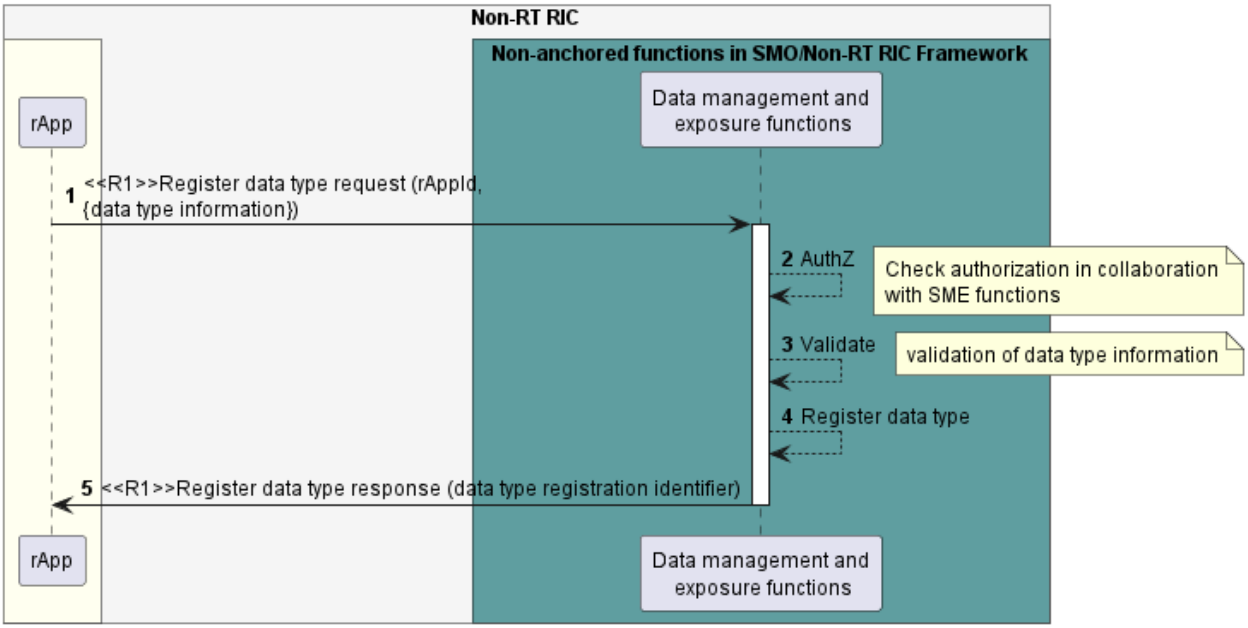

Fig 7.1.4.1-1: Register data type use case flow diagram

7.1.4.2 Deregister data type

Table 7.1.4.2-1: Data type deregistration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp deregisters as producer of data for a data type.	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer.	
Assumptions	The rApp is not producing any data for the data type	
Preconditions	The rApp has registered as producer of data for a data type. The rApp is authorized to access the Data management and exposure services.	
Begins when	The rApp determines the need to deregister as producer of data for a data type that it no longer intends to produce.	
Step 1 (M)	The rApp requests Data management and exposure functions to be deregistered as producer of data for a data type by providing rAppId and data type registration identifier.	
Step 2(M)	The Data management and exposure functions check with SME functions whether the rApp is authorized to deregister as producer of data.	
Step 3 (M)	The Data management and exposure functions remove the registration of the rApp as producer of data for the data type. Further, in case no other producer of data for the data type is registered, the Data management and exposure functions remove the data type from the set of discoverable data types.	
Step 4 (M)	The Data management and exposure functions respond to the rApp with successful deregistration.	
Ends when	The rApp was able to deregister the as producer of data for the data type.	
Exceptions	n/a	
Post Conditions	The rApp will not receive any request of data for the data type and if no other producer is registered, the data type is not discoverable any longer.	
Traceability	REQ-R1-DME-Regdatatype-FUN1	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
     participant rApp as rApp
   endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
     endbox
rApp -> DME :<<R1>>Deregister data type request \n(rAppId,data type registration
identifier)
activate DME
DME --> DME :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME :Deregister data type
DME -> rApp :<<R1>>Deregister data type response
deactivate DME
@enduml
```


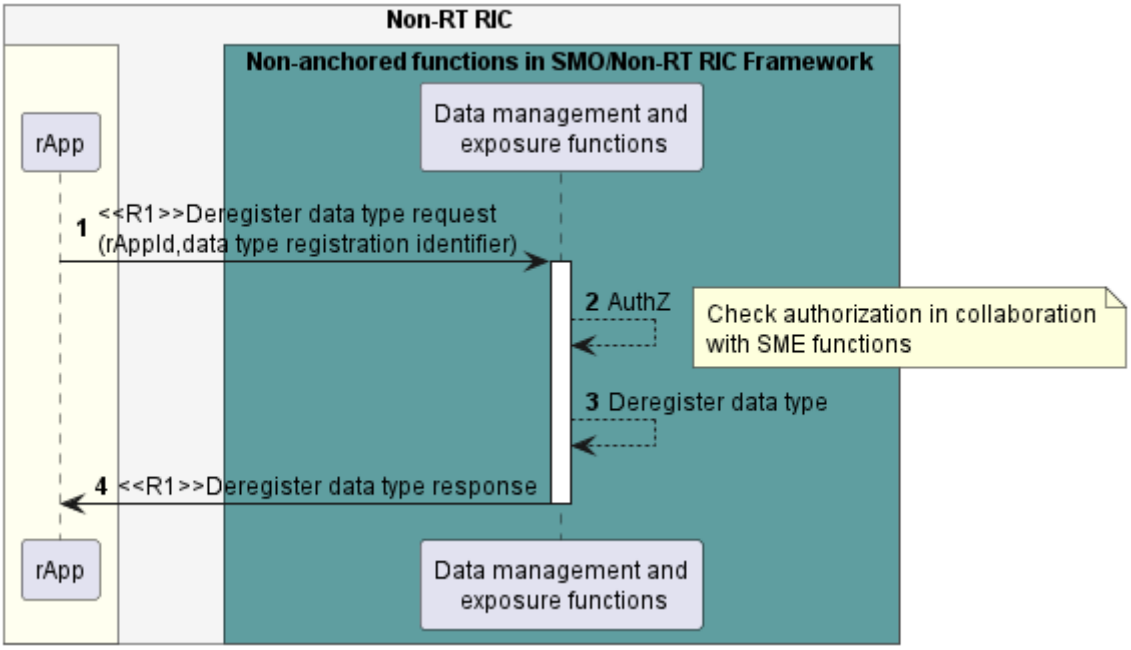

Figure 7.1.4.2-1: Deregister data type use case flow diagram

7.1.5 Required data

The data type information includes the data type identifier, metadata describing the data type, the schemas for the data content, and constraints for how data can be produced, delivered and accessed.

For registering a data type, the rApp provides the rAppId and data type information. On successful registration, Data management and exposure functions provide a data type registration identifier for the registration of the rApp as producer of data for the data type.

For deregistering a data type, the rApp provides the rAppId and the data type registration identifier.

NOTE: Data management and exposure functions are not required to check whether the rApp is not producing any data for the data type it tries to deregister.

7.2 DME use case 2: Discovery of data types

7.2.1 Overview

This use case enables an rApp to retrieve information about registered and available data types and query a data type identifier and its associated information. It also enables an rApp to subscribe to and unsubscribe from notifications regarding changes in the availability of data types.

7.2.2 Background and goal of the use case

The discover data type and query data type information , subscribe data types changes, unsubscribe data types changes and notify data types changes procedure are defined as part of the Data management and exposure services in R1GAP[[1]].

7.2.3 Entities/resources involved in the use case

1) Data management and exposure functions in the role of Data registration and discovery service Producer,

a) Support functionality allowing rApps to discover the data types that are available,

b) Support functionality allowing rApps to retrieve the information about a specific data type identified by a data type identifier,

c) Support functionality allowing rApps to subscribe to and unsubscribe from notifications regarding the availability of data types,

d) Support functionality to notify rApps about changes in the availability of data types.

2) rApp in the role of Data registration and discovery service Consumer.

a) Initiates the procedure to discover data types and query data type information, subscribe data types changes and unsubscribe data types changes,

b) Supports functionality to receive notifications regarding changes in the availability of data types.

7.2.4 Solutions

7.2.4.1 Discover data type

Table 7.2.4.1-1: Discover data type

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves the information on available data types	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer	
Assumptions	n/a	
Preconditions	– The rApp is authorized to access the Data registration and discovery services. – At least one data type is registered and available with the Data management and exposure functions.	
Begins when	The rApp determines the need to discover the available data types.	
Step 1 (M)	The rApp requests the Data management and exposure functions for the information on the available data types by providing rAppId and optional filter criteria.	
Step 2(M)	The Data management and exposure functions validate if the rApp is authorized to discover the available data types.	
Step 3(M)	The Data management and exposure functions provide the information about available data types. The list contains only those available data types that match the filtering criteria if those were provided by the rApp, or all available data types otherwise. For each data type, the data type identifier and meta data are provided.	
Ends when	The rApp was able to discover the available data types.	
Exceptions	n/a	
Post Conditions	The rApp can retrieve specific information related to a data type by providing its data type identifier.	
Traceability	REQ-R1-DME-discoverydatatype-FUN1	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
```



```
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
     participant rApp as rApp
   endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
     endbox
rApp -> DME :<<R1>> Discover data type identifiers request\n(rAppId, Query information)
activate DME
DME --> DME :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME -> rApp :<<R1>>Discover data type identifiers response\n(data type identifiers and
metadata)
deactivate DME
@enduml
```

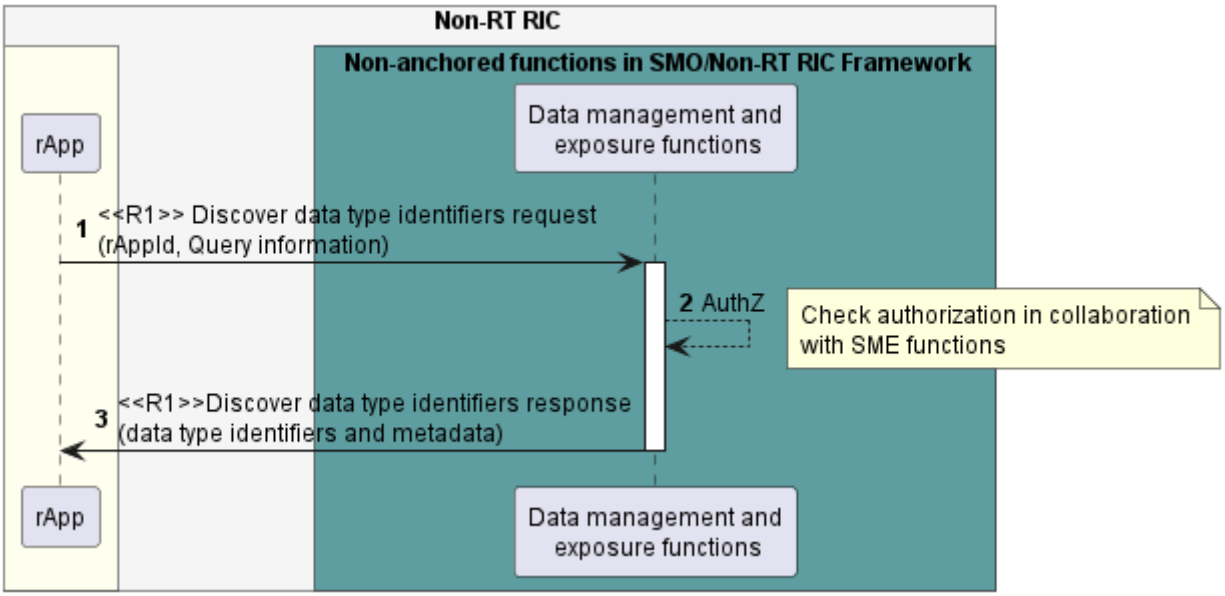

Figure 7.2.4.1-1: Discover data type use case flow diagram

7.2.4.2 Query data type information

Table 7.2.4.2-1: Query data type information

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves detailed information on a specific data type.	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer.	
Assumptions	n/a	
Preconditions	– The rApp is authorized to access the Data registration and discovery service. – The rApp is aware of the data type identifier of the data type about which it intends to retrieve detail information.	5.2.4.1
Begins when	The rApp determines the need to retrieve the information on a specific data type.	
Step 1 (M)	The rApp requests the Data management and exposure functions for the information on a specific data type by providing the data type identifier.	
Step 2(M)	The Data management and exposure functions validate if the rApp is authorized to retrieve the information.	
Step 3(M)	The Data management and exposure functions provide the information on the requested data type identifier	
Ends when	The rApp has all the information of a data type	
Exceptions	n/a	
Post Conditions	The rApp can formulate a request or subscription for data of that data type.	
Traceability	REQ-R1-DME-discoverydatatype-FUN1	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant rApp as rApp
   endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
     endbox
rApp -> DME :<<R1>>Query data type information request \n(data type identifier)
activate DME
DME --> DME :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME -> rApp :<<R1>> Query data type information response \n(data type information)
deactivate DME
@enduml
```


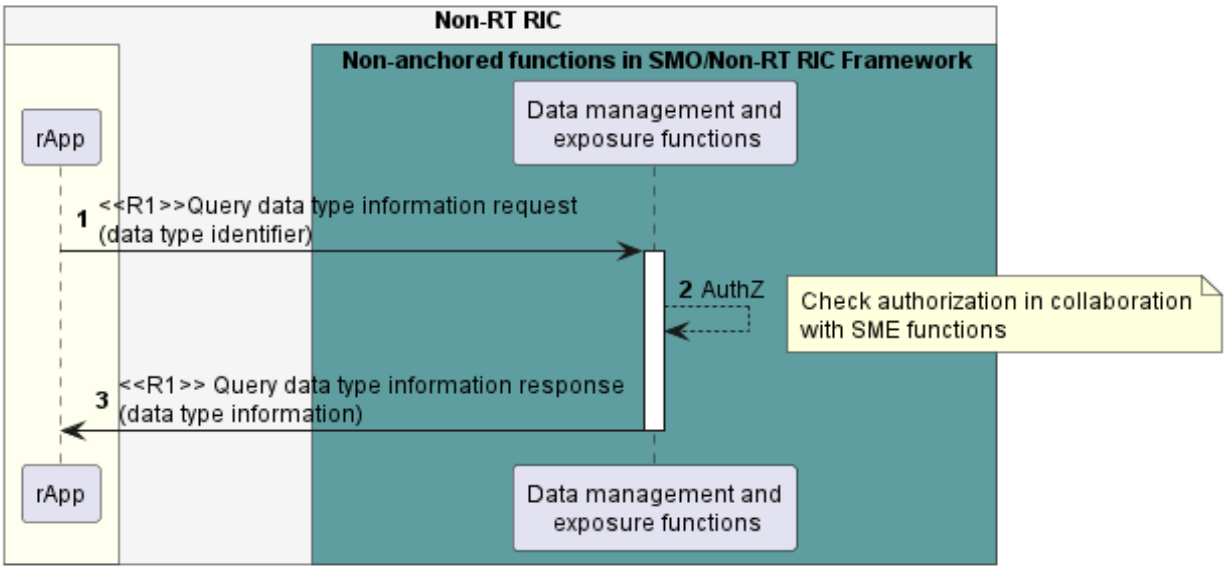

Figure 7.2.4.2-1: Query data type information use case flow diagram

7.2.4.3 Subscribe data types changes

Table 7.2.4.3-1: Subscribe data types changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp subscribes to notifications regarding changes in the availability of data types.	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorised to access the Data registration and discovery service. - The rApp is aware of the data type identifier of the data type.	5.2.4.1
Begins when	The rApp determines the need to subscribe to notifications regarding changes in the availability of data types.	
Step 1 (M)	The rApp requests the Data management and exposure functions to subscribe to notifications regarding data types availability changes by providing the rAppId and data type identifier(s).	
Step 2(M)	The Data management and exposure functions validate if the rApp is authorized to subscribe to notifications.	
Step 3(M)	The Data management and exposure functions create the subscription.	
Step 4(M)	The Data management and exposure functions respond to the request with subscription ID.	
Ends when	The rApp was able to subscribe to notifications regarding data types changes.	
Exceptions	n/a	
Post Conditions	1) The rApp can receive notifications when there are any changes in the set of availabile data types. 2) The rApp can unsubscribe from the notifications.	
Traceability	n/a	

```
@startuml
```



```
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant rApp as rApp
  endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
  endbox
rApp -> DME :<<R1>>Subscribe data types changes request \n(rAppId, data type
identifier(s))
activate DME
DME --> DME :Authz
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME : create subscription
DME -> rApp :<<R1>> Subscribe data types changes response \n(subscription ID)
deactivate DME
@enduml
```

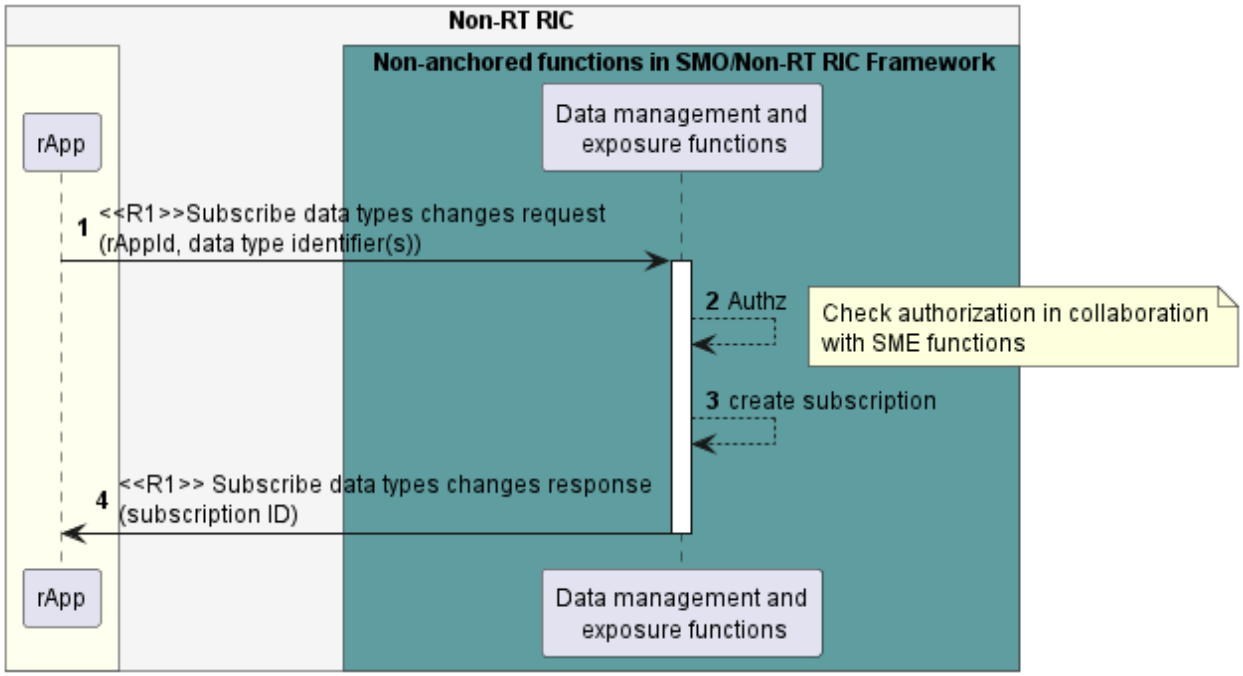

Figure 7.2.4.3-1: Subscribe data types changes use case flow diagram

7.2.4.4 Notify data types changes

Table 7.2.4.4-1: Notify data types changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp receives a notification regarding a change in the availability of a data types.	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorised to access the Data registration and discovery service. - The rApp is subscribed to notifications.	5.2.4.3
Begins when	The Data management and exposure functions determines to send a notification regarding a change in the availability of a data types to the subscribed rApp.	
Step 1(M)	The Data management and exposure functions sends a notification to the subscribed rApp with subscription ID, data type identifier(s) and information about the changes in the availability of data types	
Ends when	The rApp was able to receive the notification.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	n/a	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant rApp as rApp
   endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
  endbox
DME -> rApp : <<R1>> Notify data types changes(subscription ID, data type identifier(s),
\navailability change details)
@enduml
```

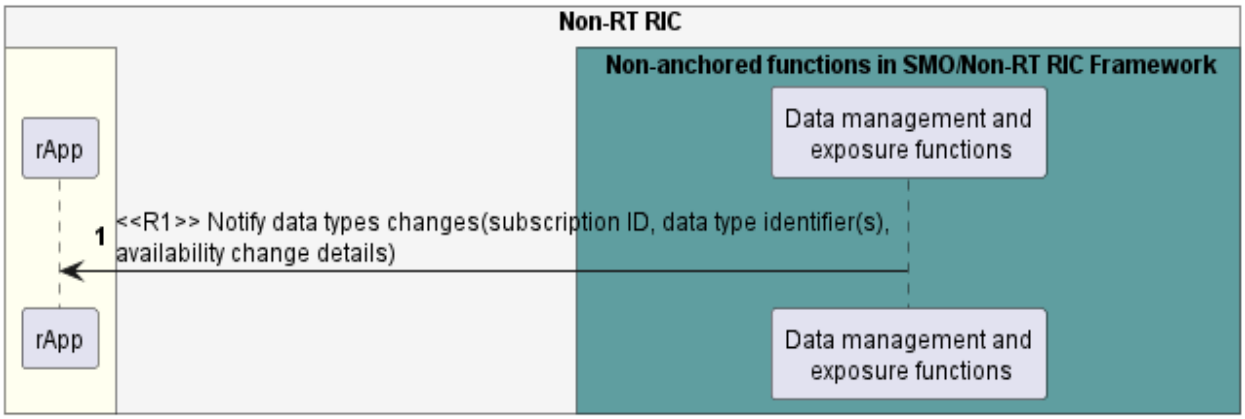

Figure 7.2.4.4-1: Notify data types changes use case flow diagram

7.2.4.5 Unsubscribe data types changes

Table 7.2.4.5-1: Unsubscribe data types changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp unsubscribes from notifications.	
Actors and Roles	- rApp in the role of Data registration and discovery service Consumer. - Data management and exposure functions in the role of Data registration and discovery service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorised to access the Data registration and discovery service. - The rApp is subscribed to notifications regarding data types changes.	5.2.4.3
Begins when	The rApp determines the need to unsubscribe from notifications regarding changes in the availability of data types.	
Step 1 (M)	The rApp requests the Data management and exposure functions to unsubscribe from notifications regarding data types changes by providing the rAppId and subscription ID.	
Step 2(M)	The Data management and exposure functions validate if the rApp is authorized to unsubscribe from notifications.	
Step 3(M)	The Data management and exposure functions cancel the subscription	
Step 4(M)	The Data management and exposure functions respond to the request.	
Ends when	The rApp was able to unsubscribe from notifications regarding changes in the availability of data types.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	n/a	

```
@startuml
 'https://plantuml.com/sequence-diagram
 !pragma teoz true
 skinparam ParticipantPadding 5
 skinparam BoxPadding 10
 skinparam defaultFontSize 12
 skinparam lifelineStrategy solid
 autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant rApp as rApp
   endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant "Data management and\n exposure functions" as DME
  endbox
rApp -> DME :<<R1>>Unsubscribe data types changes request(rAppId, \nsubscription ID)
activate DME
DME --> DME :Authz
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME : cancel subscription
DME -> rApp :<<R1>> Unsubscribe data types changes response
```



```
deactivate DME
@enduml
```

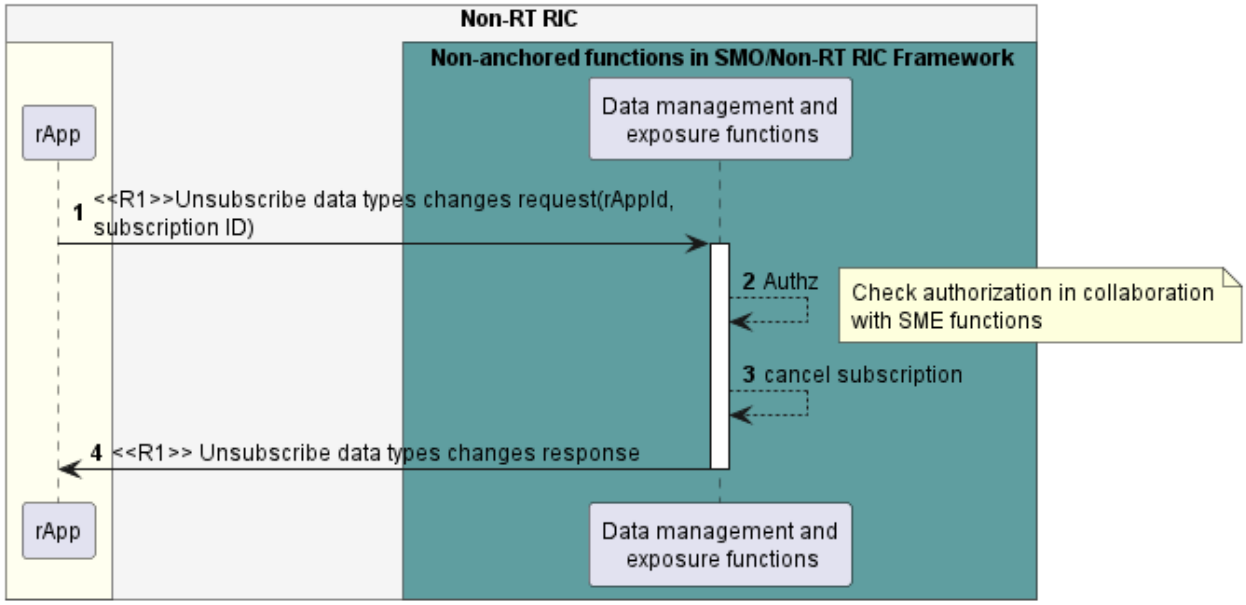

Figure 7.2.4.5-1: Unsubscribe data types changes use case flow diagram

7.2.5 Required data

For the Discover data types request, the rApp provides the rAppId and optional query information.

The Data management and exposure functions respond with information about all available data types which includes for each data type, the data type identifier and related metadata information. If filtering criteria information was provided in the related request, the response only contains information about those data types that match the filtering criteria.

For the Query data type information request, the rApp provides the data type identifier.

The Data management and exposure functions respond with information related to the data type.

For subscription to notifications regarding changes in the availability of data types, the rApp provides the rAppId and the data type identifier(s). The Data management and exposure functions respond with a unique subscription ID.

The Data management and exposure functions sends notifications to the subscribed rApp by providing the Subscription ID, data type identifier(s) and related information about the changes in the availability of data types.

For unsubscribing from notifications regarding changes in the availability of data types, the subscribed rApp needs to send the rAppId and subscription ID.

7.3 DME use case 3: Data offer

7.3.1 Overview

This use case allows a Data Producer rApp as Service Consumer to create and terminate a data offer to trigger the collection of a data instance it wishes to deliver to the Data management and exposure functions as Data management and exposure services Producer for collection.

Further, it allows the Data management and exposure functions as Data management and exposure services Producer to indicate to the Data Producer rApp as Service Consumer that they intend to stop collecting the data instance related to a data offer.

7.3.2 Background and goal of the use case

A Data Producer rApp as Service Consumer can create a data offer which indicates to the Data management and exposure functions that the rApp intends to deliver a data instance to the Data management and exposure services Producer for collection.

Further, the Data Producer rApp as Service Consumer can terminate a data offer it has created earlier to indicate to the Data management and exposure functions that the Data Producer no longer wishes to deliver the data instance for collection.

Finally, the Data management and exposure functions can indicate to the Data Producer rApp as Service Consumer via a data offer termination notification that they will stop collecting a data instance and will terminate the related data offer created earlier by the Data Producer rApp.

7.3.3 Entities/resources involved in the use case

1) Data management and exposure functions as Data management and exposure services Producer
    a. support functionality to create a data offer as requested by a Data Producer rApp,
    b. support authorization of a Data Producer rApp to determine whether it can create a data offer,
    c. support to send data offer termination notifications,
    d. support to receive the offered data from the rApp and store them,
    e. provide the response of success or failure result to the create data offer request and the terminate data offer request.
2) rApp
    a. supports initiating the procedure to create a data offer indicating its intent to deliver data for collection and subsequent storage,
    b. supports to receive data offer termination notifications,
    c. supports to request terminating a data offer created earlier,
    d. supports to produce and deliver the offered data.

7.3.4 Solutions

7.3.4.1 Create data offer

Table 7.3.4.1-1: Create data offer use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp creates a data offer to specify a data instance it produces for collection.	
Actors and Roles	- rApp in the role of Data Producer that requests data offer creation. - Data management and exposure functions in the role of Data management and exposure services Producer that handles the data offer creation.	
Assumptions	n/a	
Preconditions	The rApp is deployed and is authorized to access the Data management and exposure functions.	
Begins when	The rApp determines the need to create a data offer.	
Step 1 (M)	The rApp requests the Data management and exposure functions to create a data offer by providing the rAppId, registered data type identifier, further information about the data instance, and intended delivery method(s).	
Step 2 (M)	The Data management and exposure functions check whether the rApp is authorized to create a data offer.	
Step 3 (M)	The Data management and exposure functions validate the information provided with the request.	
Step 4 (M)	The Data management and exposure functions create the data offer and prepare an endpoint for the delivery of notifications or push data.	
Step 5 (M)	The Data management and exposure functions respond to the rApp with a success result along with the data offer identifier and endpoint for the delivery of notifications or push data.	
Ends when	The rApp was able to create a data offer.	
Exceptions	n/a	
Post Conditions	The Data management and exposure functions are ready to collect the data instance related to the data offer and the rApp is ready to produce it. Further, the rApp and the Data management and exposure services Producer will be able to terminate the data offer.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 70
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "rApp" as rapp
  endbox
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
      participant "Data management and\n exposure functions" as dme
       endbox
endbox
```



```
rapp -> dme: <<R1>> Create data offer request \n(rAppId, dataTypeId, dataInstanceInfo,
deliveryMethods, notifyEndpointInfo)
activate dme
dme --> dme: AuthZ
note right
     Check authorization in collaboration
     with SME functions
end note
dme --> dme: Validate request
dme --> dme: Create data offer\nand prepare\ndelivery endpoint
dme -> rapp: <<R1>> Create data offer response (dataOfferId, deliveryEndpointInfo)
deactivate dme
@enduml
```

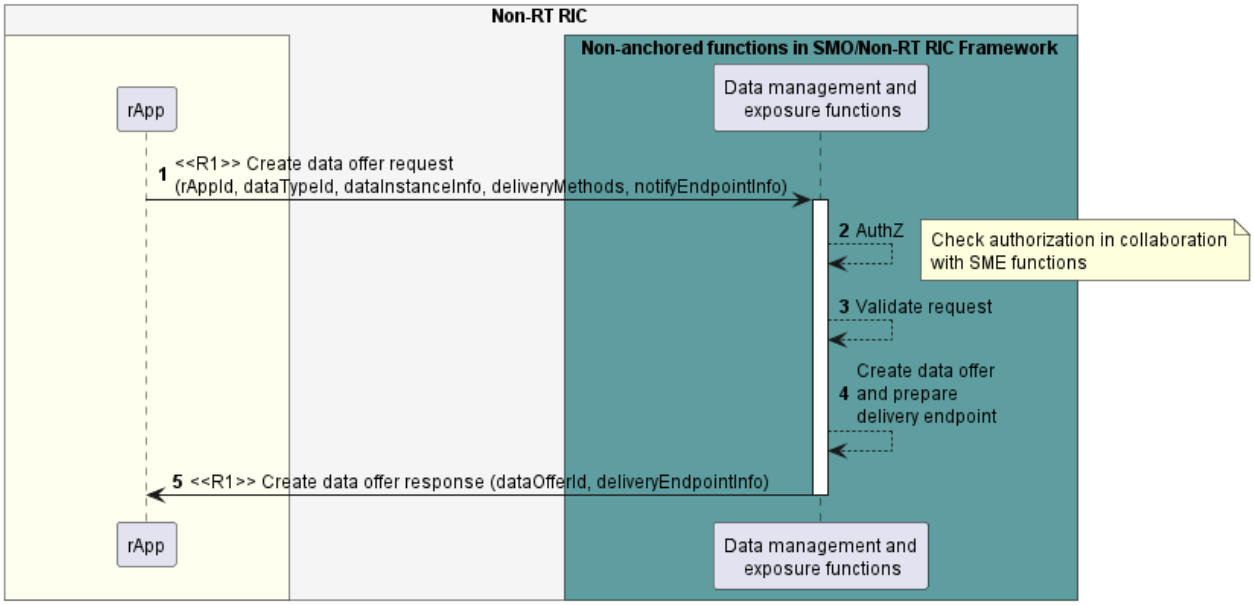

Figure 7.3.4.1-1: Create data offer use case flow diagram

7.3.4.2 Deliver offered data

Table 7.3.4.2-1: Deliver offered data use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp delivers a data instance it produces for collection, based on a data offer created earlier.	
Actors and Roles	- rApp in the role of Data Producer that produces an offered data instance. - DME functions in the role of Data management and exposure services Producer that collect the offered data instance.	
Assumptions	n/a	
Preconditions	A data offer has been created.	
Begins when	The rApp starts delivering the produced data related to the data offer.	
Alternative procedure	Each time a set of data becomes available for delivery, Steps 3-5 are executed if pull is used as the data delivery method.	
Step 1 (M)	The Data Producer rApp sends to the Data management and exposure functions a Notify data availability message, including details how to pull the requested data and the data offer identifier.	
Step 2 (M)	The Data management and exposure functions send a pull data request message, using the pull delivery details obtained in step 3	
Step 3 (M)	The Data Producer rApp responds with a pull data response, including the data to be collected.	
Alternative procedure	Each time a set of data becomes available for delivery, Step 6 is executed if push is used as the data delivery method.	
Step 4 (M)	The Data Producer rApp sends to the Data management and exposure functions a push data message, including the data to be collected and the subscription identifier.	
Step 5 (M)	The Data management and exposure functions store the delivered data.	
Ends when	The rApp has delivered to the Data management and exposure functions for collection the produced data related to the data offer.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
box #ivory
participant "rApp" as rApp
end box

box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and\nexposure functions" as dme
end box

loop when data becomes available
     alt pull data delivery method
```



```
          rApp -> dme : <<R1>> Notify data availability (pull delivery details, data
offer ID)
          dme -> rApp : <<R1>> Pull data request (pull delivery details)
          rApp -> dme : <<R1>> Pull data response (with data payload)
     else push data delivery method
          rApp -> dme : <<R1>> Push data message (data payload, data offer ID)
     end
     dme --> dme: Store data
end
@enduml
```

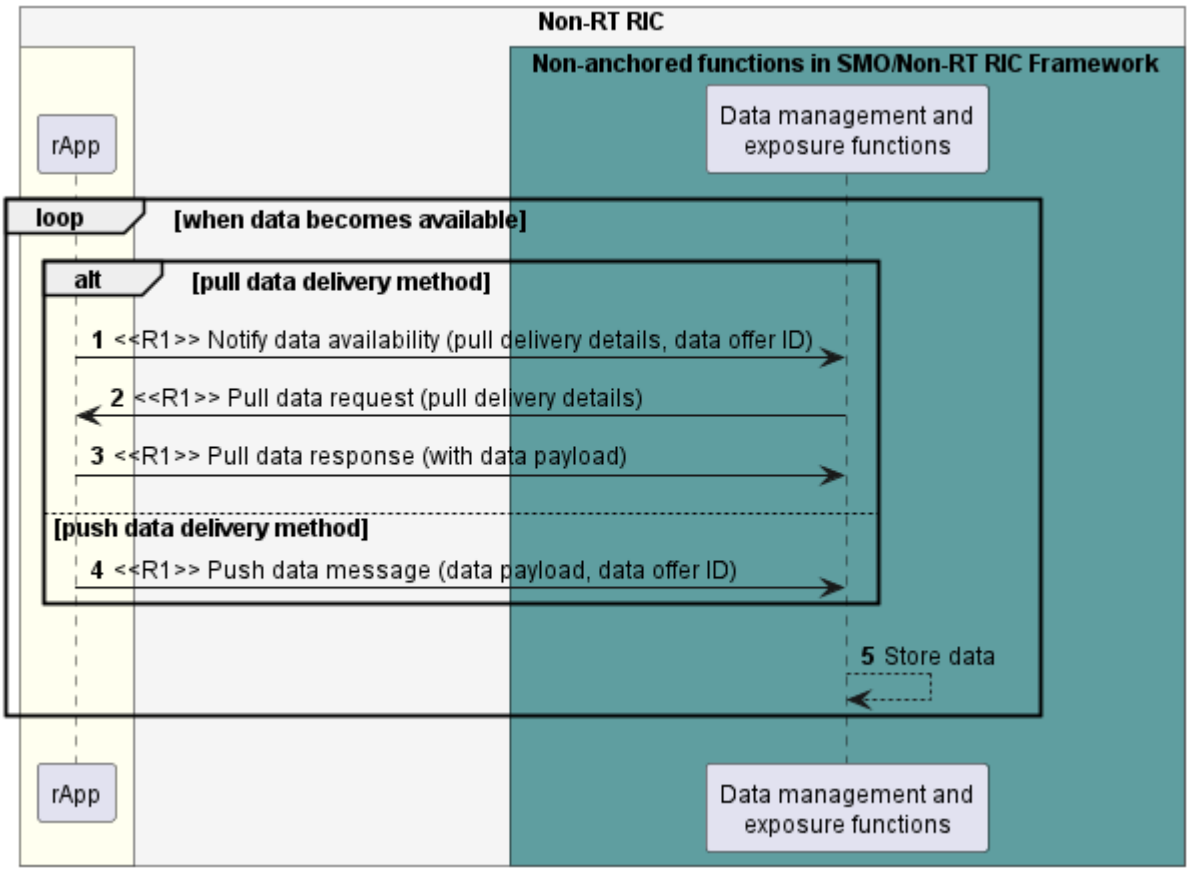

Figure 7.3.4.2-1: Deliver offered data use case flow diagram

7.3.4.3 Terminate data offer by Data Producer

Table 7.3.4.3-1: Terminate data offer by Data Producer use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The Data Producer rApp terminates a data offer to indicate it stops producing a data instance for collection.	
Actors and Roles	- rApp in the role of Data Producer that requests data offer termination. - Data management and exposure functions in the role of Data management and exposure services Producer that handles the data offer termination.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the Data management and exposure functions. - The rApp has created a data offer.	
Begins when	The rApp determines the need to terminate a data offer it has created previously.	
Step 1 (M)	The rApp requests the Data management and exposure functions to terminate a data offer by providing the rAppId and the data offer identifier.	
Step 2 (M)	The Data management and exposure functions check whether the rApp is authorized to terminate the data offer.	
Step 3 (M)	The Data management and exposure functions validate the information provided with the request.	
Step 4 (M)	The Data management and exposure functions stop collecting the data instance and terminate the data offer.	
Step 5 (M)	The Data management and exposure functions respond to the rApp with a success result along with the data offer identifier.	
Ends when	The rApp was able to terminate the data offer.	
Exceptions	n/a	
Post Conditions	The data instance associated with the data offer is no longer collected.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 70
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "rApp" as rapp
  endbox
    box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
      participant "Data management and\n exposure functions" as dme
        endbox
endbox
rapp -> dme: <<R1>> Terminate data offer request (rAppId, dataOfferId)
activate dme
dme -->dme: AuthZ
note right
     Check authorization in collaboration
     with SME functions
```



```
end note
dme --> dme: Validate request
dme --> dme: Stop collecting data\nand terminate data offer
dme -> rapp: <<R1>> Terminate data offer response (dataOfferId)
deactivate dme
@enduml
```

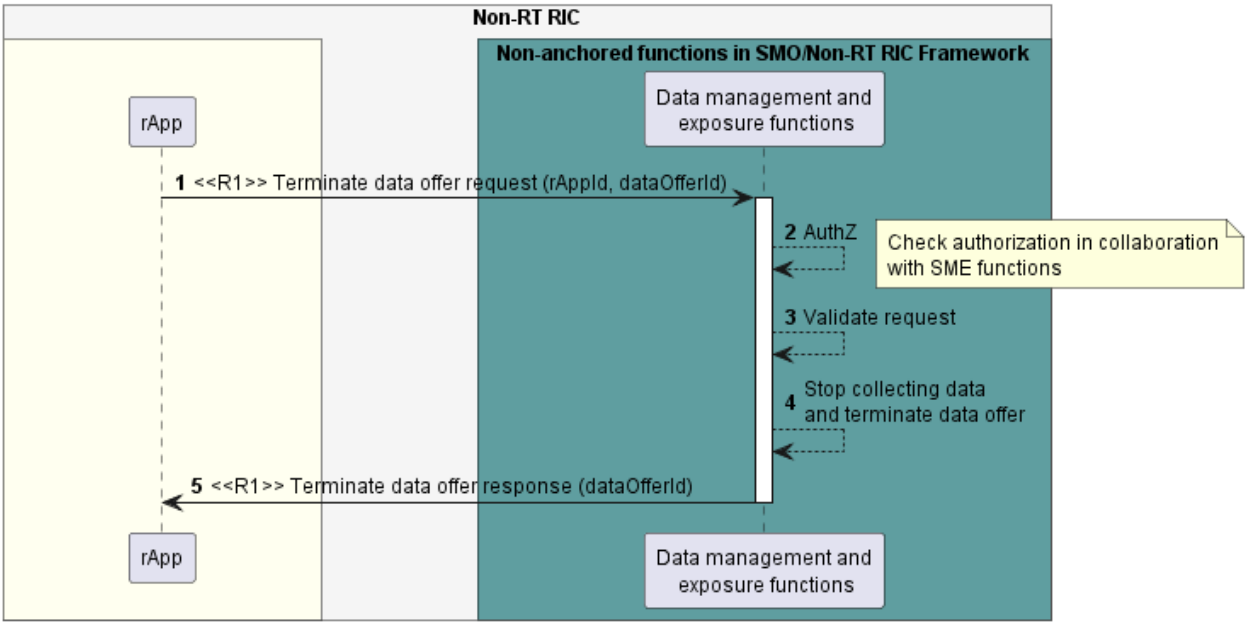

Figure 7.3.4.3-1: Terminate data offer by Data Producer use case flow diagram

7.3.4.4 Terminate data offer by DME services Producer

Table 7.3.4.4-1: Terminate data offer by DME services Producer use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The Data management and exposure functions send a data offer termination notification to indicate they do not intend to collect the data instance from the Data Producer any longer.	
Actors and Roles	- rApp in the role of Data Producer that receives a data offer termination notification. - Data management and exposure functions in the role of Data management and exposure services Producer that sends the data offer termination notification.	
Assumptions	n/a	
Preconditions	The rApp has created a data offer.	
Begins when	The Data management and exposure functions determine a need to terminate a data offer.	
Step 1 (M)	The Data management and exposure functions send to the rApp a data offer termination notification providing the data offer ID.	
Step 2 (M)	After a delay to allow the Data Producer to react to the notification, the Data management and exposure functions stop collecting data for the affected data offer and terminate the offer.	
Step 3 (M)	The rApp stops delivering to the Data management and exposure functions data related to the affected data offer.	
Ends when	The Data management and exposure functions were able to terminate the data offer.	
Exceptions	n/a	
Post Conditions	The data instance associated with the data offer is no longer collected.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 70
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "rApp" as rapp
  endbox
box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
     participant "Data managenment and\n exposure functions" as dme
  endbox
endbox
dme -> rapp: <<R1>> Terminate data offer notification (dataOfferId)
dme --> dme: Stop collecting data and\nterminate data offer
rapp --> rapp: Stop delivering data
@enduml
```


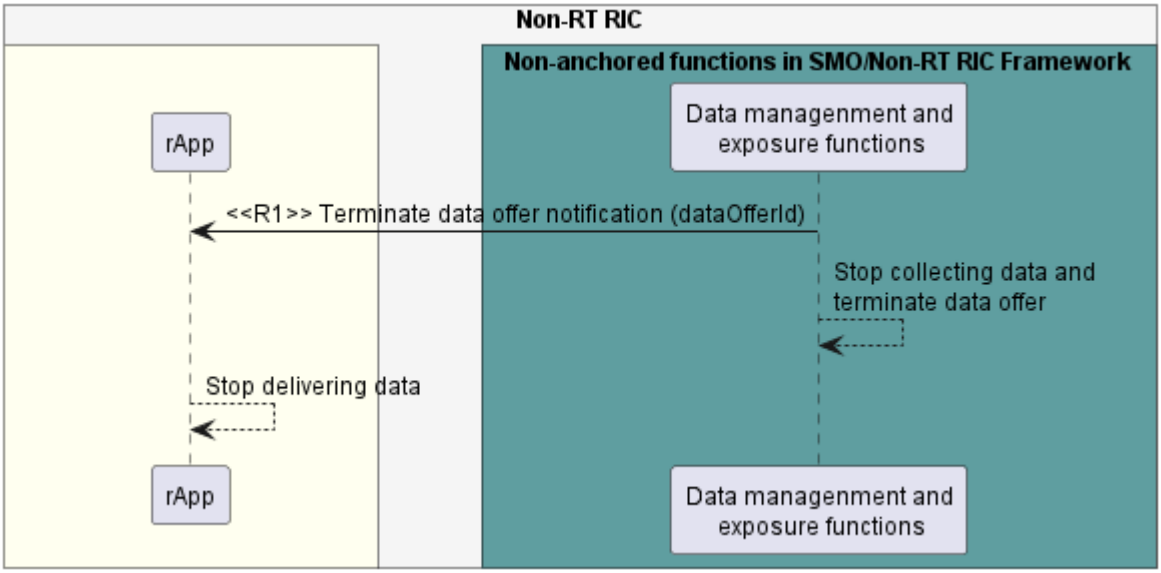

Figure 7.3.4.4-1: Terminate data offer by DME services Producer use case flow diagram

7.3.5 Required data

The Create data offer request contains the rAppId, the registered data type ID for the data to be produced and additional information related to the data instance to be produced such as scope, target, context, timing etc. Also, the request contains the supported delivery method(s) for the data and information on the endpoint where to send data offer termination notifications.

The Create data offer response contains the data offer ID. In case of the pull delivery method, the response further contains information about the endpoint where to send data availability notifications. In case of the push delivery method, the response further contains information about the endpoint where to push the data.

The Notify data availability message contains pull delivery details (e.g., a URI) and the data offer identifier.

In the Pull data request, the Data management and exposure functions use the pull delivery details provided in the notify data availability message to locate the data.

The Pull data response contains a set of data for collection.

The Push data message contains a set of data for collection and the data offer identifier.

The Terminate data offer request contains the rAppId and the data offer ID.

The Terminate data offer response contains the data offer ID.

The Data offer termination notification contains the data offer ID.

7.4 DME use case 4: Data request - Data Consumer rApp requests data for consumption from the DME functions

7.4.1 Overview

This use case describes how a Data Consumer rApp requests to obtain one-off data for consumption from the DME functions.

7.4.2 Background and goal of the use case

The Data request and subscription service and Data delivery service procedures are defined as part of the Data management and exposure services in R1GAP[[1]].

After performing the Data discovery procedure, an rApp will be able to request data from the Data management and exposure functions, and the Data management and exposure functions delivers the requested data.

7.4.3 Entities/resources involved in the use case

1) Data management and exposure functions
    a. supports functionality to receive a data request from an rApp,
    b. supports functionality to authorise the rApps request for data,
    c. supports functionality to provide data to an rApp.
2) rApp
    a. support functionality to request data,
    b. support functionality to retrieve the data.

7.4.4 Solutions

7.4.4.1 Data Consumer rApp request data

Table 7.4.4.1-1: Use case: A Data Consumer rApp requests data for consumption from the DME functions

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	An rApp obtains the data they intends to consume from the Data Management and Exposure functions via the request data procedure.	
Actors and Roles	- Data Consumer rApp in the role of Data management and exposure services Consumer. - Data management and exposure functions in the role of Data management and exposure services Producer that provides data.	
Assumptions	n/a	
Preconditions	- rApp is deployed, authenticated, and authorized for usage of the Data management and exposure services. - Data management and exposure functions produce the Data management and exposure services.	
Begins when	An rApp, as a Data Consumer, initiates the request data procedure.	
Step 1 (M)	The Data Consumer rApp sends to the Data management and exposure functions a request data request to obtain data for consumption, providing its rAppId and data instance information.	
Step 2 (M)	The Data management and exposure functions check the Data Consumer rApps authorisation.	
Step 3 (M)	The Data management and exposure functions validate the Data Consumer's data request.	
Step 4 (M)	The Data management and exposure functions send to the Data Consumer rApp the data request response and include pull delivery details for the requested data instance.	
Step 5 (M)	The Data Consumer rApp sends to the Data management and exposure functions a pull data request message, using the pull delivery details received in step 4.	
Step 6 (M)	The Data management and exposure functions respond with a pull data response, including in the payload the data to be consumed.	
Ends when	The requested data have been delivered to the Data Consumer rApp for consumption.	
Exceptions	n/a	
Post Conditions	The Data Consumer rApp has consumed the requested data.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
  box #ivory
participant "rApp" as rApp
end box
box "Non-RT RIC" #whitesmoke
box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and \n exposure functions" as dme
end box
dme <- rApp : <<R1>> Request data request (rAppId, Data instance information)
```



```
dme --> dme :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
dme --> dme: Validate
rApp <- dme : <<R1>> Request data response (pull delivery details)
dme <- rApp : <<R1>> Pull data request(pull delivery details)
rApp <- dme : <<R1>> Pull data response (with data payload)
@enduml
```

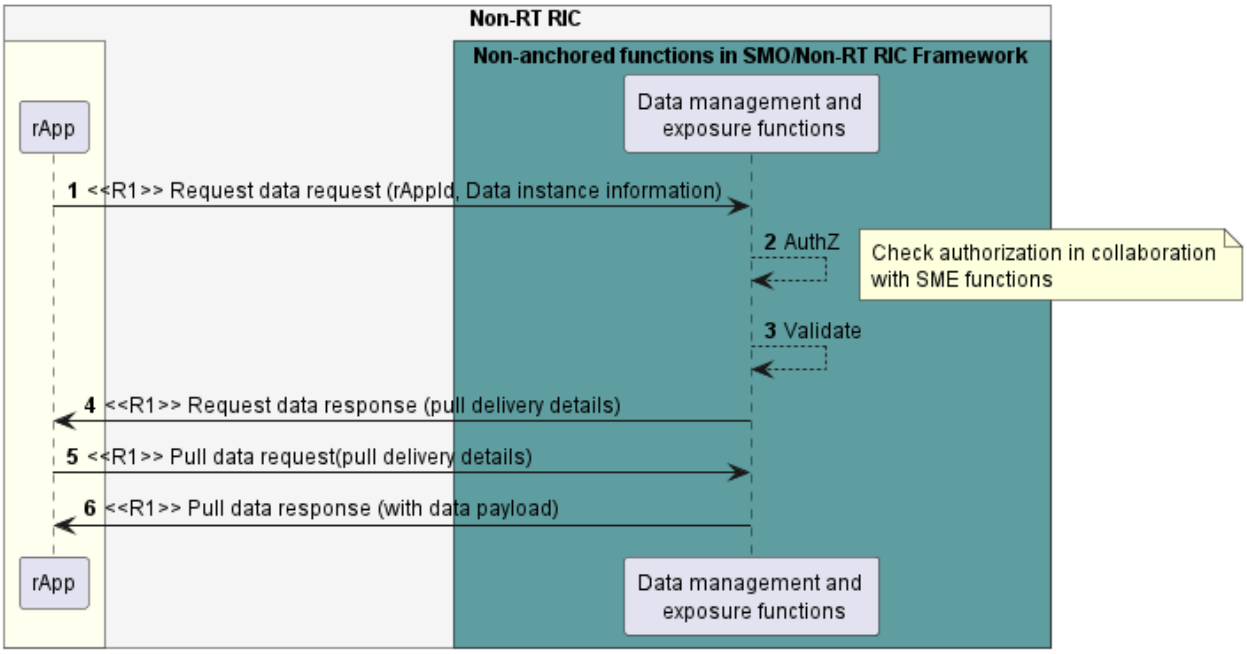

Figure 7.4.4.1-1: Procedure for use case: A Data Consumer rApp requests data for consumption from the DME functions use case flow diagram

7.4.5 Required data

For Request data request, the rApp provides to the Data management and exposure functions its rApp identifier (rAppId) and data instance information. Data instance information includes data type identifier, scope (i.e., filter on the data, and optional constraints, e.g., time interval (i.e., start time and end time), etc..

For Request data response, the Data management and exposure functions provide pull delivery details. The pull delivery details are used to locate the requested data instance, e.g., using a URI.

NOTE: A request identifier, assigned by the service producer, can be a part of pull delivery details if necessary, during the protocol design stage.

For Pull data request, the rApp provides pull delivery details.

For Pull data response, the Data management and exposure functions provide the required data.

7.5 DME use case 5: Data request – DME functions request data for collection from a Data Producer rApp

7.5.1 Overview

This use case describes how the Data management and exposure functions request to obtain one-off data for consumption from a Data Producer rApp.

7.5.2 Background and goal of the use case

The Data request and subscription service and Data delivery service procedures are defined as part of Data management and exposure services in R1GAP[[1]].

The Data management and exposure functions request data from a Data Producer rApp, and the Data Producer rApp delivers the requested data.

7.5.3 Entities/resources involved in the use case

1) Data management and exposure functions
    a. support functionality to request data,
    b. support functionality to retrieve the data.
2) rApp
    a. supports functionality to provide data to the DME

7.5.4 Solutions

7.5.4.1 DME functions request data for collection from a Data Producer rApp

Table 7.5.4.1-1: Use case: DME functions request data for collection from a Data Producer rApp

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The DME functions obtain the data they intends to collect from the Data Producer rApp via request data procedure	
Actors and Roles	- rApp in the role of Data Producer - Data management and exposure functions in the role of Data management and exposure services Consumer	
Assumptions	n/a	
Pre conditions	- rApp is deployed and authenticated. - Data management and exposure functions have the data type identifier of the needed data.	
Begins when	The Data management and exposure functions, initiate the request data procedure to collect data from the Data Producer rApp.	
Step 1 (M)	The Data management and exposure functions send the Data Producer rApp a request data request to obtain data for collection, providing data instance information.	
Step 2 (M)	The Data Producer rApp validates the Data management and exposure functions data request.	
Step 3 (M)	The Data Producer rApp replies with a request data response message, including details how to pull the requested data.	
Step 4 (M)	The Data management and exposure functions send the Data Producer rApp a pull data request message, using the pull delivery details received in step 3.	
Step 5 (M)	The Data Producer rApp responds with a pull data response, including in the payload the data to be collected.	
Ends when	The requested data have been delivered to the Data management and exposure functions for collection.	
Exceptions	n/a	
Post Conditions	The Data management and exposure functions have collected the requested data.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
box #ivory
participant "rApp" as rApp
end box
box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and\n exposure functions" as dme
end box

dme -> rApp : <<R1>> Request data request (Data instance information)
rApp --> rApp: validate
rApp -> dme : <<R1>> Request data response (pull delivery details)
dme -> rApp : <<R1>> Pull data request (pull delivery details)
rApp -> dme : <<R1>> Pull data response (with data payload)
@enduml
```


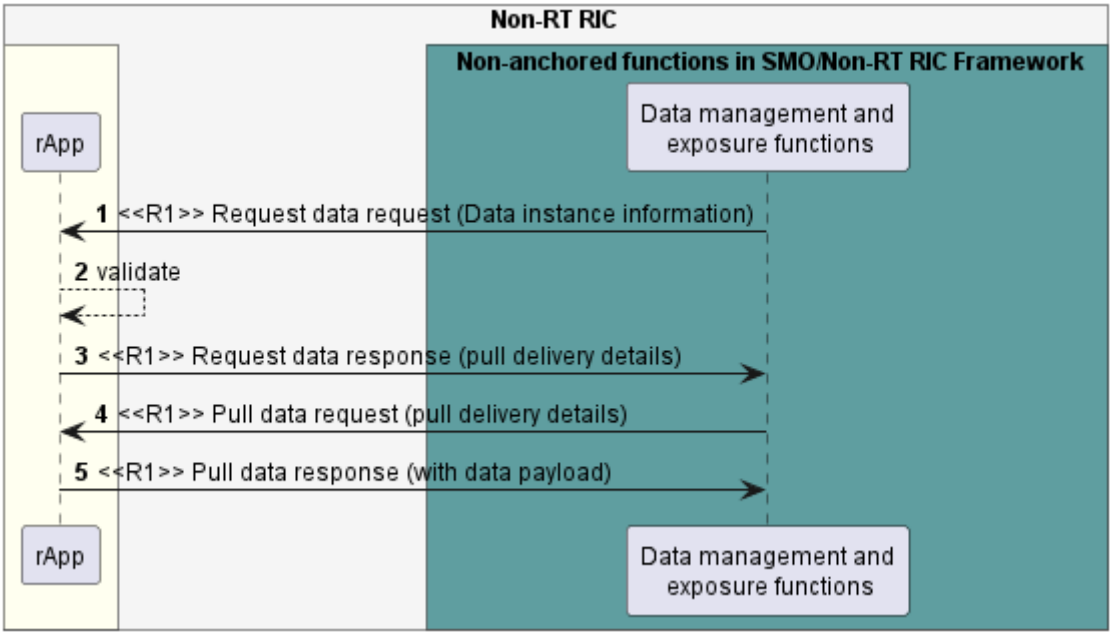

Figure 7.5.4.1-1: Procedure for use case: DME functions request data for collection from a Data Producer rApp use case flow diagram

7.5.5 Required data

For the Request data request, the Data management and exposure functions provide data instance information. Data instance information includes data type identifier, scope (i.e., filter on the data), and optional constraints, e.g., time interval (i.e., start time and end time), etc.

For Request data response, the Data Producer rApp provides pull delivery details. The pull delivery details are used to locate the requested data instance, e.g., using a URI.

NOTE: A request identifier, assigned by the service producer, can be a part of pull delivery details if necessary, during the protocol design stage.

For Pull data request, the Data management and exposure functions use the pull delivery details provided in the Request data response to locate the data.

For Pull data response, the rApp provides the requested data.

7.6 DME use case 6: The DME functions subscribe data for collection from a Data Producer rApp

7.6.1 Overview

This use case describes how the DME functions subscribe data for collection from a Data Producer rApp, obtain the data and terminate the data subscription.

7.6.2 Background and goal of the use case

The Data request and subscription service and Data delivery service procedures are defined as part of Data management and exposure services in R1GAP[[1]].

The Data management and exposure functions subscribe data from a Data Producer rApp for collection and terminate the data subscription.

The Data Producer rApp delivers the requested data.

7.6.3 Entities/resources involved in the use case

1) Data management and exposure functions
    a. support functionality to request subscription to data and termination of the subscription,
    b. support functionality to obtain the data for collection.
2) Data Producer rApp
    a. supports functionality to subscribe to data,
    b. supports functionality to terminate a data subscription,
    c. support functionality to provide data to the DME functions.

7.6.4 Solutions

7.6.4.1 DME functions subscribe data for collection from a Data producer rApp

Table 7.6.4.1-1: Use case: The DME functions subscribe data for collection from a Data Producer rApp

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The Data management and exposure functions obtain data they intend to collect from a Data Producer rApp via the subscribe data procedure.	
Actors and Roles	- rApp in the role of Data Producer. - Data management and exposure functions: entity in the role of Data request and subscription service Consumer and Pull data service Consumer and/or Push data service Producer.	
Assumptions	n/a	
Preconditions	- rApp is deployed, and authenticated - Data management and exposure functions have the data type identifier of the needed data.	
Begins when	The Data management and exposure functions initiate the Subscribe data procedure to collect a data instance.	
Step 1 (M)	The Data management and exposure functions send to the Data Producer rApp a subscribe data request to subscribe the data to be collected providing Data instance information.	
Step 2 (M)	The Data Producer rApp validates the request.	
Step 3 (M)	The Data Producer rApp creates the subscription.	
Step 4 (M)	The Data Producer rApp replies with a subscribe data response message, including the subscription identifier.	
Alternative procedure	Each time a set of data becomes available for delivery, Steps 5-7 are executed if pull data is used as the data delivery method.	
Step 5 (M)	The Data Producer rApp sends to the Data management and exposure functions a Notify data availability message, including details how to pull the requested data and the subscription identifier.	
Step 6 (M)	The Data management and exposure functions send a pull data request message, using the pull delivery details obtained in step 5.	
Step 7 (M)	The Data Producer rApp responds with a pull data response, including the data to be collected.	
Alternative procedure	Each time a set of data becomes available for delivery, Step 8 is executed if push data is used as the data delivery method.	
Step 8 (M)	The Data Producer rApp sends to the Data management and exposure functions a push data message, including the subscription identifier and the data to be collected.	
Ends when	The subscribed data instance has been delivered for collection to the Data management and exposure functions.	
Exceptions	n/a	
Post Conditions	The Data management and exposure functions have collected the needed data.	
Traceability		

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
box #ivory
participant "rApp" as rApp
end box
```



```
box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and\n exposure functions" as dme
end box
dme -> rApp : <<R1>> subscribe data request (data instance information)
activate dme
rApp --> rApp : Validate request
rApp --> rApp : Create subscription
rApp -> dme : <<R1>> subscribe data response (subscription ID)
deactivate dme

loop when data becomes available
alt pull data delivery method
rApp -> dme : <<R1>> Notify data availability (pull delivery details, subscription ID)
dme -> rApp : <<R1>> Pull data request
rApp -> dme : <<R1>> Pull data response (with data payload)
else push data delivery method
rApp -> dme : <<R1>> Push data message (subscription ID, data payload)
end
end
@enduml
```

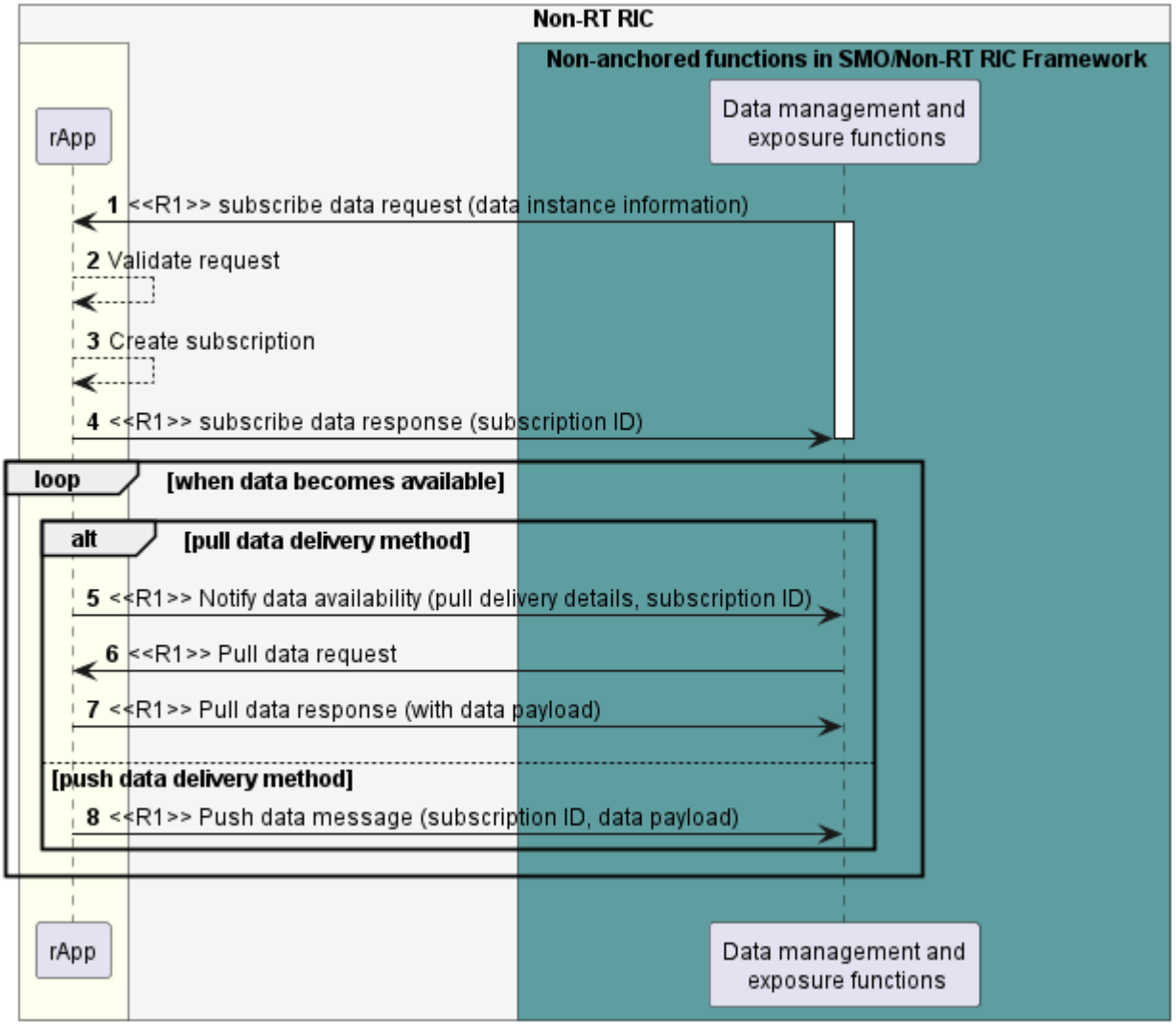

Figure 7.6.4.1-1: Procedure for use case: DME functions subscribe data for collection from a Data Producer rApp use case flow diagram

7.6.4.2 DME functions terminate the data subscription

Table 7.6.4.2-1: Use case: The DME functions terminate the data subscription.

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	Data management and exposure functions terminate the data subscription.	
Actors and Roles	- rApp in the role of Data Producer - Data management and exposure functions in the role of Data request and subscription service Consumer	
Assumptions	n/a	
Preconditions	The DME functions have subscribed data.	
Begins when	The DME functions determine to terminate the data subscription.	
Step 1 (M)	The DME functions send the Unsubscribe data request with subscription ID as parameter.	
Step 2 (M)	The rApp validates the request.	
Step 3 (M)	The rApp removes the data subscription.	
Step 4 (M)	The rApp responds to the request.	
Ends when	The DME functions have terminated the data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
box #ivory
participant rApp as rApp
end box
box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and\n exposure functions" as DME
end box

rApp <- DME: <<R1>> Unsubscribe data request\n (subscription ID)
activate rApp
rApp --> rApp : validate request
rApp --> rApp : remove subscription
rApp -> DME: <<R1>> Unsubscribe data response
deactivate rApp
@enduml
```


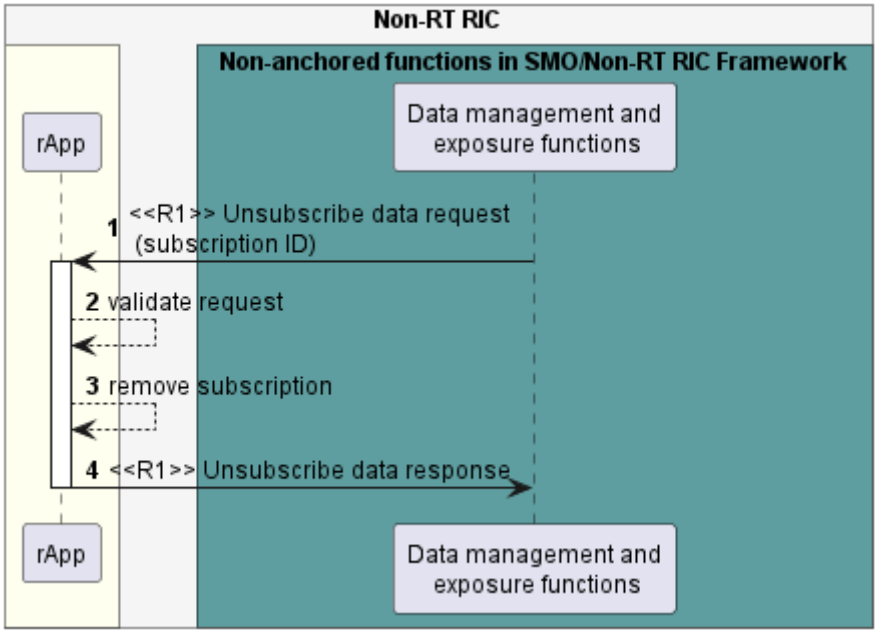

Figure 7.6.4.2-1: Procedure for use case: The DME functions terminate the data subscription

7.6.5 Required data

In the subscribe data request, the Data management and exposure functions provide data instance information. Data instance information includes the data type identifier, scope (i.e., filter on the data), periodicity of collection and additional optional constraints periodicity of reporting, etc.

In the subscribe data response, the Data Producer rApp provides the subscription identifier. The subscription identifier is a unique identifier assigned by service Producer to identify a specific subscription.

In the unsubscribe data request, the Data management and exposure functions provide the subscription identifier.

In the notify data availability message, the Data Producer rApp provides pull delivery details (e.g., a URI) and the subscription identifier.

In the pull data request, the Data management and exposure functions use the pull delivery details provided in the notify data availability message to locate the data.

In the pull data response, the Data Producer rApp provides a set of data for collection.

In the push data message, the Data Producer rApp provides the subscription identifier and a set of data for collection.

7.7 DME use case 7: A Data Consumer rApp subscribes data for consumption using the Data request and subscription service

7.7.1 Overview

This use case describes how a Data Consumer rApp subscribes data it needs to consume, obtains the data and terminates the data subscription.

7.7.2 Background and goal of the use case

The Data request and subscription service and Data delivery service procedures are defined as part of Data management and exposure services in R1GAP[[1]].

A Data Consumer rApp will be able to subscribe data based on the data type information it is aware of e.g., from discovery or configuration and terminate the data subscription using Data management and exposure procedures.

Data management and exposure functions deliver the subscribed data using Data delivery services.

7.7.3 Entities/resources involved in the use case

1) Data management and exposure functions
    a. support services that allow an rApp to subscribe data and terminate the data subscription,
    b. deliver the subscribed data to an rApp that has subscribed.
2) rApp
    a. supports functionality to initiate a data subscription,
    b. supports functionality to initiate the termination of a data subscription,
    c. supports functionality to obtain the subscribed data.

7.7.4 Solutions

7.7.4.1 Subscribe data and delivery of subscribed data

Table 7.7.4.1-1: Subscribing data and delivery of subscribed data

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	rApp subscribes the data it needs to consume, and the subscribed data is delivered.	
Actors and Roles	- rApp in the role of Data management and exposure services Consumer. - Data management and exposure functions in the role of Data management and exposure services Producer.	
Assumptions	n/a	
Preconditions	1) The rApp is authorized to access the Data management and exposure services, 2) The rApp has obtained the data type information e.g., by discovery or configuration.	
Begins when	The rApp determines the need to subscribe data it needs to consume.	
Step 1 (M)	The rApp subscribes data that it needs to consume with rAppId and Data instance information as parameters.	
Step 2 (M)	The Data management and exposure functions check for authorization with SME functions whether the rApp is authorized to subscribe data.	
Step 3 (M)	The Data management and exposure functions validate the request.	
Step 4 (M)	The Data management and exposure functions create the subscription.	
Step 5 (M)	The Data management and exposure functions respond to the request with subscription ID as a parameter.	
Alternative procedure	Each time a set of subscribed data becomes available for delivery, Steps 5-7 are executed if “pull data” is used as the data delivery method.	
Step 6 (M)	The Data management and exposure functions notify the availability of data with subscription ID and pull delivery details as parameters.	
Step 7 (M)	The rApp send a pull data request based on the pull delivery details received in the previous step.	
Step 8 (M)	The Data management and exposure functions respond to the pull data request with data payload as a parameter.	
Alternative procedure	Each time a set of subscribed data becomes available for delivery, Steps 9 is executed if the “push data” is used as the data delivery method.	
Step 9 (M)	The Data management and exposure functions send a push data message with subscription ID and data payload as parameters.	
Ends when	The rApp has terminated the subscription.	5.7.4.2
Exceptions	n/a	
Post Conditions	The rApp has the data it determined it needed.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
box #ivory
participant rApp as rApp
end box
box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and\n exposure functions" as DME
end box
```



```
rApp -> DME : <<R1>> Subscribe data request (rAppId, Data instance information)
DME --> DME : AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME : Validate request
DME --> DME : Create subscription
rApp <- DME : <<R1>> Subscribe data response(subscription ID)

loop when data becomes available
alt pull data delivery method
rApp <- DME : <<R1>> Notify data availability (pull delivery details, subscription ID )
rApp -> DME : <<R1>> Pull data request
rApp <- DME : <<R1>> Pull data response (data payload)
else push data delivery method
DME -> rApp : <<R1>> Push data message (subscription ID, data payload)
end
end

@enduml
```

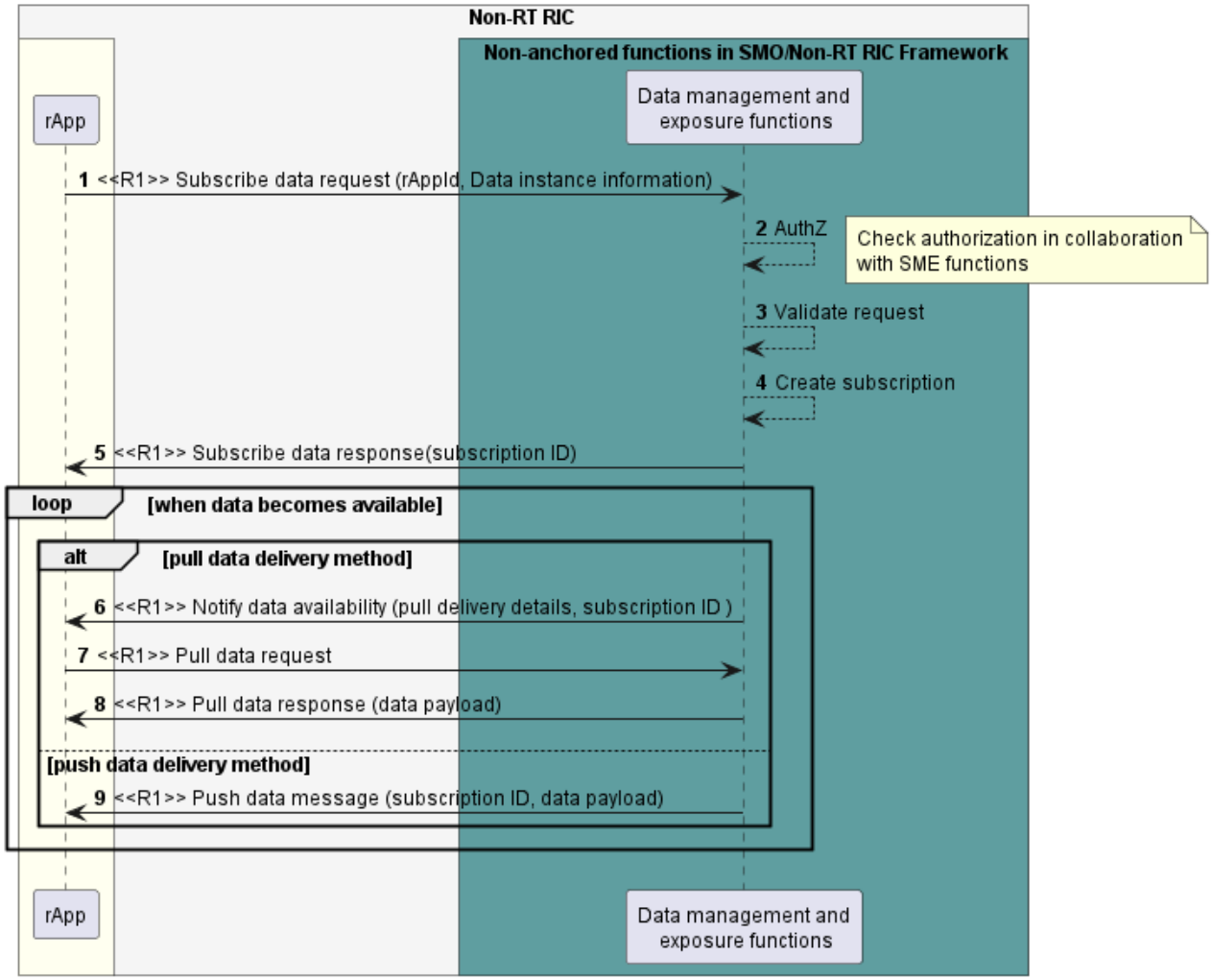

Figure 7.7.4.1-1: Subscribe data and delivery of subscribed data use case flow diagram

7.7.4.2 Terminate data subscription

Table 7.7.4.2-1: Terminate data subscription

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	rApp terminates the data subscription.	
Actors and Roles	- rApp in the role of Data management and exposure services Consumer - Data management and exposure functions in the role of Data management and exposure services Producer.	
Assumptions	n/a	
Preconditions	The rApp has created a data subscription.	
Begins when	The rApp determines to terminate the data subscription.	
Step 1 (M)	The rApp sends the Unsubscribe data request with rAppId and subscription ID as parameters.	
Step 2 (M)	The Data management and exposure functions check for authorization with SME functions whether the rApp is authorized to terminate subscriptions.	
Step 3 (M)	The Data management and exposure functions validate the request.	
Step 4(M)	The Data management and exposure functions remove the subscription.	
Step 5 (M)	The Data management and exposure functions respond to the request.	
Ends when	rApp has terminated the data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 5
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid
autonumber
box "Non-RT RIC" #whitesmoke
box #ivory
participant rApp as rApp
end box
box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "Data management and\n exposure functions" as DME
end box

rApp -> DME: <<R1>> Unsubscribe data request\n (rAppId, subscription ID)
activate DME
DME --> DME : AuthZ
note right
Check authorization in collaboration
with SME functions
end note

DME --> DME : Validate request
DME --> DME : Remove subscription
rApp <- DME: <<R1>> Unsubscribe data response
deactivate DME
@enduml
```


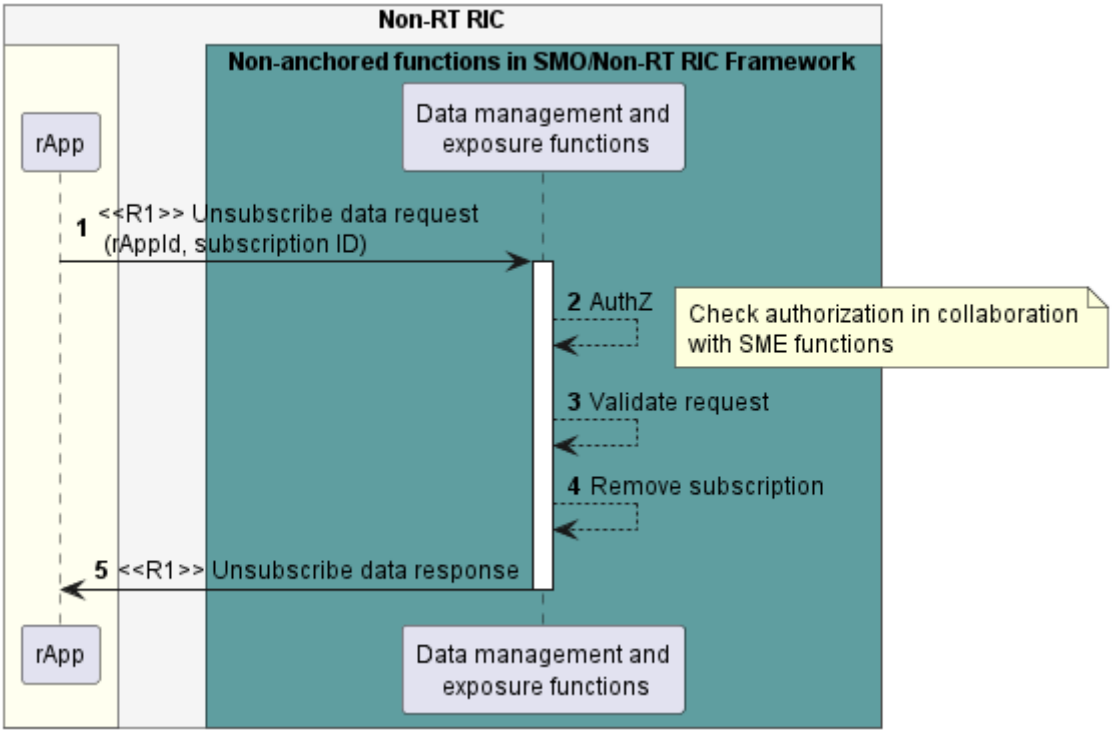

Figure 7.7.4.2-1: Terminate data subscription use case flow diagram

7.7.5 Required data

The rApp defines the data it needs to consume in data instance information parameter. Data instance information includes data type identifier, scope (i.e., filter on the data), periodicity of collection and additional optional constraints e.g., periodicity of reporting., etc.

A subscription ID is a unique identifier per subscription assigned by the Data management and exposure functions. The rApp requests the Data management and exposure functions with rAppId and subscription identifier as parameters to terminate the data subscription.

In the notify data availability message, the Data management and exposure functions provide pull delivery details (e.g., a URI) and the subscription identifier.

In the pull data request, the rApp uses the pull delivery details provided in the notify data availability message to locate the data.

In the pull data response, the Data management and exposure functions provide a set of data for consumption.

In the push data message, the Data management and exposure functions provide the subscription identifier and a set of data for consumption.

8 Use cases for RAN OAM-Related Services

8.1 RAN OAM-Related use case 1: Alarm query

8.1.1 Overview

This use case allows an rApp acting as Service Consumer to query alarm information.

8.1.2 Background and goal of the use case

An rApp acting as Service Consumer can query from the Fault management service Producer information about an individual alarm, a set of alarms matching provided filtering criteria or all active alarms available in the alarm list.

8.1.3 Entities/resources involved in the use case

1) RAN OAM-related functions as Fault management service Producer
    a. receive the request to query alarm information,
    b. provide the response of success or failure result to the Query alarm information request.
2) rApp
    a. supports functionality to initiate the procedure to query alarm information.

8.1.4 Solutions

8.1.4.1 Query alarm information

Table 8.1.4.1-1: Query alarm information use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp obtains alarm information from the Fault management service Producer.	
Actors and Roles	- rApp in the role of Service Consumer that queries alarm information. - RAN AOM-related functions in the role of Fault management service Producer that provides alarm information in response to the alarm queries.	
Assumptions	n/a	
Preconditions	The rApp is deployed and authorized to query alarm information.	
Begins when	The rApp determines the need to query alarm information.	
Step 1 (M)	The rApp queries alarm information from the RAN OAM-related functions by providing the rAppId and optional query information that determines the requested result set.	
Step 2 (M)	The RAN OAM-related functions check whether the rApp is authorized to query alarm information.	
Step 3 (M)	The RAN OAM-related functions validate the query information, if it has been provided with the request.	
Step 4 (M)	The Ran OAM-related functions respond to the rApp with a success result along with the requested alarm information.	
Ends when	The rApp was able to query alarm information.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 70
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid

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

  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
      participant "RAN OAM-related functions" as fmsp
  endbox
endbox
rapp -> fmsp: <<R1>> Query alarm information request (rAppId, queryInfo)
fmsp --> fmsp: AuthZ
note right
Check authorization in collaboration
with SME functions
end note
fmsp --> fmsp: Validate request
fmsp -> rapp: <<R1>> Query alarm information response (alarmInfo)
```


@enduml

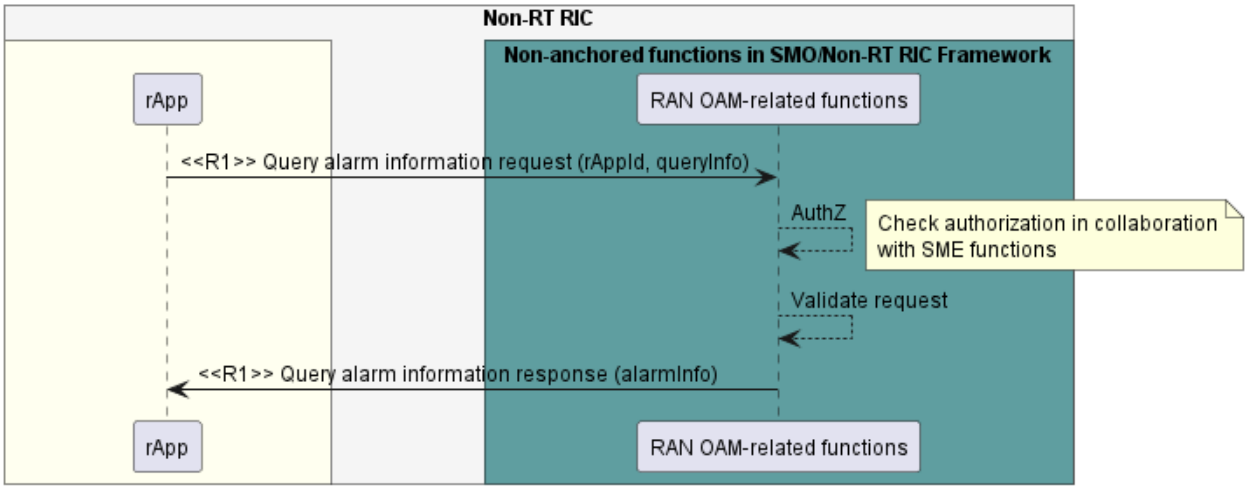

Figure 8.1.4.1-1: Query alarm information use case flow diagram

8.1.5 Required data

The Query alarm information request contains the rAppId and optional query information. The query information determines the requested alarm information, for instance a single alarm with a particular alarm ID, all alarms in the alarm list or a subset of these alarms that match a set of filtering parameters.

The Query alarm information response contains the requested alarm information scoped by the query information provided in the request.

8.2 RAN OAM-Related use case 2: Alarm acknowledgement / unacknowledgement

8.2.1 Overview

This use case allows an rApp acting as Service Consumer to acknowledge and unacknowledge an alarm with the Fault management service Producer.

8.2.2 Background and goal of the use case

An rApp acting as Service Consumer can acknowledge and unacknowledge an individual alarm managed by the Fault management service Producer.

8.2.3 Entities/resources involved in the use case

1) RAN OAM-related functions in the role of Fault management service Producer
    a. receives the request to change the acknowledgement state of an alarm,
    b. store the alarm acknowledgement state information,
    c. provides the response of success or failure result to the Change alarm acknowledgement state request.
2) rApp

a. supports functionality to initiate the procedure to change the acknowledgement state of an alarm.

8.2.4 Solutions

8.2.4.1 Change alarm acknowledgement state

Table 8.2.4.1-1: Change alarm acknowledgement state use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp requests the Service Producer to acknowledge or unacknowledge an alarm.	
Actors and Roles	- rApp in the role of Service Consumer that requests to acknowledge or unacknowledge an alarm. - RAN OAM-related functions in the role of Fault management service Producer that handles the request to acknowledge or unacknowledge an alarm.	
Assumptions	The rApp is aware of the alarm ID of the alarm to be acknowledged or unacknowledged.	
Preconditions	The rApp is deployed and is authorized to consume the Fault management service.	
Begins when	The rApp determines the need to acknowledge or unacknowledge an alarm.	
Step 1 (M)	The rApp requests the RAN OAM-related functions to change the acknowledgement state of an alarm by providing the rAppId, the alarm ID and the new acknowledgement state (i.e., acknowledged or unacknowledged).	
Step 2 (M)	The RAN OAM-related functions check whether the rApp is authorized to change the acknowledgement state of alarms.	
Step 3 (M)	The RAN OAM-related functions validate the alarm ID that has been provided with the request.	
Step 4 (M)	The RAN OAM-related functions persist the new state of the alarm identified by the alarm ID to represent the alarm acknowledgement state change.	
Step 5 (M)	The RAN OAM-related functions respond to the rApp with a success result.	
Ends when	The rApp was able to acknowledge or unacknowledge an alarm.	
Exceptions	n/a	
Post Conditions	The state of the alarm identified by the alarm ID has been changed according to the request.	
Traceability	n/a	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 70
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid

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

  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
      participant "RAN OAM-related functions" as fmsp
```



```
  endbox
endbox
rapp -> fmsp: <<R1>> Change alarm acknowledgement state request\n(rAppId,alarmId,
newAlarmState)
fmsp --> fmsp: AuthZ
note right
Check authorization in collaboration
with SME functions
end note

fmsp --> fmsp: Validate request
fmsp --> fmsp: Persist new alarm state
fmsp -> rapp: <<R1>> Change alarm acknowledgement state response
@enduml
```

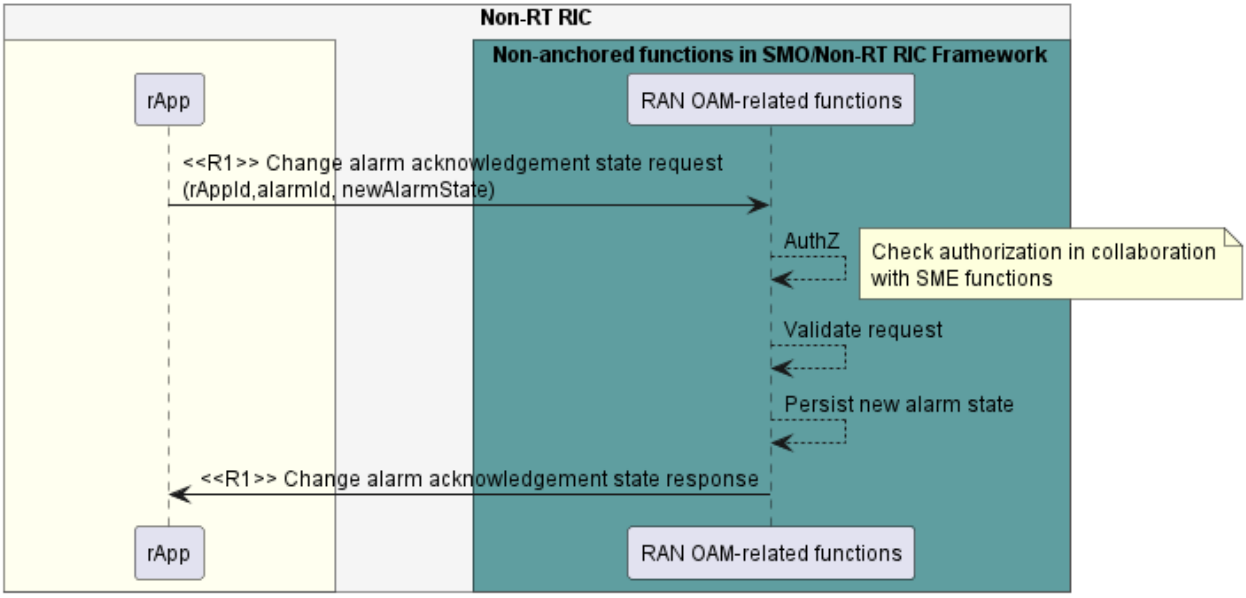

Figure 8.2.4.1-1: Change alarm acknowledgement state use case flow diagram

8.2.5 Required data

The Change alarm acknowledgement state request contains the rAppId, the alarm ID of the alarm to be acknowledged or unacknowledged and the requested new acknowledgement state of the alarm.

NOTE: It is FFS if and how the RAN OAM-related functions ensure that the alarm IDs from multiple network elements do not collide.

8.3 RAN OAM-Related use case 3: Performance information query

8.3.1 Overview

This use-case allows an rApp acting as a Service Consumer to query the performance information pertaining to the managed entities.

8.3.2 Background and goal of the use-case

An rApp acting as a Service Consumer can query the performance information pertaining to one or more managed entities from the Performance management Service Producer.

8.2.3 Entities/resources involved in the use-case

1) RAN OAM-related functions as Performance Management Service Producer

a. receives the request to query the performance information related to one or more managed entities

b. provides the response of success or failure result to the performance information query request

2) rApp

a. supports functionality to initiate the procedure to query the performance information

8.3.4 Solutions

8.3.4.1 Query performance information

Table 8.3.4.1-1: Query performance information use-case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp obtains performance information from the Performance management service Producer.	
Actors and Roles	- rApp in the role of Service Consumer that queries performance information pertaining to one or more managed entities. - RAN OAM-related functions in the role of Performance Management service Producer that provide the requested performance information.	
Assumptions	n/a	
Preconditions	The rApp is deployed and authorized to query performance information.	
Begins when	The rApp determines the need to query performance information.	
Step 1 (M)	The rApp queries performance information from the RAN OAM-related functions by providing the rAppId, optional query criteria and information about the managed entities, along with the desired performance information, that determines the requested result set.	
Step 2 (M)	The RAN OAM-related functions check whether the rApp is authorized to query the performance information.	
Step 3 (M)	The RAN OAM-related functions validate the performance information criteria if they were provided with the request.	
Step 4 (M)	The RAN OAM-related functions respond to the rApp with a success result along with the desired performance management information.	
Ends when	The rApp was able to receive the performance management response.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-OAM-PMservice-FUN1	

```
@startuml
!pragma teoz true
skinparam ParticipantPadding 70
skinparam BoxPadding 10
skinparam defaultFontSize 12
skinparam lifelineStrategy solid

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



```
  box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
      participant "RAN OAM-related functions" as fmsp
  endbox
endbox
rapp -> fmsp: <<R1>> Query performance information (rAppId, queryCriteria)
fmsp --> fmsp: AuthZ
note right
Check authorization in collaboration
with SME functions
end note
fmsp --> fmsp: Validate request
fmsp -> rapp: <<R1>>Query Performance information (performanceInformation)
@enduml
```

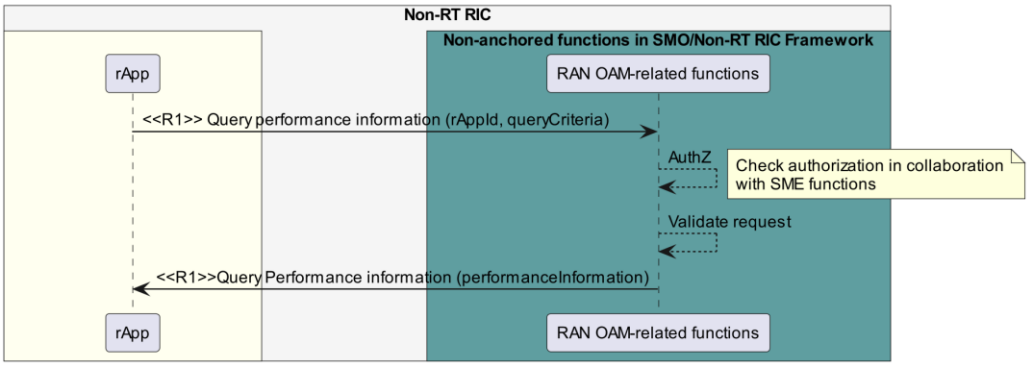

Fig 8.3.4.1-1: Query performance informationuse case flow diagram

8.3.5 Required data

The request for querying performance information contains the rAppId, and query criteria (including information about the related managed entities and about the requested performance information). The performance information query response is scoped by the information provided in the request.

NOTE: The querying of performance information, and how this relates to the Data management and exposure services, is FFS.

9 Use cases for O2-Related Services

NOTE: Use cases for O2-Related services are FFS.

10 Use cases for A1-Related Services

10.1 A1-Related use case 1: Query an A1 policy

10.1.1 Overview

This use case provides the description and requirements for querying an A1 policy.

10.1.2 Background and goal of the use case

Query A1 policy procedure is defined as part of A1 policy management service in R1GAP [2].

10.1.3 Entities/resources involved in the use case

1) A1 Policy management functions
    a. Support functionality to allow rApps to query an A1 policy.
2) rApp
    a. Initiates the Query A1 policy procedure.

10.1.4 Solutions

10.1.4.1 Query A1 policy

Table 10.1.4.1-1: Query A1 policy

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves information about an existing A1 policy.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy management functions in the role of A1 policy management service Producer.	
Assumptions	rApp is authorized to consume the A1 policy management service.	
Preconditions	rApp is aware of the identifier of the A1 policy that it is interested in.	
Begins when	The rApp determines the need to query the information of an A1 policy.	
Step 1 (M)	The rApp queries the information of a single A1 policy with the A1 policy management functions by providing the rAppId and A1 policy identifier.	
Step 2 (M)	The A1 policy management functions validate if the rApp is authorized to access the information of an A1 policy.	
Step 3 (M)	The A1 policy management functions respond with information about the A1 policy.	
Ends when	The rApp has received the information about the A1 policy.	
Exceptions	n/a	
Post Conditions	The rApp has the information about the A1 policy.	
Traceability	n/a	

```
@startuml
   'https://plantuml.com/sequence-diagram
   !pragma teoz true
      skinparam ParticipantPadding 5
      skinparam BoxPadding 10
      skinparam defaultFontSize 12
      skinparam lifelineStrategy solid
      autonumber
     box "Non-RT RIC" #whitesmoke
       box #ivory
          participant rApp as rApp
        endbox
       box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
         participant "A1 policy management functions " as A1P
          endbox
   rApp -> A1P : <<R1>> Query A1 policy request (rAppId, A1 policy identifier)
   activate A1P
   A1P --> A1P : AuthZ
   note right
   Check authorization in collaboration
```



```
            with SME functions
    end note
     A1P -> rApp : <<R1>> Query A1 policy response (A1 policy information)
    Deactivate A1P
@enduml
```

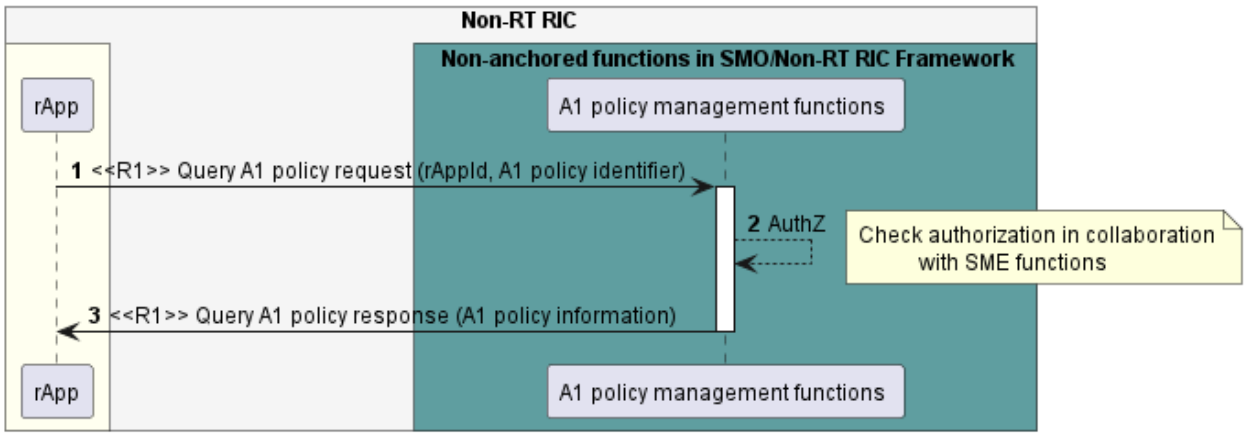

Figure 10.1.4.1-1: Query A1 policy use case flow diagram

10.1.5 Required data

For querying a single A1 policy, the rApp provides the rAppId and the A1 policy identifier. The A1 policy management functions respond with the information about the A1 policy.

10.2 A1-Related use case 2: Query the status of an A1 policy

10.2.1 Overview

This use case provides the description and requirements for querying the status of an A1 policy status.

10.2.2 Background and goal of the use case

Query A1 policy status procedure is defined as part of A1 policy management service in R1GAP [2].

10.2.3 Entities/resources involved in the use case

1) A1 policy management functions
    a. Support functionality to allow rApps to query the status of an A1 policy.
2) rApp
    a. Initiates the procedure for querying status of an A1 policy.

10.2.4 Solutions

10.2.4.1 Query the status of an A1 policy

Table 10.2.4.1-1: Query status of A1 policy

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp is informed about the status of an A1 policy.	

Actors and Roles	• rApp in the role of A1 policy management service Consumer. • A1 policy management functions in the role of A1 policy management service Producer.	
Assumptions	rApp is authorized to use A1 policy management service.	
Preconditions	rApp is aware of the identifier of the A1 policy that it wants to get the status for.	
Begins when	The rApp determines the need to query the status of an A1 policy.	
Step 1 (M)	The rApp queries the status of an A1 policy with A1 policy management functions by providing an rAppId and A1 policy identifier.	
Step 2 (M)	The A1 policy management functions validate if the rApp is authorized to query the status of an A1 policy.	
Step 3 (M)	The A1 policy management functions respond with the status of the corresponding A1 policy.	
Ends when	The rApp has received the status of an A1 policy.	
Exceptions	n/a	
Post Conditions	The rApp has the status of an A1 policy.	
Traceability	n/a	

```
@startuml
'https://plantuml.com/sequence-diagram
!pragma teoz true
    skinparam ParticipantPadding 5
    skinparam BoxPadding 10
    skinparam defaultFontSize 12
    skinparam lifelineStrategy solid
    autonumber
   box "Non-RT RIC" #whitesmoke
     box #ivory
        participant rApp as rApp
      endbox
     box "Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
       participant "A1 policy management functions" as A1P
        endbox
rApp -> A1P: <<R1>>Query A1 policy status request(rAppId,A1 policy identifier)
activate A1P
A1P --> A1P :AuthZ
    note right
    Check authorization in collaboration
              with SME functions
    end note
A1P -> rApp : <<R1>>Query A1 policy status response
Deactivate A1P
@enduml
```

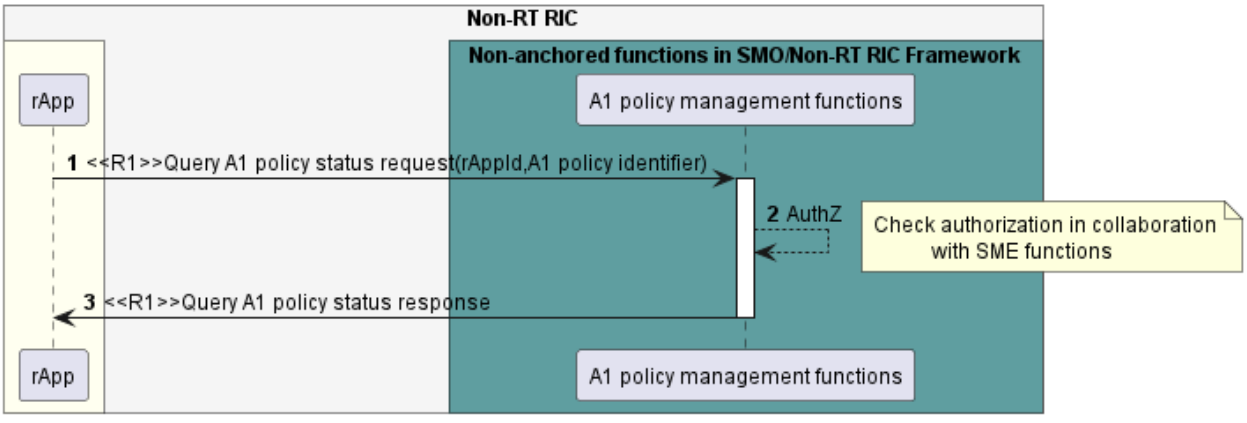

Figure 10.2.4.1-1: Query status of A1 policy use case flow diagram

10.2.5 Required data

For the Query A1 policy status request, the rApp provides the rAppId and the A1 policy identifier.

The A1 policy management functions respond with the status of the A1 policy. The status information of A1 policy is defined in A1 TD [**Error! Reference source not found.**].

11 Use cases for A/ML Workflow Services

NOTE: Use cases for AI/ML Workflow services are FFS.

Revision history

Date	Revision	Description
17-03-2022	01.00.02	Implemented CR 0057, CR0058, CR0060 and CR0064
04-06-2022	01.00.03	Implemented CR0041-v03.00, CR0035v01.00 and CR0065v05.00
08-07-2022	01.00.05	Implemented CR 0043-v02.00, CR044-v3.0, CR0013-v07.00, CR0014-v08.00,CR0048-v02.00, CR0079-v04.00
14-07-2022	01.00.06	Implemented CR 0066 v02.00, CR0095 v02.00.
21-07-2022	01.00.7	Implemented CR0067 v04.00 , CR0096 v02.00, CR0097 v02.00, CR0098 v03.00 , CR0099 v02.00, CR0016 v11.00 , CR 101 v02.00 , CR0050 v01.00 , CR0051 v02.00 along with Editorial updates to remove the O-RAN License AnnexZZZ.
19-08-2022	02.00.01	Implemented CR0058 v02.00
01-09-2022	02.00.02	Implemented CR0109 v01.00
15-09-2022	02.00.03	Implemented CR0110 v04.00
10-11-2022	02.00.04	Implemented CR0141 to migrate to New Template and CR 0013 Version04 and CR 0014 Version02

History

Date	Revision	Description
29-07-2022	02.00	Published as Final version 02.00
01-04-2022	01.00	Published as Final version 01.00

