ETSI TS 104 230 V10.0.0 (2026-02)

TECHNICAL SPECIFICATION

Publicly Available Specification (PAS); O-RAN R1 interface Use Cases and Requirements (O-RAN.WG2.TS.R1UCR-R004-v10.00)

CAUTION

The present document has been submitted to ETSI as a PAS produced by O-RAN Alliance and approved by the ETSI Technical Committee Mobile Standards Group (MSG).

Reference
DTS/MSG-001174

Keywords
interface, PAS, requirements

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by O-RAN Alliance and approved by ETSI Technical Committee Mobile Standards Group (MSG).

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 ETSI Drafting Rules (Verbal forms for the expression of provisions).

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

1 Scope

The present document specifies the Use cases and Requirements for R1 interface.

2 References

2.1 Normative references

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

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

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

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 104 228: "Publicly Available Specification (PAS); O-RAN R1 interface General Aspects and Principles (O-RAN.WG2.TS.R1GAP-R004-v11.00)".

[2] O-RAN.WG2.TS.Use-Case-Requirements-R004: "Non-RT RIC & A1/R1 Interface: Use Cases and Requirements" ("UCR").

[3] ETSI TS 103 988: "Publicly Available Specification (PAS); A1 interface: Type Definitions (O-RAN.WG2.A1TD-R004-v09.00)".

[4] ETSI TS 104 104: "Publicly Available Specification (PAS); O-RAN Security Requirements and Controls Specifications (O-RAN.WG11.SecReqSpecs-R003-v09.01)".

[5] ETSI TS 103 987: "Publicly Available Specification (PAS); A1 interface: Application Protocol (O-RAN.WG2.A1AP-R004-v04.03)".

[6] ETSI TS 103 985: "Publicly Available Specification (PAS); A1 interface: Use Cases and Requirements (O-RAN.WG2.A1UCR-R004-v01.04)".

[7] Recommendation ITU-T M.3020: "Management interface specification methodology".

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, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

Not applicable.

3 Definition of terms, symbols, abbreviations and conventions

3.1 Terms

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

DME type: data type managed and exposed by the DME services and identified by a data type identifier

O-RAN Non-Real-Time RAN Intelligent Controller (Non-RT RIC): 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

NOTE: The Non-RT RIC is comprised of the Non-RT RIC framework and Non-RT RIC applications (rApps).

Non-RT RIC application (rApp): 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.

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

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
FM	Fault Management
FWK	Framework
ID	Identifier
Non-RT RIC	Non-Real Time RAN Intelligent Controller
PM	Performance Management
SME	Service Management and Exposure
SMO	Service Management and Orchestration

3.4 Conventions

3.4.1 General

For the purposes of the present document, the conventions in the following clauses apply.

3.4.2 Requirements

The requirements and solutions for use cases are based on the methodology specified in Recommendation ITU-T M.3020 [7], clause A.1.2.

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 Security

The security requirements applicable to the R1 interface are specified in SRS [4], clause 5.2.6.

5.3 SME services functional requirements

This clause specifies the SME services functional requirements.

The functional requirements are specified in table 5.3-1.

Table 5.3-1: 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.	

5.4 DME services functional requirements

This cause specifies the DME services functional requirements.

The functional requirements are specified in table 5.4-1.

Table 5.4-1: DME services functional requirements

REQ	Description	Note
REQ-R1-DME-Regdatatype-FUN1	The DME services shall support registration, query, update and deregistration of a DME type.	
REQ-R1-DME-Discoverydatatype-FUN1	The DME services shall support querying available DME types.	
REQ-R1-DME-Subscribedatatypechanges-FUN1	The DME services shall support subscription to notifications regarding changes in the availability of DME types.	
REQ-R1-DME-Notifydatatypechanges-FUN1	The DME services shall support sending notifications about changes in the availability of DME types.	
REQ-R1-DME-Unsubscribedatatypechanges-FUN1	The DME services shall support to unsubscribe from notifications regarding changes in the availability of DME types.	
REQ-R1-DME-Dataoffer-FUN1	The DME services shall support the creation of data offers.	
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 subscription-FUN3	The DME services shall support the update of data subscriptions.	
REQ-R1-DME-Data subscription-FUN4	The DME services shall support the query 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.	
REQ-R1-DME-Data subscription-FUN1	The DME services shall support the query the status of data subscriptions.	

5.5 RAN OAM-related services functional requirements

This clause specifies the FM, CM and PM services functional requirements.

The functional requirements are specified in table 5.5-1.

Table 5.5-1: FM, CM 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.	
REQ-R1-OAM-Cmservice-FUN1	The CM service should support retrieving configuration schema related to the O-RAN managed entities.	
REQ-R1-OAM-Cmservice-FUN2	The CM service shall support retrieving configuration data related to O-RAN managed entities.	
REQ-R1-OAM-Cmservice-FUN3	The CM service shall support to write configuration changes to O-RAN managed entities.	

NOTE: The querying of performance information, and how this relates to the Data management and exposure services, is not defined in the present document.

5.6 AI/ML workflow services functional requirements

This clause specifies the AI/ML workflow services functional requirements.

The functional requirements are specified in table 5.6-1.

Table 5.6-1: AI/ML workflow services functional requirements

REQ	Description	Note
REQ-R1-AIML-Deploymodel-FUN1	The AI/ML workflow services shall support the request of deployment of an AI/ML model.	
REQ-R1-AIML-Retrievemodel-FUN1	The AI/ML workflow services shall support retrieving the location details of a registered AI/ML model.	
REQ-R1-AI/ML-training-FUN1	The AI/ML workflow services shall support training of AI/ML models.	
REQ-R1-AI/ML-training-FUN2	The AI/ML workflow services shall support query of AI/ML training job status.	
REQ-R1-AI/ML-training-FUN3	The AI/ML workflow services shall support notification for AI/ML training job status changes.	
REQ-R1-AI/ML-Discovery-FUN1	The AI/ML workflow services shall support discovery of registered AI/ML models.	
REQ-R1-AIML-Registermodel-FUN1	The AI/ML workflow services shall support registration and deregistration of an AI/ML model, and query and update of a model registration.	
REQ-R1-AIML-Performance-FUN1	The AI/ML workflow services shall support subscription to AI/ML model performance information.	
REQ-R1-AIML-Performance-FUN2	The AI/ML workflow services shall support notification of AI/ML model performance information.	
REQ-R1-AIML-inference-FUN1	The AI/ML workflow services shall support request and cancel for the inference of an AI/ML model.	
REQ-R1-AIML-inference-FUN2	The AI/ML workflow services shall support query of the inference capability information for a registered AI/ML model.	
REQ-R1-AI/ML-Registertraincap-FUN1	The AI/ML workflow services should support registration, deregistration, update and query of an AI/ML model trainer's training capability.	

5.7 A1-related services functional requirements

This clause specifies the A1 policy management services and A1 enrichment information related services functional requirements.

The functional requirements are specified in table 5.7-1.

Table 5.7-1: A1 policy management services functional requirements

REQ	Description	Note
REQ-R1-A1P-FUN1	The A1 policy management service shall support querying an A1 policy type.	
REQ-R1-A1P-FUN2	The A1 policy management service shall support querying A1 policy identifiers.	
REQ-R1-A1P-FUN3	The A1 policy management service shall support querying A1 policy type identifiers.	
REQ-R1-A1P-FUN4	The A1 policy management service shall support querying an A1 policy.	
REQ-R1-A1P-FUN5	The A1 policy management service shall support querying the status of an A1 policy.	
REQ-R1-A1P-FUN6	The A1 policy management service shall support subscription to notifications regarding status changes of an A1 policy.	
REQ-R1-A1P-FUN7	The A1 policy management services shall support sending notifications regarding status changes of an A1 policy.	
REQ-R1-A1P-FUN8	The A1 policy management service shall support unsubscription from notifications regarding status changes of an A1 policy.	
REQ-R1-A1P-FUN9	The A1 policy management service shall support creating, querying, updating, and deleting an A1 policy.	

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.

6.1.2 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.

6.1.3 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) assign 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.

6.1.4 Solutions

6.1.4.1 rApp registration and bootstrap

Table 6.1.4.1-1: 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
SEFà SEF : AuthN
note right
registration processing
assign rAppId
end note
SEF -> rApp :<<R1>>Registration response (rAppId)
Deactivate SEF
@enduml
```

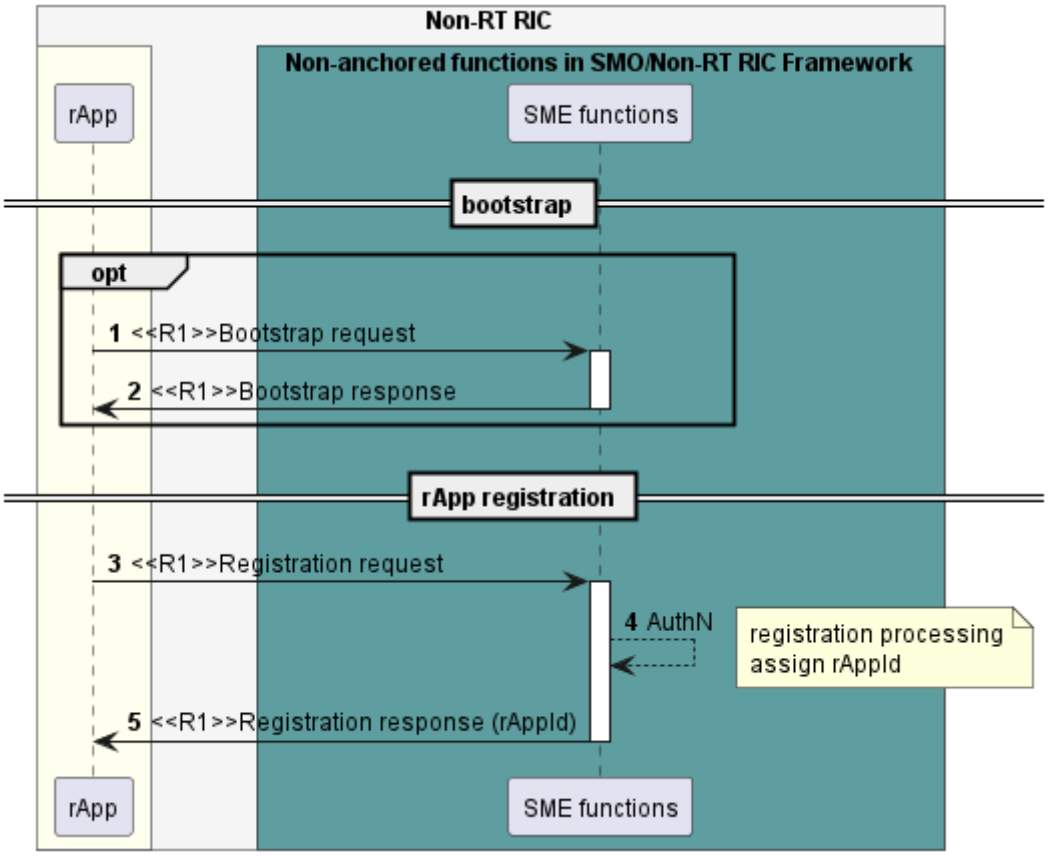

Figure 6.1.4.1-1: rApp registration and bootstrap use case flow diagram

6.1.5 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 its produces. 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 6.2.4.1-1: 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
```

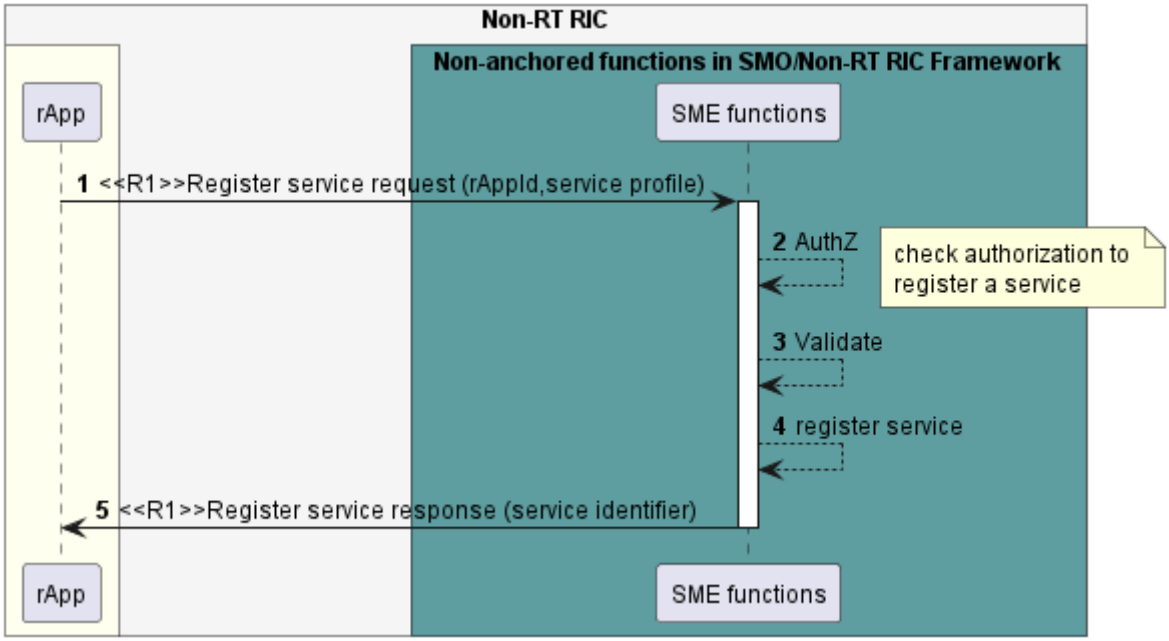

Figure 6.2.4.1-1: Register service use case flow diagram

6.2.4.2 Update service registration

Table 6.2.4.2-1: 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
```

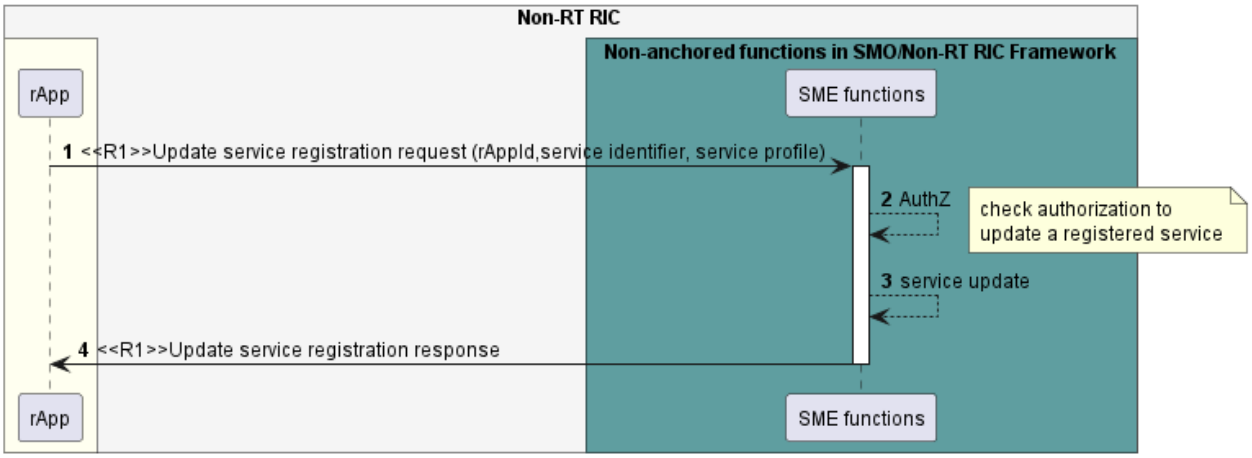

Figure 6.2.4.2-1: Update service registration use case flow diagram

6.2.4.3 Deregister Service

Table 6.2.4.3-1: 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 endpoints 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
```

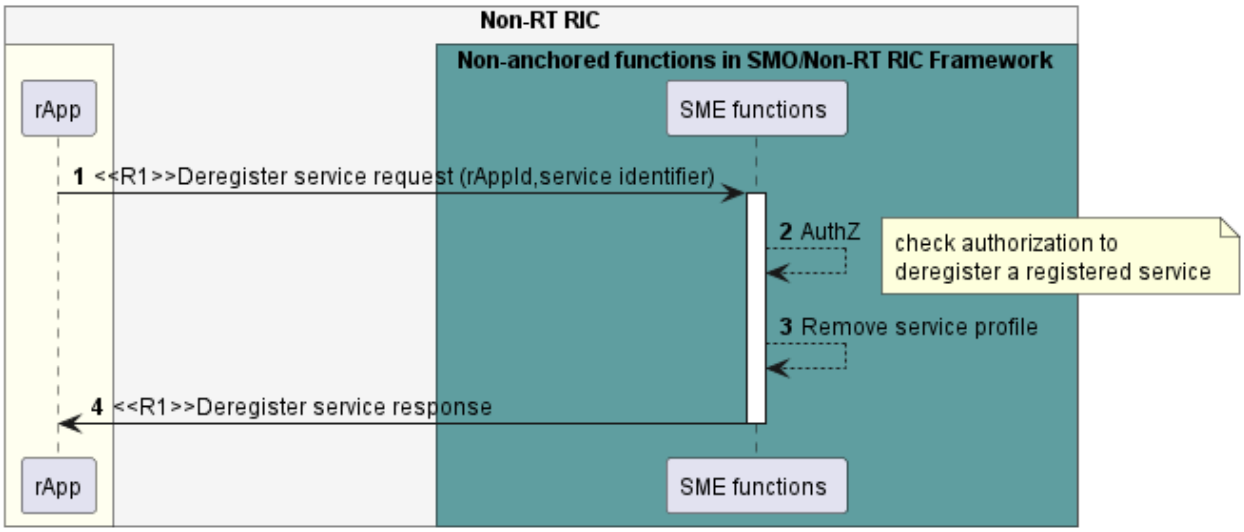

Figure 6.2.4.3-1: Deregister service use case flow diagram

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 [1].

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 6.3.4.1-1: 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 can 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
```

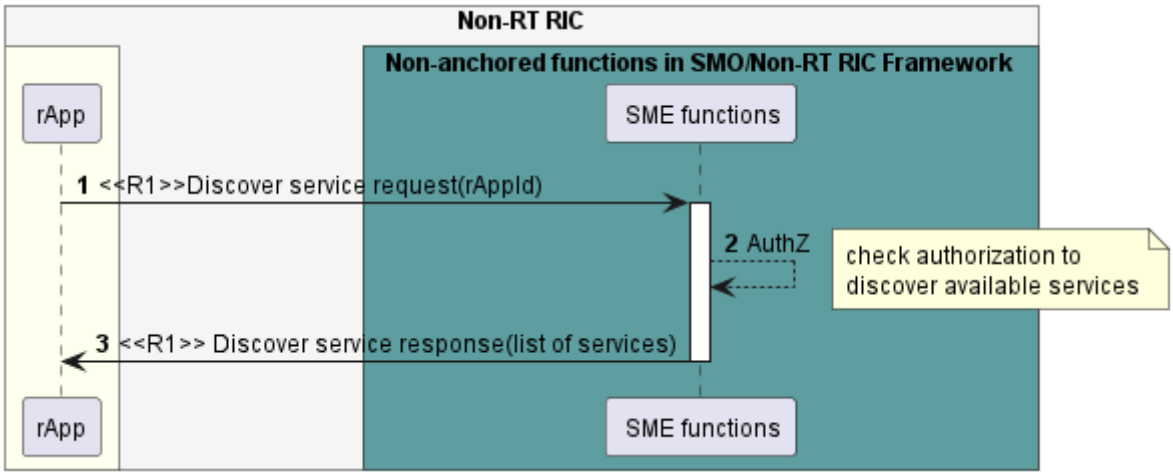

Figure 6.3.4.1-1: Discover service use case flow diagram

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: Whether an rApp may be allowed to discover services for which it is currently not authorized but might be able to obtain authorization later is not defined in the present document.

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, to receive notifications about subscribed events and to update its subscriptions.

6.4.2 Background and goal of the use case

The subscribe service availability, notify service availability changes, unsubscribe service availability and update service availability subscriptions procedures are defined as part of Service management and exposure services in R1GAP [1].

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 as well as to update its subscriptions;

   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, update service availability subscription request and unsubscribe service availability request.

2) rApp:

a) supports functionality to initiate requests to subscribe to and unsubscribe from service availability notifications as well as to update its subscriptions;

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 6.4.4.1-1: 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
```

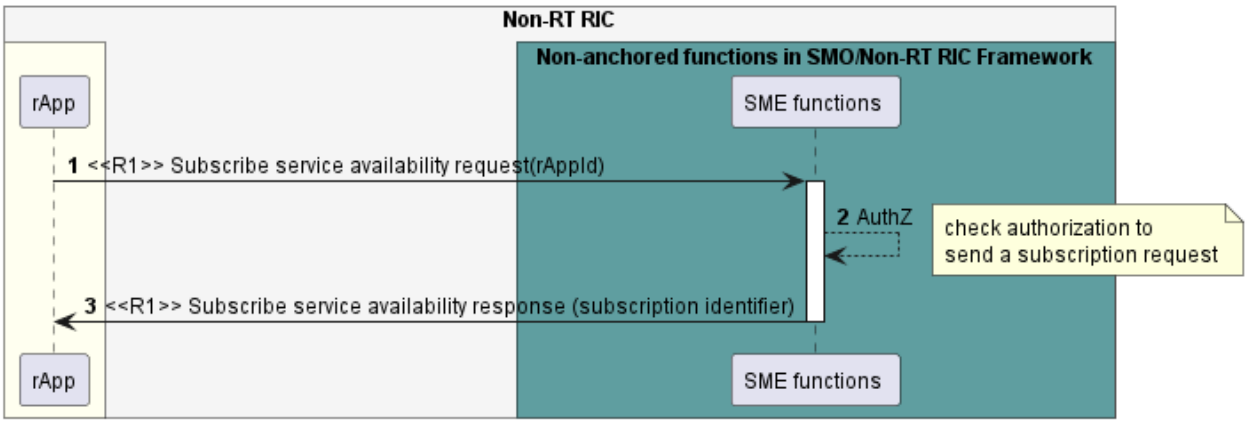

Figure 6.4.4.1-1: Subscribe to of services availability notifications use case flow diagram

6.4.4.2 Notify service availability changes

Table 6.4.4.2-1: 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	
Preconditions	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
```

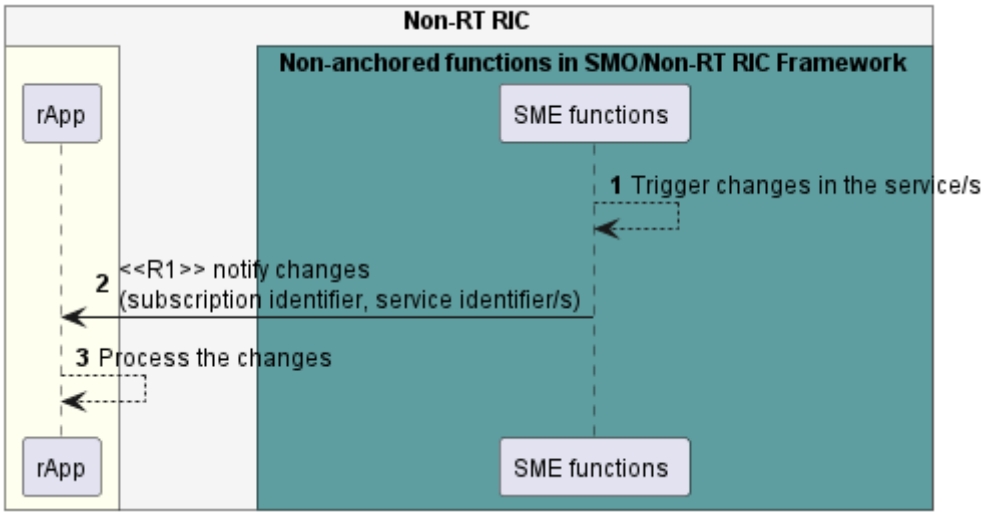

Figure 6.4.4.2-1: Notify service availability changes use case flow diagram

6.4.4.3 Unsubscribe from service availability notifications

Table 6.4.4.3-1: 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	
Preconditions	- 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
```

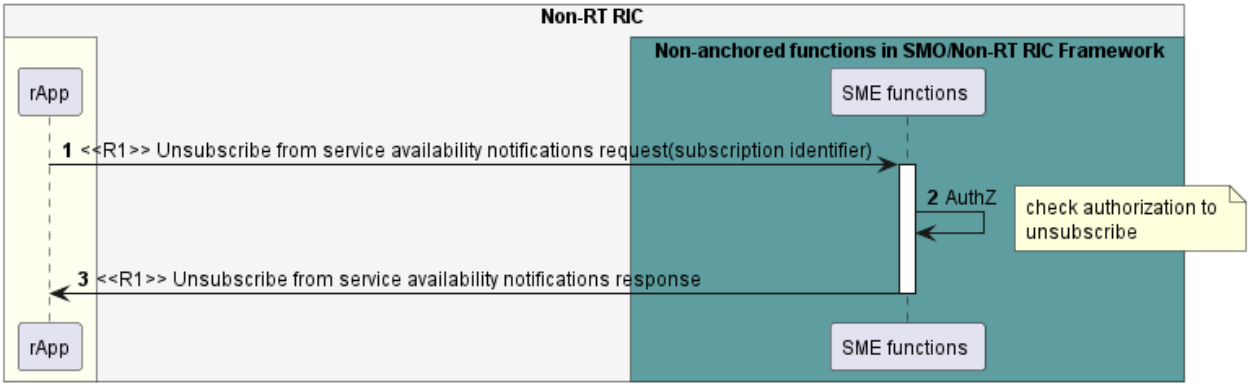

Figure 6.4.4.3-1: Unsubscribe service availability use case flow diagram

6.4.4.4 Update subscriptions to service availability notifications

Table 6.4.4.4-1: Update subscriptions to service availability notifications

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp updates its subscriptions to notifications about changes of the availability of services.	
Actors and Roles	- rApp in the role of Service Consumer that requests updating one of its subscriptions to service availability notifications. - SME functions in the role of SME services Producer entity that handles the update subscription requests.	
Assumptions	n/a	
Preconditions	The rApp is authorized to update its subscriptions to notifications about changes in service availability.	
Begins when	The rApp determines to update one of its subscriptions to notifications regarding changes in the availability of services.	
Step 1 (M)	The rApp requests the SME functions to update one of its subscriptions to notifications regarding changes in the available services with rAppId and subscription identifier.	
Step 2 (M)	The SME functions check whether the rApp is authorized to update the subscription.	
Step 3 (M)	The SME functions send a response.	
Ends when	The rApp was able to update one of its subscriptions to notifications.	
Exceptions	n/a	
Post Conditions	None	
Traceability	REQ-R1-SME-Servicesub-FUN4.	

```
@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>> Update service availability subscription request(subscriptionId, subscription
update information)
```

```
activate SME
 SME --> SME : AuthZ
note right
check authorization to
update the subscription
end note
   SME -> rApp : <<R1>> Update service availability subscription response
deactivate SME
@enduml
```

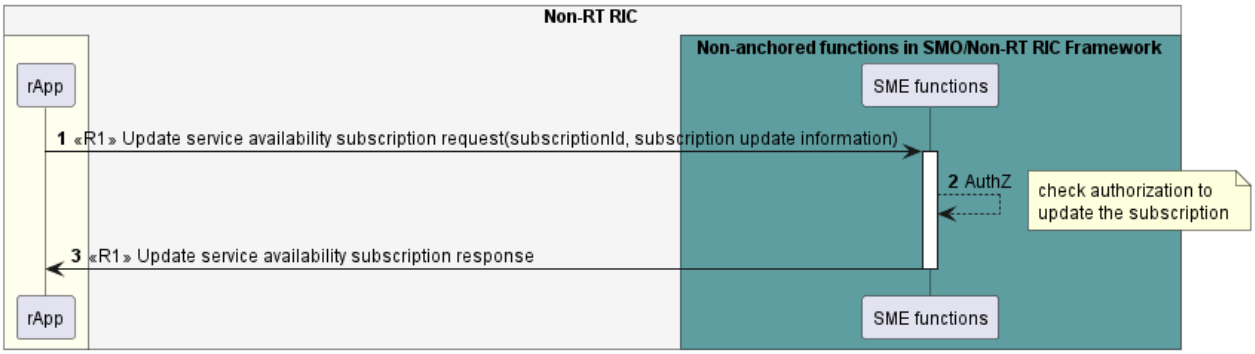

Figure 6.4.4.4-1: Update subscriptions to service availability notifications use case flow diagram

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 respond 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 updating a subscription, a subscribed rApp needs to provide the subscription identifier and subscription update information.

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

6.5 SME use case 5: Query registered services

6.5.1 Overview

This use case allows an rApp as Service Producer to retrieve the service information of registered services as specified in R1GAP [1].

6.5.2 Background and goal of the use case

An rApp as a Service registration service Consumer can retrieve the service information of its registered services from the Service registration services Producer.

6.5.3 Entities/resources involved in the use case

1) SME functions in the role of Service registration service Producer:

   a) support functionality to query the information of registered services;

   b) support authorization of an rApp to determine whether an rApp can query the information of registered services;

   c) retrieve the stored service information and performs filtering of available services based on the selection criteria that may be provided by rApp;

d) provide the response of success or failure result to Query registered services request.

2) rApp:

a) supports to initiate the procedure to query the service information of its registered services.

6.5.4 Solutions

6.5.4.1 Query registered services

Table 6.5.4.1-1: Query registered services use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp is aware of the service information of its registered services.	
Actors and Roles	- rApp in the role of Service registration service Consumer that queries registered services. - SME functions in the role of Service registration service Producer that handles the queries for registered services.	
Assumptions	n/a	
Preconditions	The rApp is authorized to query its registered services.	
Begins when	The rApp determines the need to query registered services.	
Step 1 (M)	The rApp requests the SME functions to query registered services based on the rAppId and optional selection criteria.	
Step 2 (M)	The SME functions check if the rApp is authorized to query registered services.	
Step 3 (M)	The SME functions respond with service API descriptions that match the query.	
Ends when	The rApp was able to query registered services.	
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  " SME functions"   as SME
endbox
rApp -> SME : <<R1>> Query registered services request (rAppId)
activate SME
SME --> SME: AuthZ
note right
check authorization to
query registered services
end note
  SME -> rApp :<<R1>> Query registered services response (service API descriptions)
deactivate SME
@enduml
```

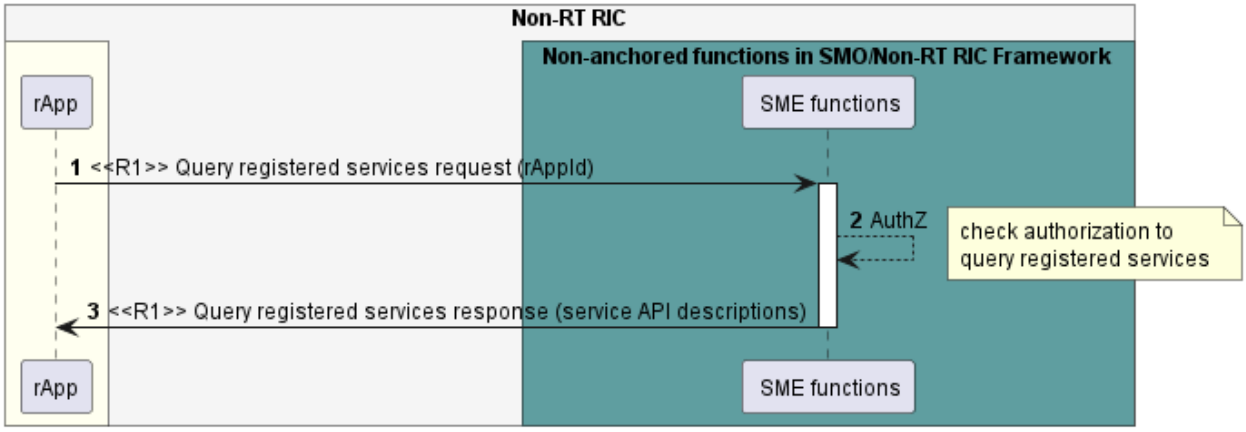

Figure 6.5.4.1-1: Query registered services use case flow diagram

6.5.5 Required data

For querying registered services, the rApp needs to provide the rAppId and optional selection criteria e.g. service name, service type and capabilities.

The SME functions respond with service API descriptions that match the selection criteria.

7 Use cases for Data Management and Exposure Services

7.1 DME use case 1: Data registration, update, and deregistration

7.1.1 Overview

This use case defines how an rApp as DME registration and discovery services consumer registers as producer of data for a DME type, queries a DME type registration, updates a DME type registration and deregisters as producer of data for a DME type.

7.1.2 Background and goal of the use case

The Register DME type Query DME type registration, Update DME type registration and Deregister DME 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) support functionality to allow an rApp to register, query, update and deregister as producer of data for a DME type;

   b) support functionality to allow an rApp to register the constraints for how it can produce and deliver data for the registered DME type;

   c) support validation of DME type information (e.g. schema validation).

2) rApp:

   a) initiates the procedure to register, query, update and deregister as producer of data for a DME type.

7.1.4 Solutions

7.1.4.1 Register DME type

Table 7.1.4.1-1: DME type registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp registers as producer of data for a DME 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 DME 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 DME type.	
Step 1 (M)	The rApp requests the Data management and exposure functions to register as producer of data for a DME type by providing the rAppId and DME 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 DME type information.	
Step 4 (M)	The Data management and exposure functions register the rApp as Data Producer of the DME type. Further, in case this rApp is the first producer of data that registers for the DME type, the Data management and exposure functions add the DME type to the set of discoverable DME types.	
Step 5 (M)	The Data management and exposure functions respond to rApp with successful DME type registration along with DME type registration identifier.	
Ends when	The rApp was able to register as producer of data for a DME type.	
Exceptions	n/a	
Post Conditions	The DME 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 DME type request (rAppId,\n{ DME 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 DME type information
end note
DME --> DME :Register DME type
```

```
DME -> rApp :<<R1>>Register DME type response (DME type registration identifier)
deactivate DME
@enduml
```

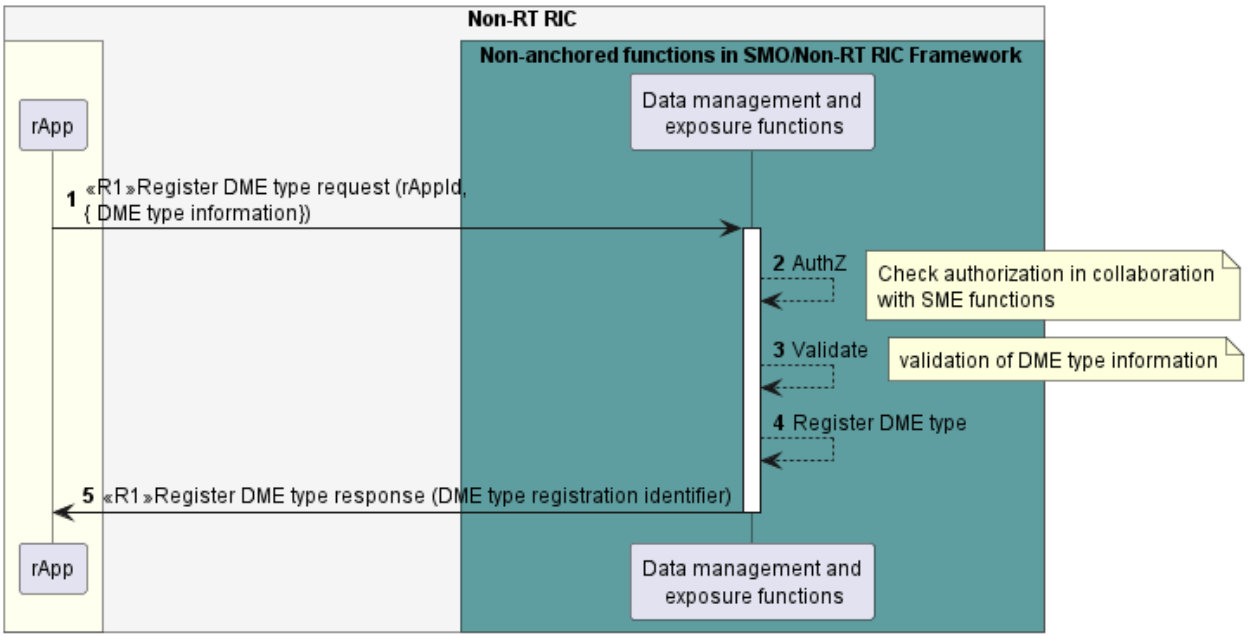

Figure 7.1.4.1-1: Register DME type use case flow diagram

7.1.4.2 Deregister DME type

Table 7.1.4.2-1: DME type deregistration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp deregisters as producer of data for a DME 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 DME type.	
Preconditions	The rApp has registered as producer of data for a DME 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 DME 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 DME type by providing rAppId and DME 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 DME type. Further, in case no other producer of data for the DME type is registered, the Data management and exposure functions remove the DME type from the set of discoverable DME 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 DME type.	
Exceptions	n/a	
Post Conditions	The rApp will not receive any request of data for the DME type and if no other producer is registered, the DME 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 DME type request \n(rAppId,DME type registration identifier)
activate DME
DME --> DME :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME :Deregister DME type
DME -> rApp :<<R1>>Deregister DME type response
deactivate DME
@enduml
```

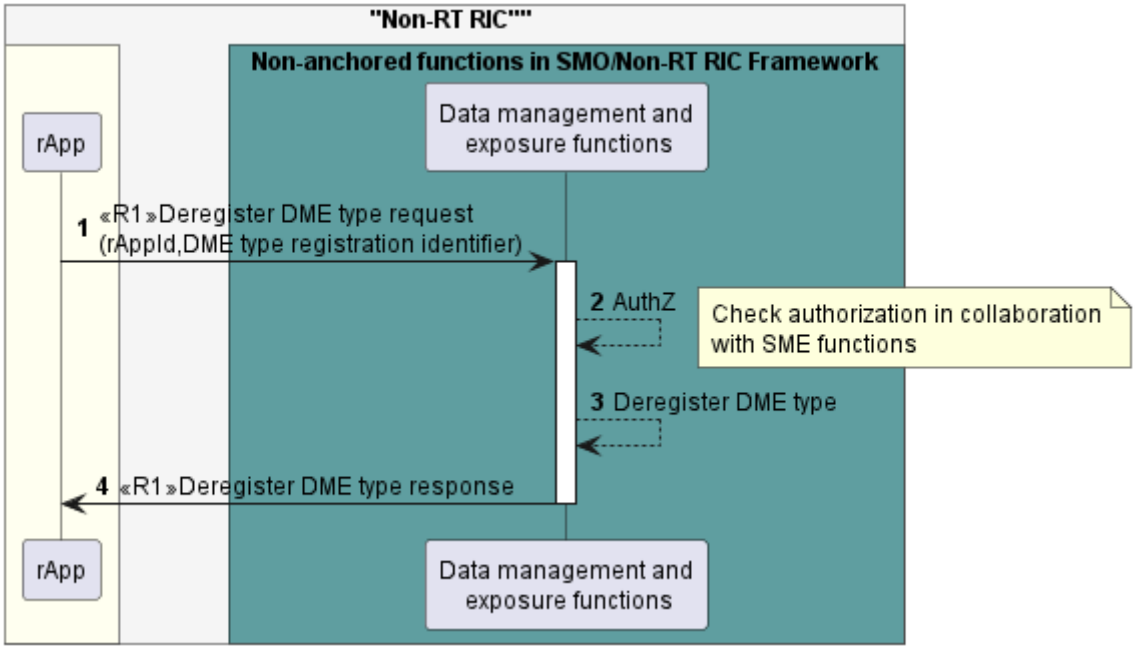

Figure 7.1.4.2-1: Deregister DME type use case flow diagram

7.1.4.3 Update DME type registration

Table 7.1.4.3-1: Update DME type registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp as producer of data updates a DME type registration it has previously created.	
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 has registered as producer of data for a DME type. The rApp is authorized to access the Data management and exposure services.	
Begins when	The rApp determines the need to update a DME type registration it has previously created.	
Step 1 (M)	The rApp requests Data management and exposure functions to update a DME type registration it has previously created by providing rAppId, DME type registration identifier and DME type registration update information.	
Step 2 (M)	The Data management and exposure functions check with SME functions whether the rApp is authorized to update its registered DME type.	
Step 3 (M)	The Data management and exposure functions update the registered DME type.	
Step 4 (M)	The Data management and exposure functions respond to the rApp with successful result.	
Ends when	The rApp has updated the DME type registration.	
Exceptions	n/a	
Post Conditions	The updated DME 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>>Update DME type registration request \n(rAppId,DME type registration identifier,
DME type registration update information)
activate DME
DME --> DME :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME :Update DME type registration
DME -> rApp :<<R1>>Update DME type registration response
deactivate DME
@enduml
```

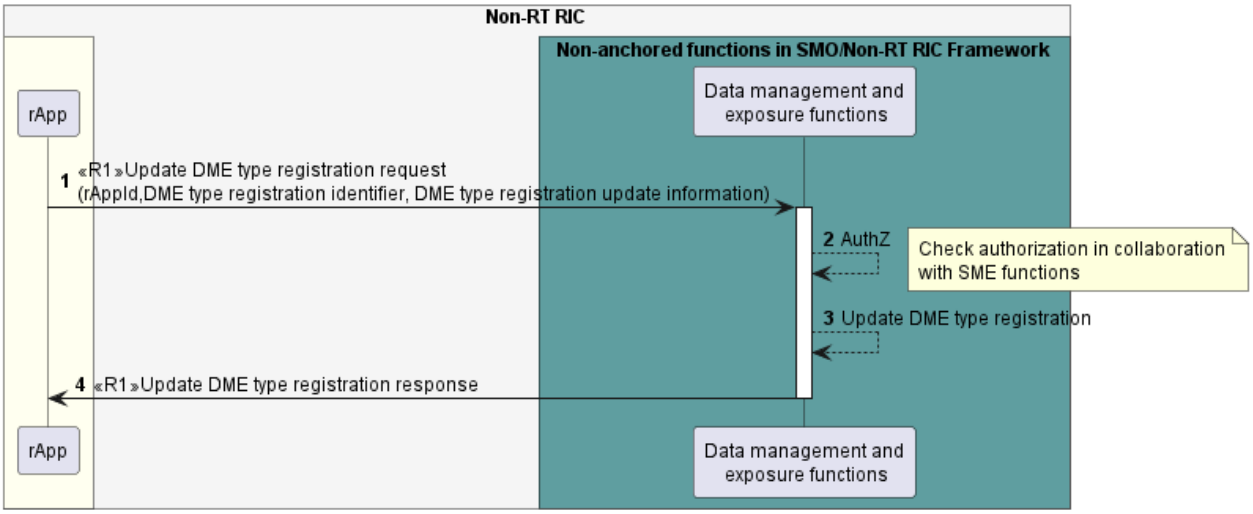

Figure 7.1.4.3-1: Update DME type registration use case flow diagram

7.1.4.4 Query DME type registration

Table 7.1.4.4-1: Query DME type registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp as producer of data query a DME type registration it has previously created.	
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 has registered as producer of data for a DME type. The rApp is authorized to access the Data management and exposure services.	
Begins when	The rApp determines the need to query a DME type registration it has previously created.	
Step 1 (M)	The rApp requests Data management and exposure functions to query a DME type registration it has previously created by providing rAppId and DME type registration identifier.	
Step 2 (M)	The Data management and exposure functions check with SME functions whether the rApp is authorized to query its registered DME type.	
Step 3 (M)	The Data management and exposure functions query the registered DME type.	
Step 4 (M)	The Data management and exposure functions respond to the rApp with DME type registration information.	
Ends when	The rApp has queried the DME type registration.	
Exceptions	n/a	
Post Conditions	n/a	
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>>Query DME type registration request (rAppId, \n DME type registration identifier)
```

```
activate DME
DME --> DME :AuthZ
note right
Check authorization in collaboration
with SME functions
end note
DME --> DME :Query DME type registration
DME -> rApp :<<R1>>Query DME type registration response (DME type registration information)
deactivate DME
@enduml
```

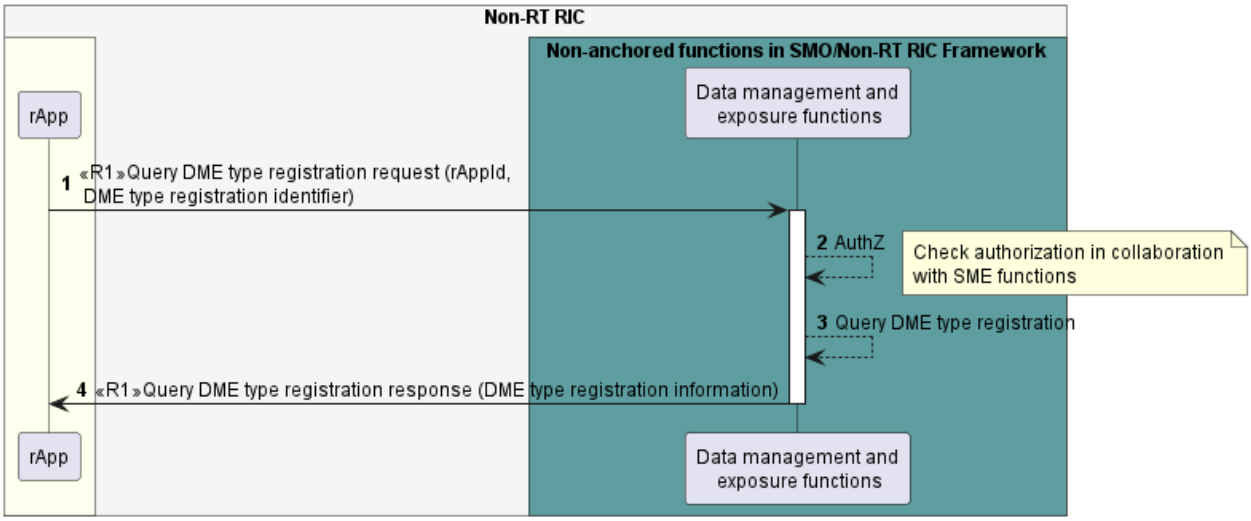

Figure 7.1.4.4-1: Query DME type registration use case flow diagram

7.1.5 Required data

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

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

For querying a DME type registration, the rApp provides the rAppId and DME type registration identifier.

For updating a DME type registration, the rApp provides the rAppId, DME type registration identifier and DME type registration update information.

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

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

7.2 DME use case 2: Discovery of DME types

7.2.1 Overview

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

7.2.2 Background and goal of the use case

The discover DME type and query DME type information, subscribe DME types changes, unsubscribe DME types changes and notify DME 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 DME types that are available;

   b) support functionality allowing rApps to retrieve the information about a specific data type identified by a DME type identifier;

   c) support functionality allowing rApps to subscribe to and unsubscribe from notifications regarding the availability of DME types;

   d) support functionality to notify rApps about changes in the availability of DME types.

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

   a) initiates the procedure to discover DME types and query DME type information, subscribe DME types of changes and unsubscribe DME types changes;

   b) supports functionality to receive notifications regarding changes in the availability of DME types.

7.2.4 Solutions

7.2.4.1 Discover DME type

Table 7.2.4.1-1: Discover DME types

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves the information on available DME 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 DME type is registered and available with the Data management and exposure functions.	
Begins when	The rApp determines the need to discover the available DME types.	
Step 1 (M)	The rApp requests the Data management and exposure functions for the information on the available DME 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 DME types.	
Step 3 (M)	The Data management and exposure functions provide the information about available DME types. The list contains only those available DME types that match the filtering criteria if those were provided by the rApp, or all available DME types otherwise. For each DME type, the data type identifier and meta data are provided.	
Ends when	The rApp was able to discover the available DME types.	
Exceptions	n/a	
Post Conditions	The rApp can retrieve specific information related to a DME 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 DME types 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 DME types response\n(data type identifiers and metadata)
deactivate DME
@enduml
```

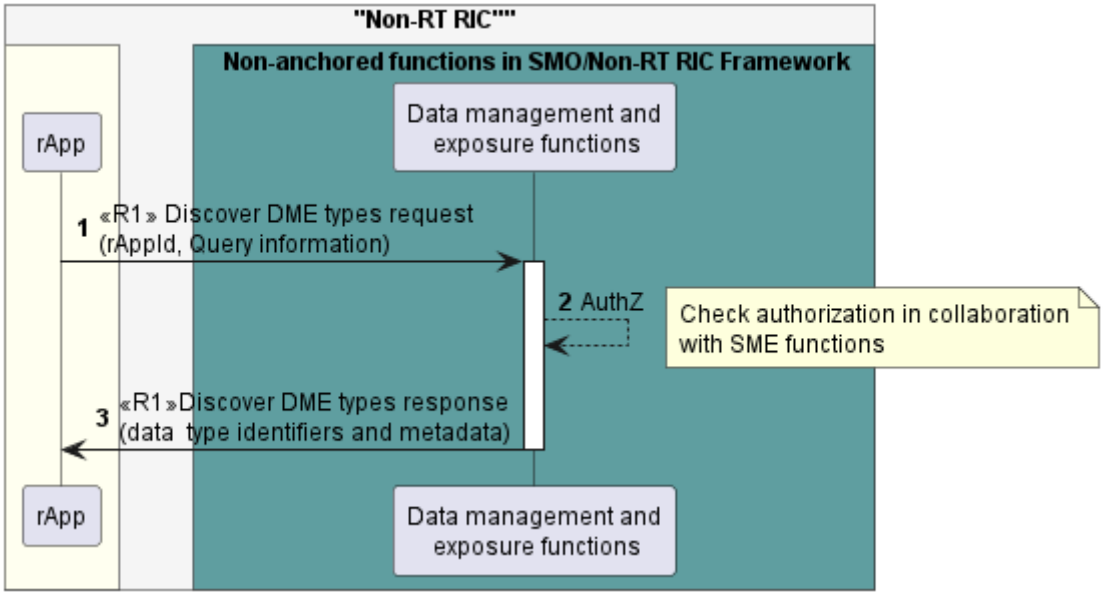

Figure 7.2.4.1-1: Discover DME types use case flow diagram

7.2.4.2 Query DME type information

Table 7.2.4.2-1: Query DME type information

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves detailed information on a specific DME 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 DME type about which it intends to retrieve detail information.	
Begins when	The rApp determines the need to retrieve the information on a specific DME type.	
Step 1 (M)	The rApp requests the Data management and exposure functions for the information on a specific DME 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 DME type.	
Exceptions	n/a	
Post Conditions	The rApp can formulate a request or subscription for data of that DME 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 DME 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 DME type information response \n(DME type information)
deactivate DME
@enduml
```

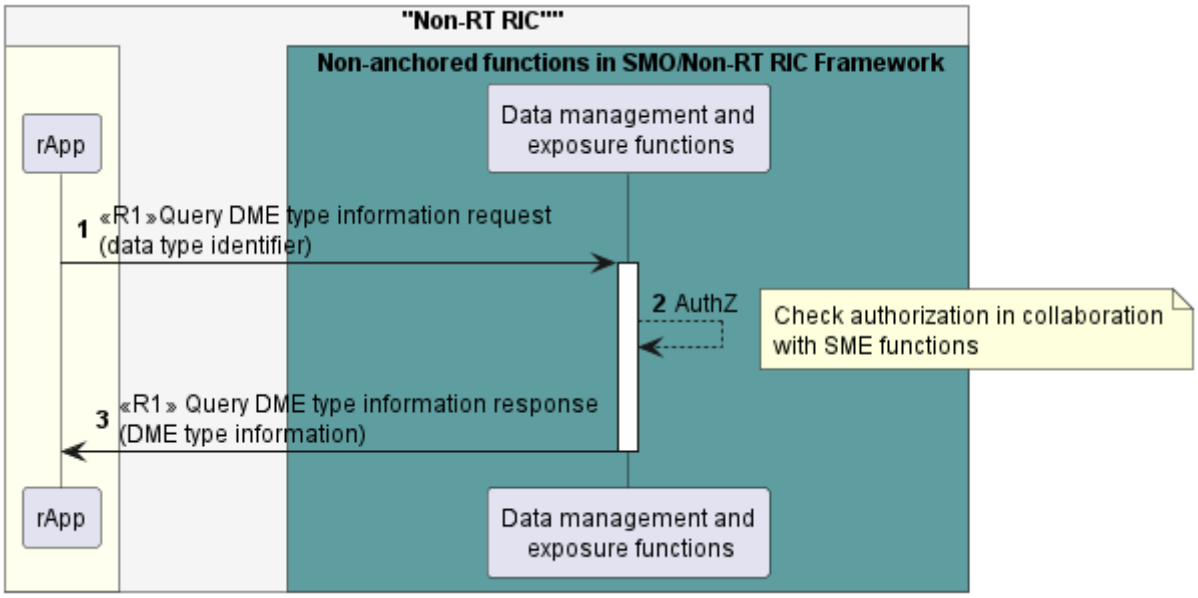

Figure 7.2.4.2-1: Query DME type information use case flow diagram

7.2.4.3 Subscribe DME types changes

Table 7.2.4.3-1: Subscribe DME types changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp subscribes to notifications regarding changes in the availability of DME 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 service. - The rApp is aware of the data type identifier of the DME type.	
Begins when	The rApp determines the need to subscribe to notifications regarding changes in the availability of DME types.	
Step 1 (M)	The rApp requests the Data management and exposure functions to subscribe to notifications regarding DME 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 DME types changes.	
Exceptions	n/a	
Post Conditions	1) The rApp can receive notifications when there are any changes in the set of DME 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 DME 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 DME types changes response \n(subscription ID)
deactivate DME
@enduml
```

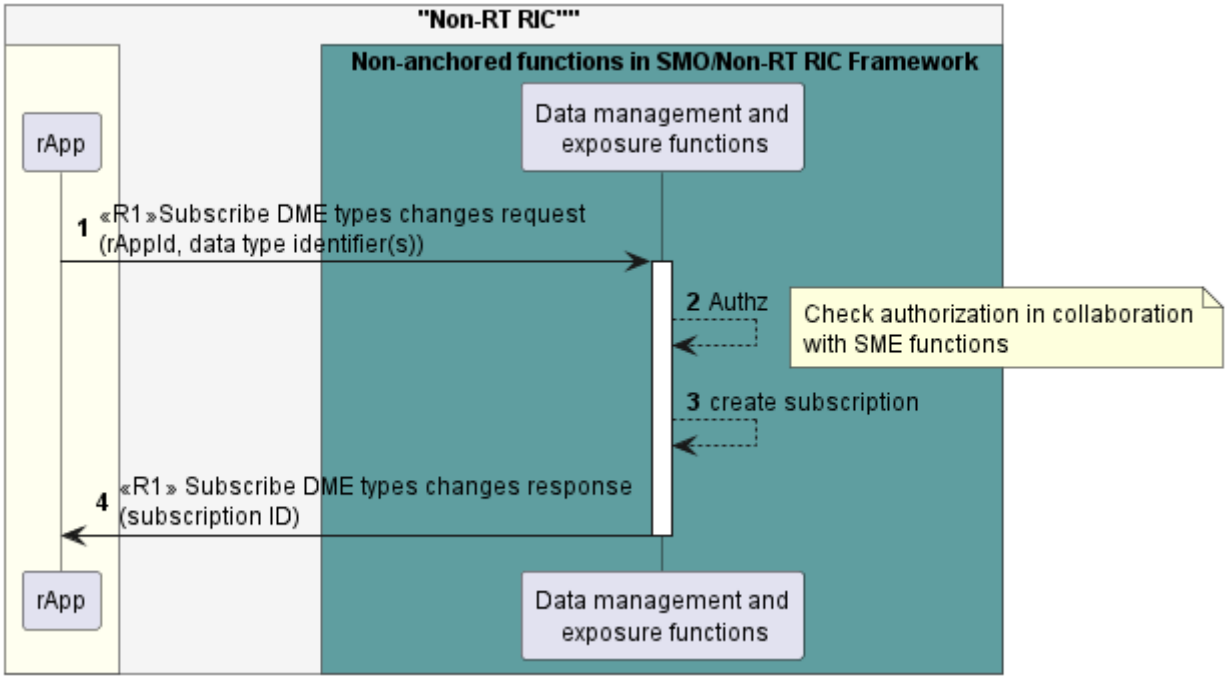

Figure 7.2.4.3-1: Subscribe DME types changes use case flow diagram

7.2.4.4 Notify DME types changes

Table 7.2.4.4-1: Notify DME types changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp receives a notification regarding a change in the availability of DME 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 service. - The rApp is subscribed to notifications.	
Begins when	The Data management and exposure functions determine to send a notification regarding a change in the availability of DME types to the subscribed rApp.	
Step 1 (M)	The Data management and exposure functions send a notification to the subscribed rApp with subscription ID, data type identifier(s) and information about the changes in the availability of DME 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 DME types changes(subscription ID, data type identifier(s),
\navailability change details)
@enduml
```

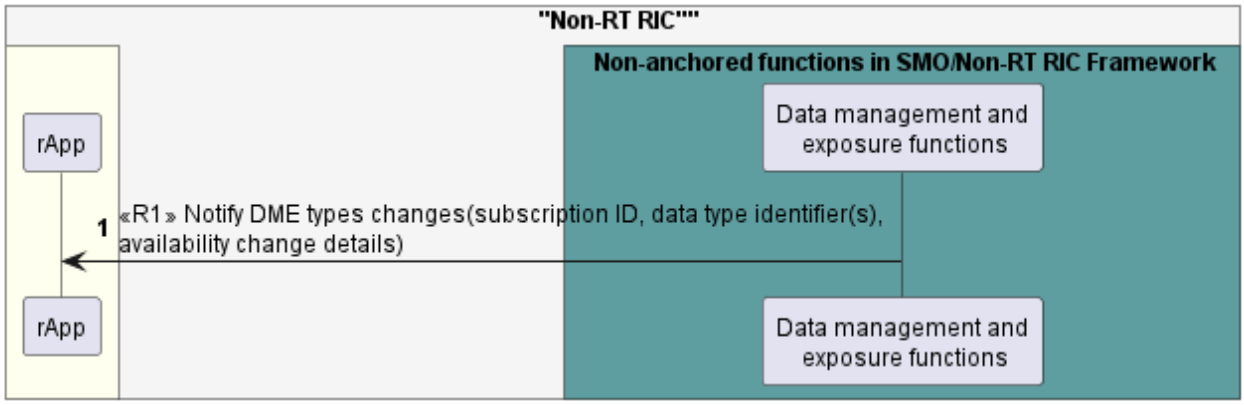

Figure 7.2.4.4-1: Notify DME types changes use case flow diagram

7.2.4.5 Unsubscribe DME types changes

Table 7.2.4.5-1: Unsubscribe DME 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 authorized to access the Data registration and discovery service. - The rApp is subscribed to notifications regarding DME types changes.	
Begins when	The rApp determines the need to unsubscribe from notifications regarding changes in the availability of DME types.	
Step 1 (M)	The rApp requests the Data management and exposure functions to unsubscribe from notifications regarding DME 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 DME 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 DME 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 DME types changes response
deactivate DME
@enduml
```

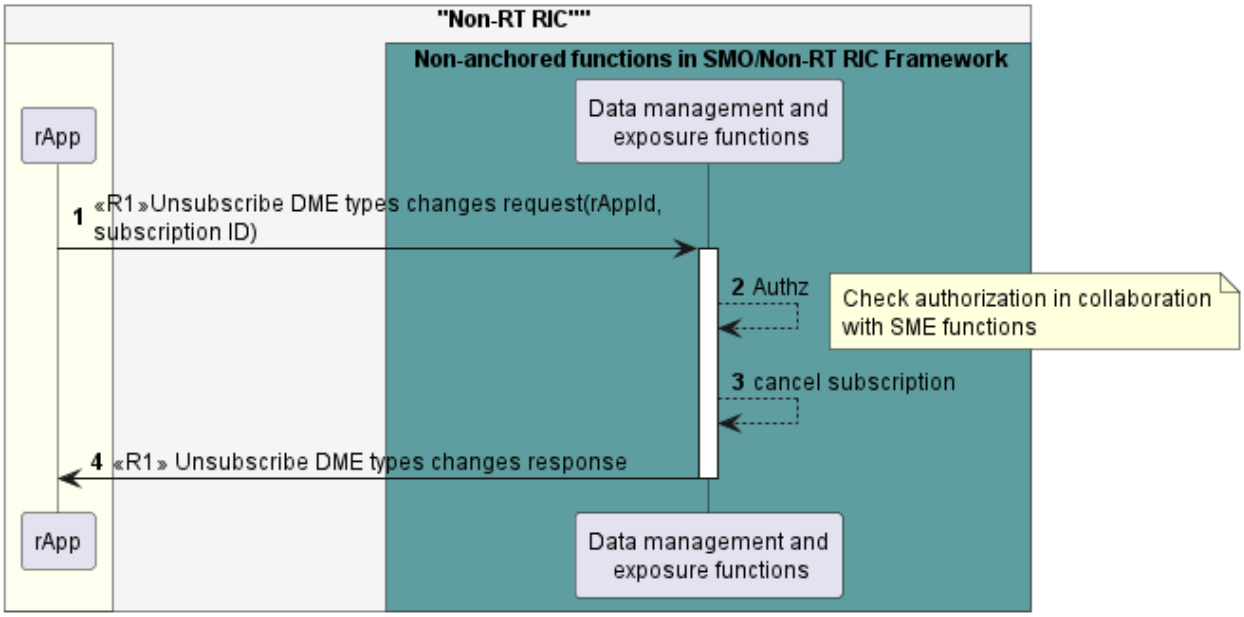

Figure 7.2.4.5-1: Unsubscribe DME types of changes use case flow diagram

7.2.5 Required data

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

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

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

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

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

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

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

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.	

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
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
```

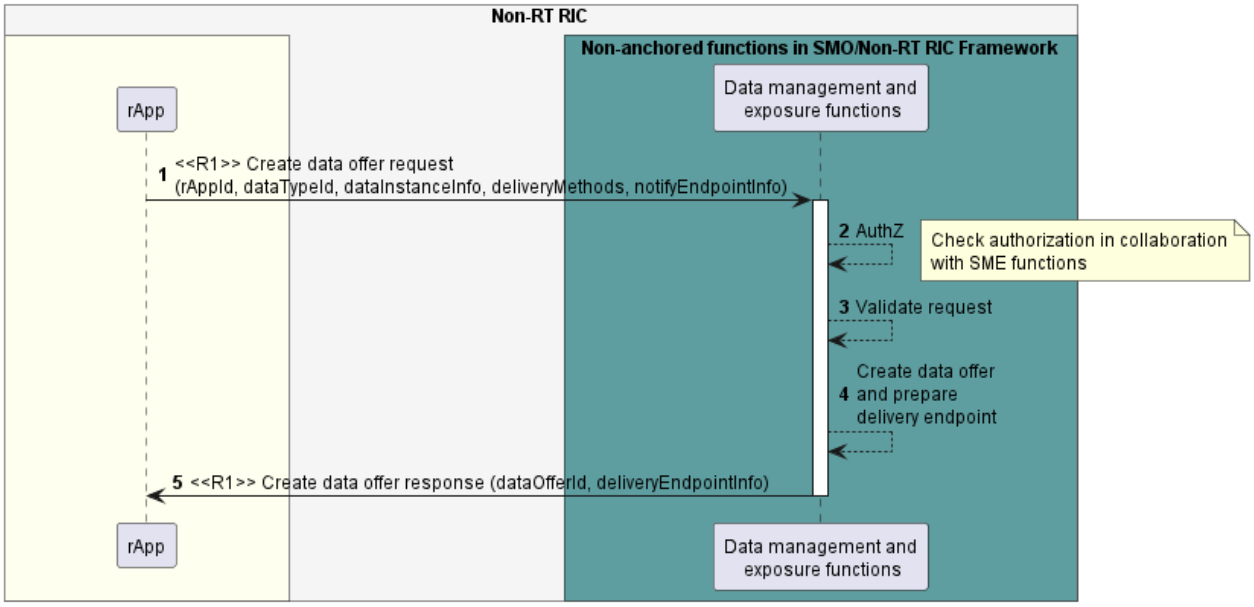

Figure 7.3.4.1-1: Create data offer use case flow diagram

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 1-3 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 1.	
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 4 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
```

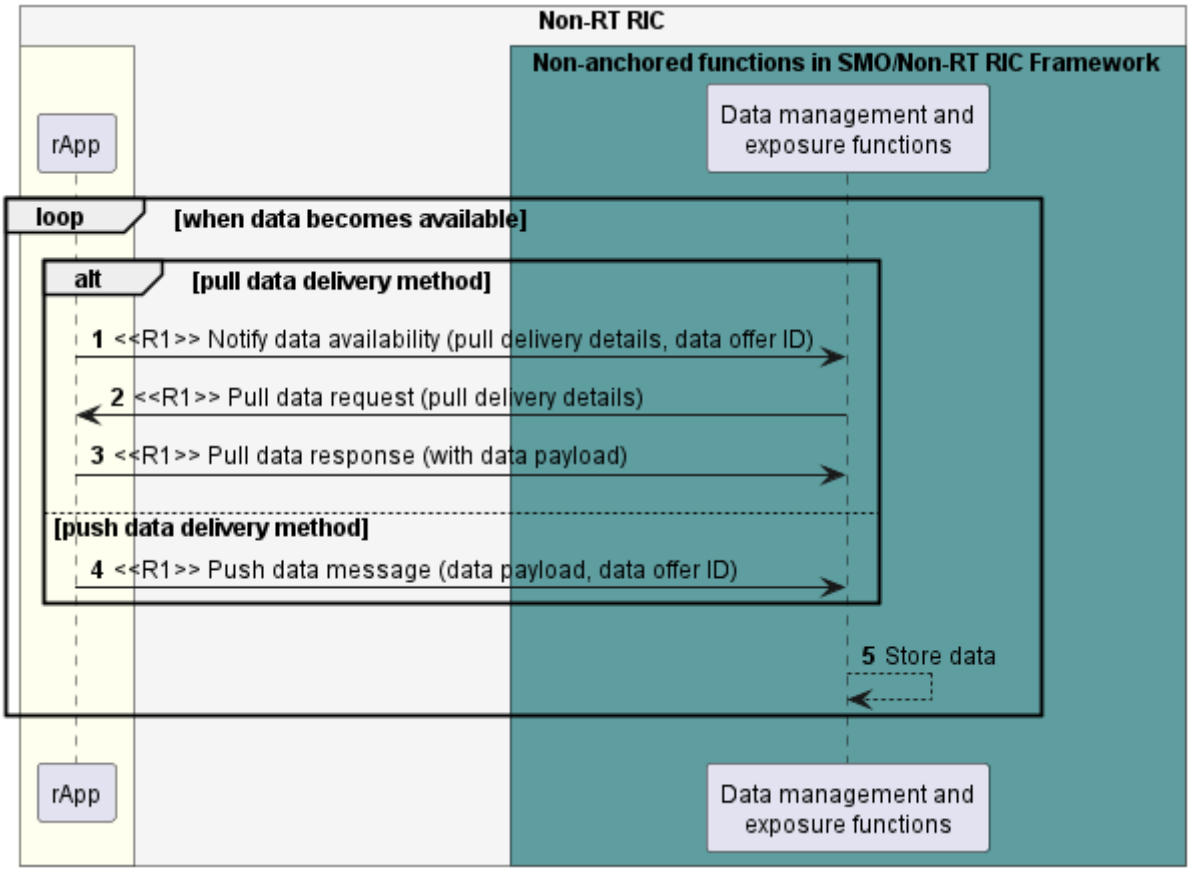

Figure 7.3.4.2-1: Deliver offered data use case flow diagram

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
```

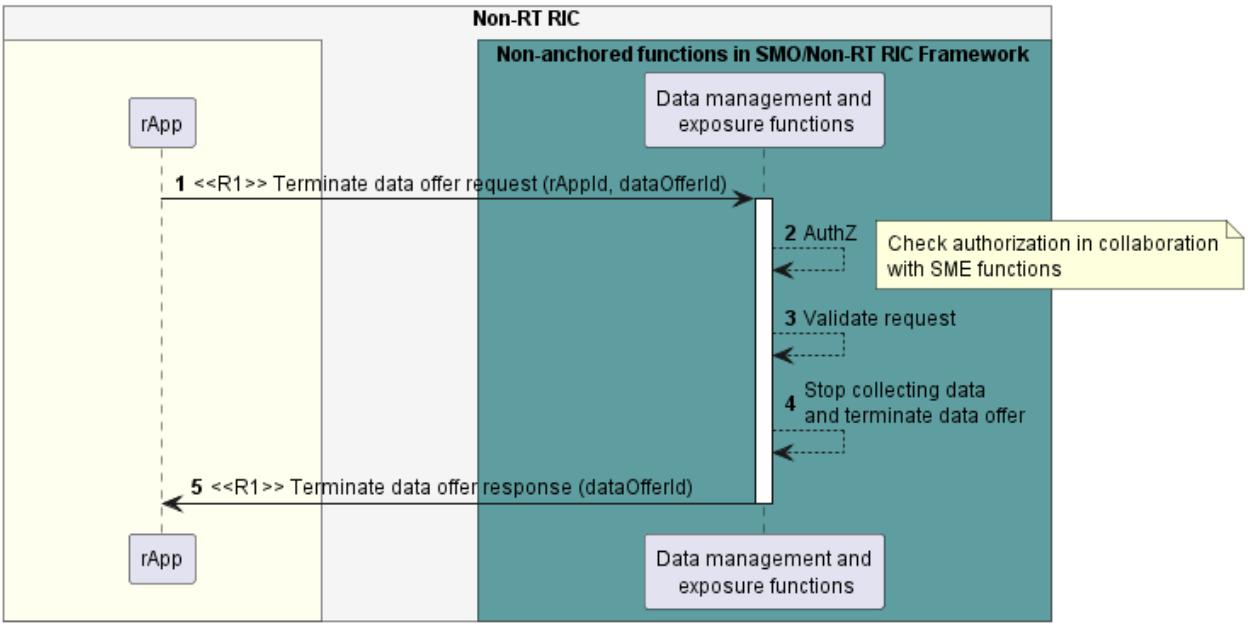

Figure 7.3.4.3-1: Terminate data offer by Data Producer use case flow diagram

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
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
dme -> rapp: <<R1>> Terminate data offer notification (dataOfferId)
dme --> dme: Stop collecting data and\nterminate data offer
rapp --> rapp: Stop delivering data
@enduml
```

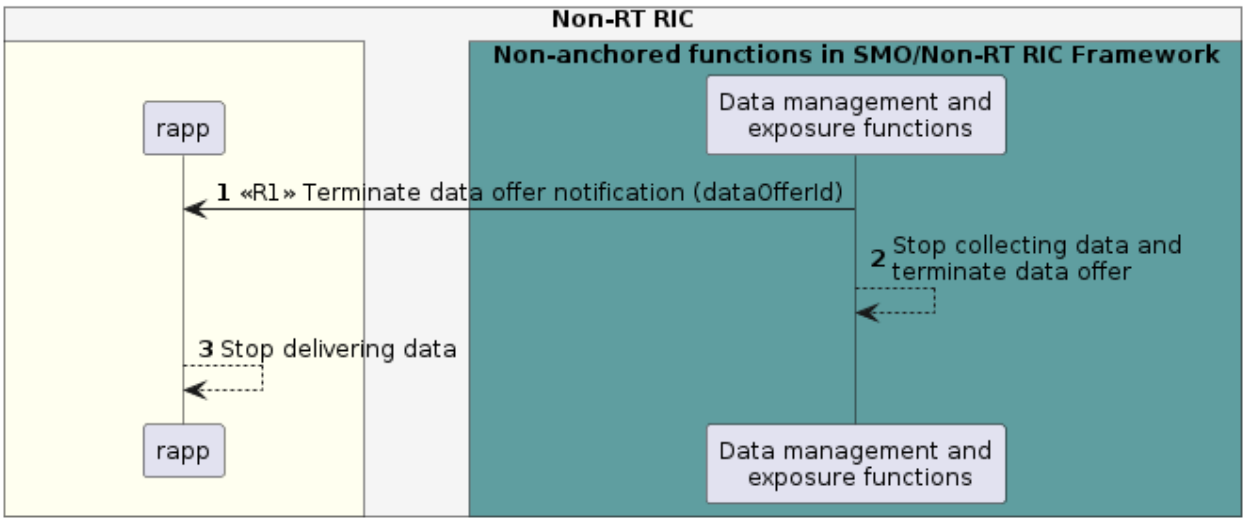

Figure 7.3.4.4-1: Terminate data offer by DME services Producer use case flow diagram

7.3.5 Required data

The Create data offer request contains the rAppId, the registered data type identifier 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 identifier. 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 deliver the requested data.

7.4.3 Entities/resources involved in the use case

1) Data management and exposure functions:
    a) support functionality to receive a data request from an rApp;
    b) support functionality to authorize the rApps request for data;
    c) support 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 intend 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 authorization.	
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
```

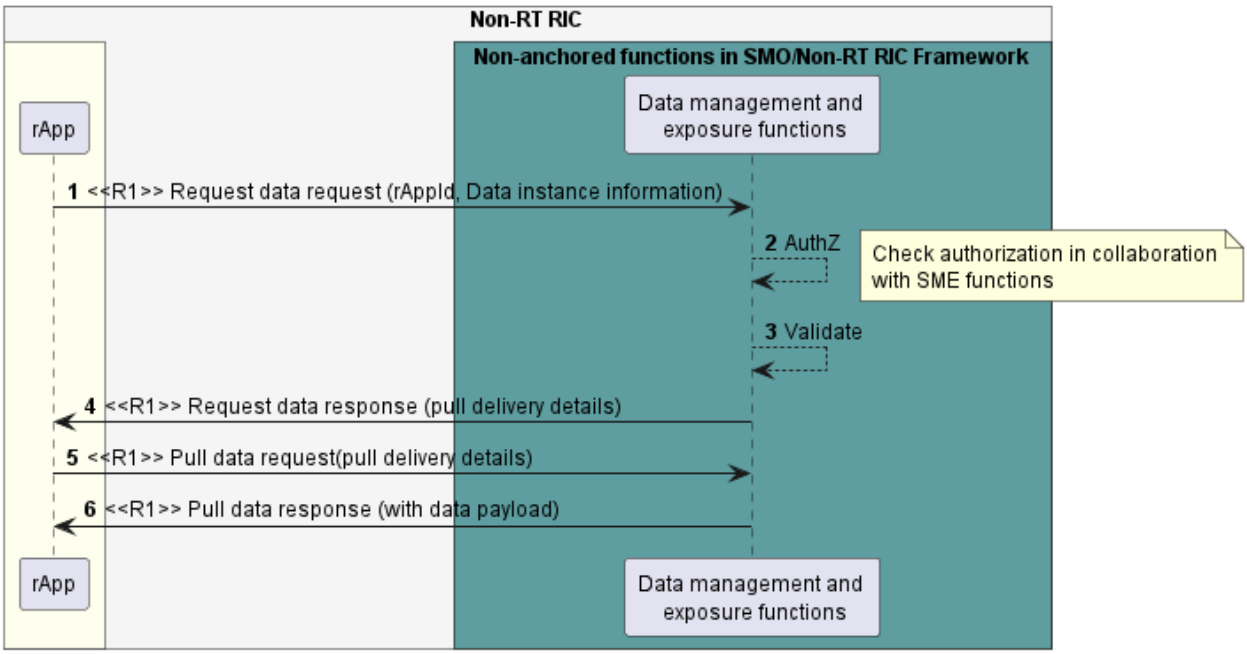

Figure 7.4.4.1-1: A Data Consumer rApp requests data for consumption from the DME functions use case flow diagram

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 requested 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 intend 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	
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 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 to 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
```

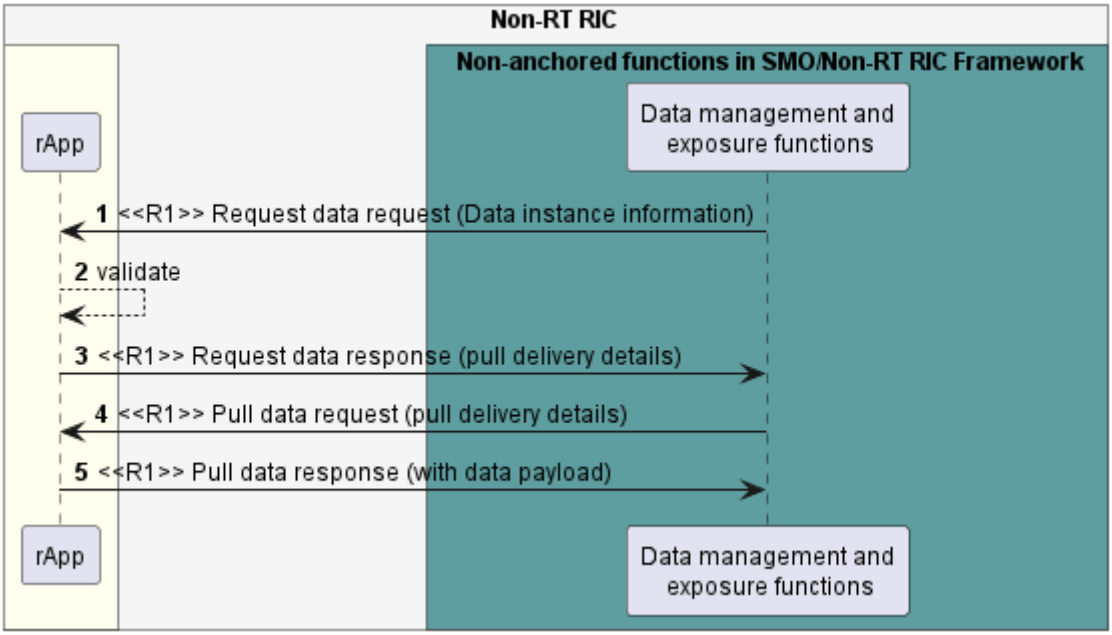

Figure 7.5.4.1-1: DME functions request data for collection from a Data Producer rApp use case flow diagram

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, update the data subscription, query the data subscription, obtain the data, query the status of the data subscription, and terminate the data subscription.

7.6.2 Background and goal of the use case

The Data 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, update the data subscription, query the data subscription, and its status, 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, update the data subscription, query the data subscription, query the subscription status, 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 update a data subscription;

   c) supports functionality to query a data subscription;

   d) supports functionality to terminate a data subscription;

   e) supports functionality to provide data to the DME functions;

   f) supports functionality to provide the status of data subscriptions.

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 subscription service Consumer and Pull data service Consumer and/or Push data service Producer and/or owner of the point to multipoint data stream configuration.	
Assumptions	If point to multipoint data stream is used as delivery method, data stream endpoint is configured as part of preconditions, or the configuration may also take place as part of the subscription.	
Preconditions	- rApp is deployed and authenticated. - Data management and exposure functions have the data type identifier of the needed data. - For point to multipoint data stream delivery method, the data stream endpoint has been configured.	
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.	
Alternative procedure	If point to multipoint data stream is used as delivery method, the Data management and exposure functions provide a data stream endpoint.	
Step 2 (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 and data stream endpoint.	
Step 3 (M)	The Data Producer rApp validates the request.	
Step 4 (M)	The Data Producer rApp creates the subscription.	
Step 5 (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 6 (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 7 (M)	The Data management and exposure functions send a pull data request message, using the pull delivery details obtained in step 5.	
Step 8 (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 9 (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.	
Alternative procedure	Each time a set of data becomes available for delivery, step 10 is executed if point to multipoint data stream is used as the delivery method.	
Step 10 (M)	The Data Producer rApp send the data-to-data stream endpoint.	
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	REQ-R1-DME-Data subscription-FUN1, REQ-R1-DME-Data delivery-FUN1.	

```
@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
alt Pull or push data delivery method
dme -> rApp : <<R1>> subscribe data request (data instance information)
else Point to multipoint data stream delivery method
dme -> rApp : <<R1>> subscribe data request (data instance information,\n data stream endpoint)
end
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)
else Point to multipoint data stream delivery method
rApp -> dme: <<R1>> rApp sends data payload to data stream endpoint

end
end
@enduml
```

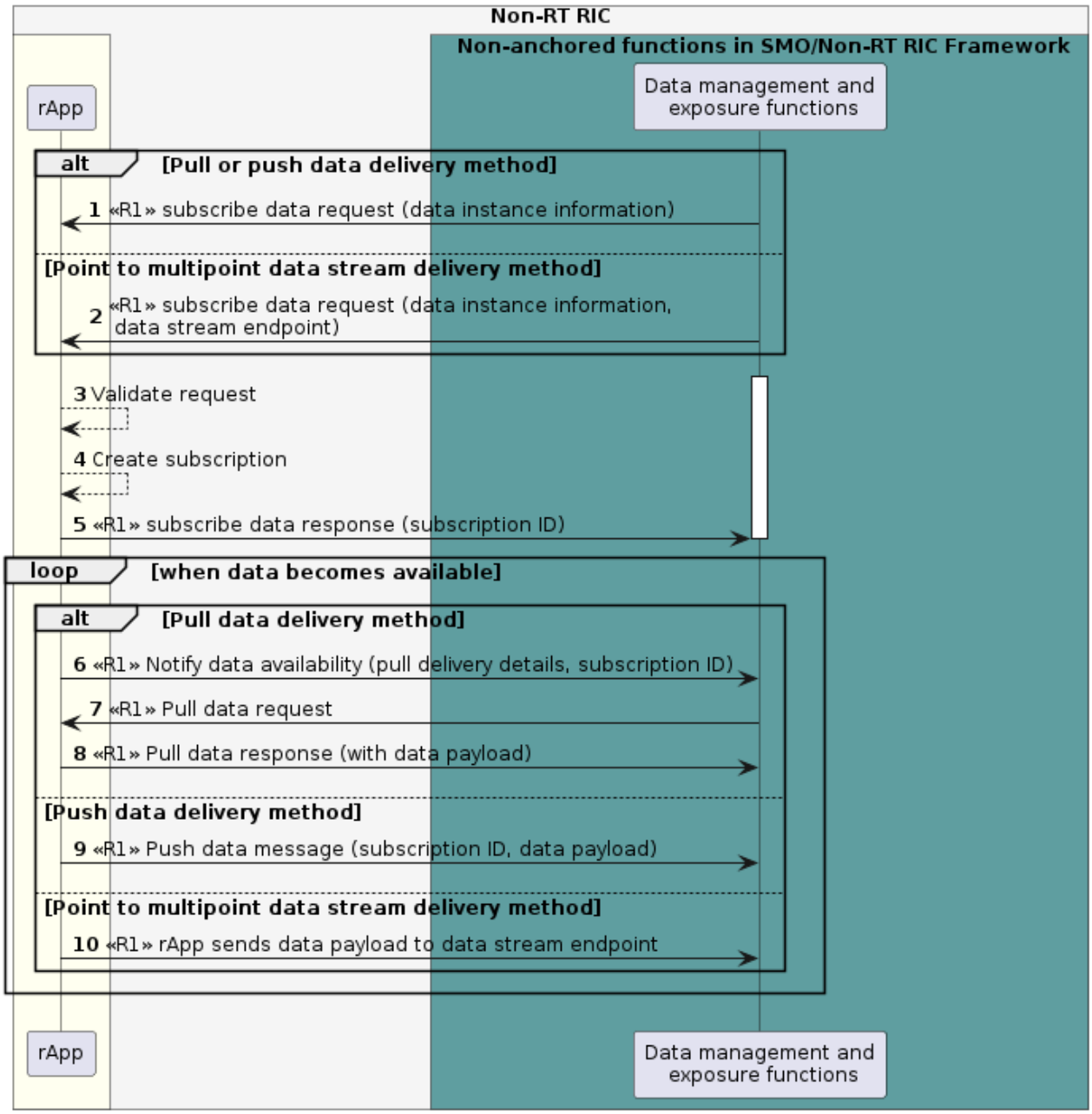

Figure 7.6.4.1-1: DME functions subscribe data for collection from a Data Producer rApp use case flow diagram

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 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	REQ-R1-DME-Data subscription-FUN2.	

```
@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
```

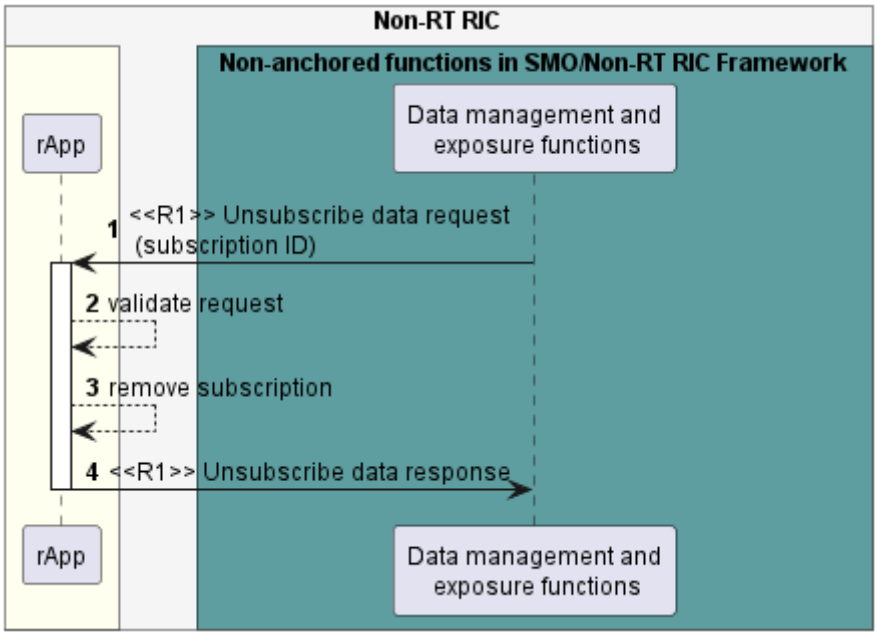

Figure 7.6.4.2-1: The DME functions terminate the data subscription

7.6.4.3 DME functions update a data subscription

Table 7.6.4.3-1: Use case: The DME functions update a data subscription

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	Data management and exposure functions update a data subscription.	
Actors and Roles	- rApp in the role of Data Producer. - Data management and exposure functions in the role of Data subscription service Consumer.	
Assumptions	n/a	
Preconditions	The DME functions have subscribed to data instance.	
Begins when	The DME functions determine to update a data subscription.	
Step 1 (M)	The DME functions send the Update data subscription request with subscription ID and data subscription update information as parameter.	
Step 2 (M)	The rApp validates the request.	
Step 3 (M)	The rApp updates data subscription.	
Step 4 (M)	The rApp responds to the request.	
Ends when	The DME functions have updated the data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-DME-Data subscription-FUN3.	

```
@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>> Update data subscription request\n (subscription ID,\n data subscription update
information)
activate rApp
rApp --> rApp : validate request
```

```
rApp --> rApp : update data subscription
rApp -> DME: <<R1>> Update data subscription response
deactivate rApp
@enduml
```

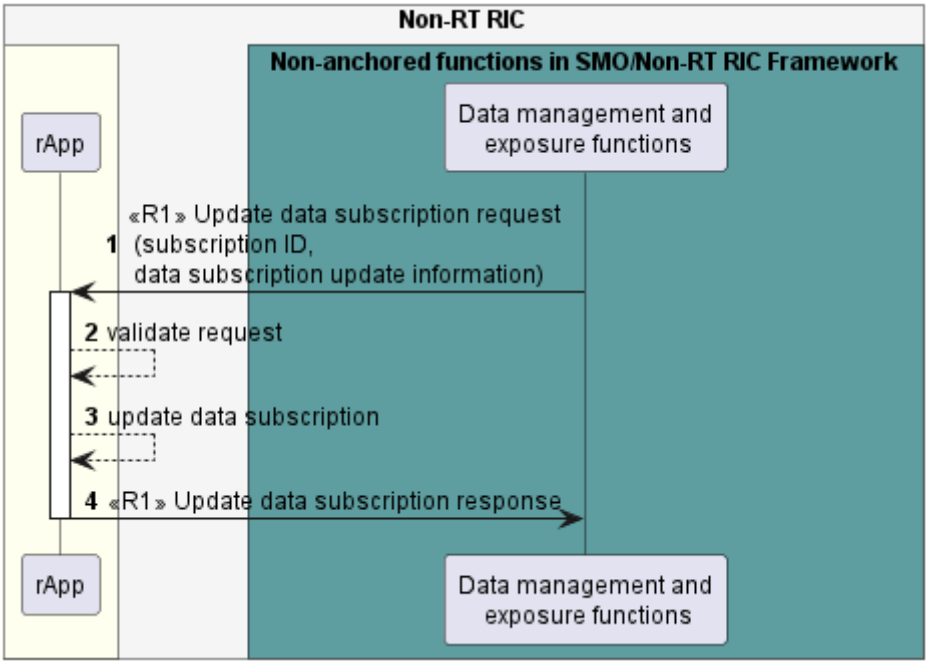

Figure 7.6.4.3-1: The DME functions update the data subscription

7.6.4.4 DME functions query a data subscription

Table 7.6.4.4-1: Use case: The DME functions query a data subscription

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	Data management and exposure functions query a data subscription.	
Actors and Roles	- rApp in the role of Data Producer. - Data management and exposure functions in the role of Data subscription service Consumer.	
Assumptions	n/a	
Preconditions	The DME functions have subscribed to data instance.	
Begins when	The DME functions determine to query a data subscription.	
Step 1 (M)	The DME functions send the Query data subscription request with subscription ID as parameter.	
Step 2 (M)	The rApp validates the request.	
Step 3 (M)	The rApp responds to the request with data subscription information.	
Ends when	The DME functions have queried the data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-DME-Data subscription-FUN4.	

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

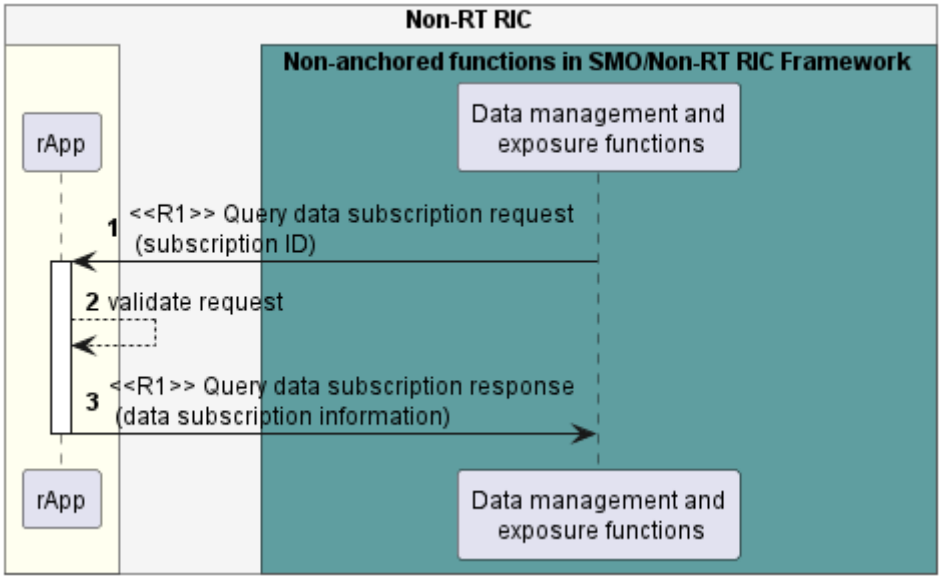

Figure 7.6.4.4-1: The DME functions query the data subscription

7.6.4.5 DME functions query status of data subscription

Table 7.6.4.5-1: Use case: The DME functions query status of data subscription

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	Data management and exposure functions query status of a created data subscription.	
Actors and Roles	- rApp in the role of Data Producer. - Data management and exposure functions in the role of Data subscription service Consumer.	
Assumptions	n/a	
Preconditions	The DME functions have subscribed to data instance.	
Begins when	The DME functions determine the need to query the status a data subscription.	
Step 1 (M)	The DME functions send the Query data subscription status request with subscription ID as parameter.	
Step 2 (M)	The rApp validates the request.	
Step 3 (M)	The rApp responds to the request with the status of the data subscription.	
Ends when	The DME functions have queried the status of data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-DME-Data subscription-FUN1.	

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

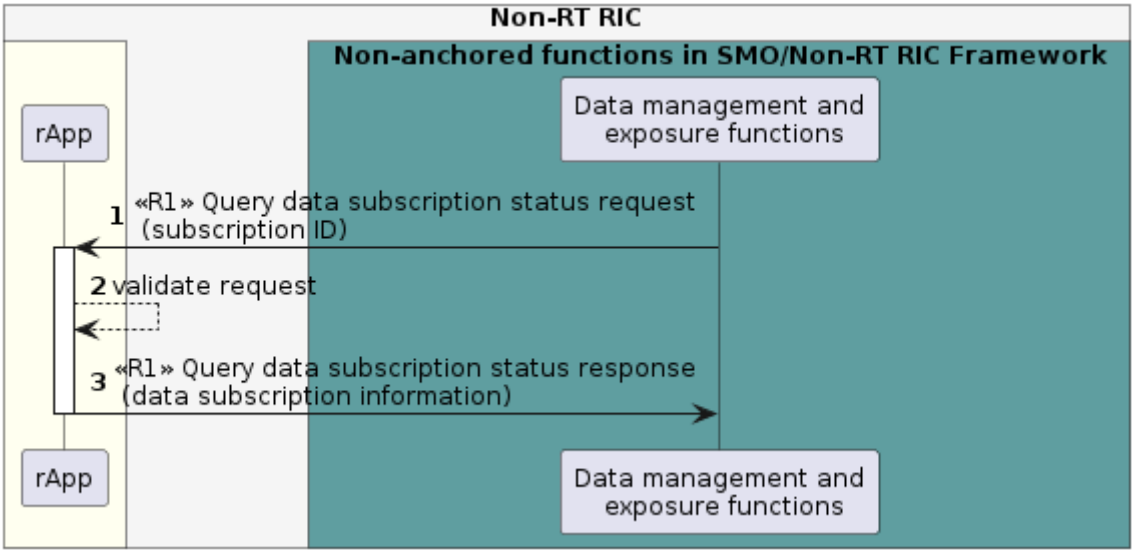

Figure 7.6.4.5-1: The DME functions query the status of data subscription

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.

If point to multipoint data stream is used as delivery method, the Data management and exposure functions provide the data stream endpoint to Data Producer to send data.

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 update data subscription request, the Data management and exposure functions provide the subscription identifier and data subscription update information.

In the query data subscription request, the Data management and exposure functions provide the subscription identifier. In the query data subscription response, the Data Producer rApp provides the data subscription information.

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 query data subscription status request, the Data management and exposure functions provide the subscription identifier. In the query data subscription status response, the Data Producer rApp provides the data subscription status information.

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

When point to multipoint data stream is used, Data Producer rApp sends the data to the data stream endpoint.

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, update the data subscription, query the data subscription obtains the data, and terminates the data subscription.

7.7.2 Background and goal of the use case

The Data 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 DME type information it is aware of e.g. from discovery or configuration, update the data subscription, query the data subscription, 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, update the data subscription, query the data subscription 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 an update of a data subscription;

   c) supports functionality to initiate a query of a data subscription;

   d) supports functionality to initiate the termination of a data subscription;

   e) 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	If point to multipoint data stream is used as delivery method, data stream endpoint configured as part of preconditions, or the configuration may also take place as part of the subscription.	
Preconditions	1) The rApp is authorized to access the Data management and exposure services. 2) The rApp has obtained the DME type information e.g. by discovery or configuration. 3) For point to multipoint data stream delivery method, data stream endpoint has been configured by Data management and exposure functions.	
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	If point to multipoint data stream is used as a delivery method, the Data management and exposure functions provide the data stream endpoint.	
Step 6 (M)	The Data management and exposure functions respond to the request with subscription ID as a parameter and data stream endpoint.	
Alternative procedure	Each time a set of subscribed data becomes available for delivery, steps 7-9 are executed if "pull data" is used as the data delivery method.	
Step 7 (M)	The Data management and exposure functions notify the availability of data with subscription ID and pull delivery details as parameters.	
Step 8 (M)	The rApp send a pull data request based on the pull delivery details received in the previous step.	
Step 9 (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, step 10 is executed if the "push data" is used as the data delivery method.	
Step 10 (M)	The Data management and exposure functions send a push data message with subscription ID and data payload as parameters.	
Alternative procedure	Each time a set of data becomes available for delivery, step 11 is executed if point to multipoint data stream is used as the data delivery method.	
Step 11 (M)	rApp fetches the data from the data stream endpoint.	
Ends when	The rApp has terminated the subscription.	
Exceptions	n/a	
Post Conditions	The rApp has the data it determined it needed.	
Traceability	REQ-R1-DME-Data subscription-FUN1, REQ-R1-DME-Data delivery-FUN1.	

```
@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
Alt Pull or push delivery method

rApp <- DME : <<R1>> Subscribe data response(subscription ID)
else Point to multipoint data stream delivery method
rApp <- DME : <<R1>> Subscribe data response(subscription ID, data stream endpoint)
end


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)
else Point to multipoint data stream delivery method
rApp -> DME: <<R1>> rApp fetches data payload from data stream endpoint

end
end

@enduml
```

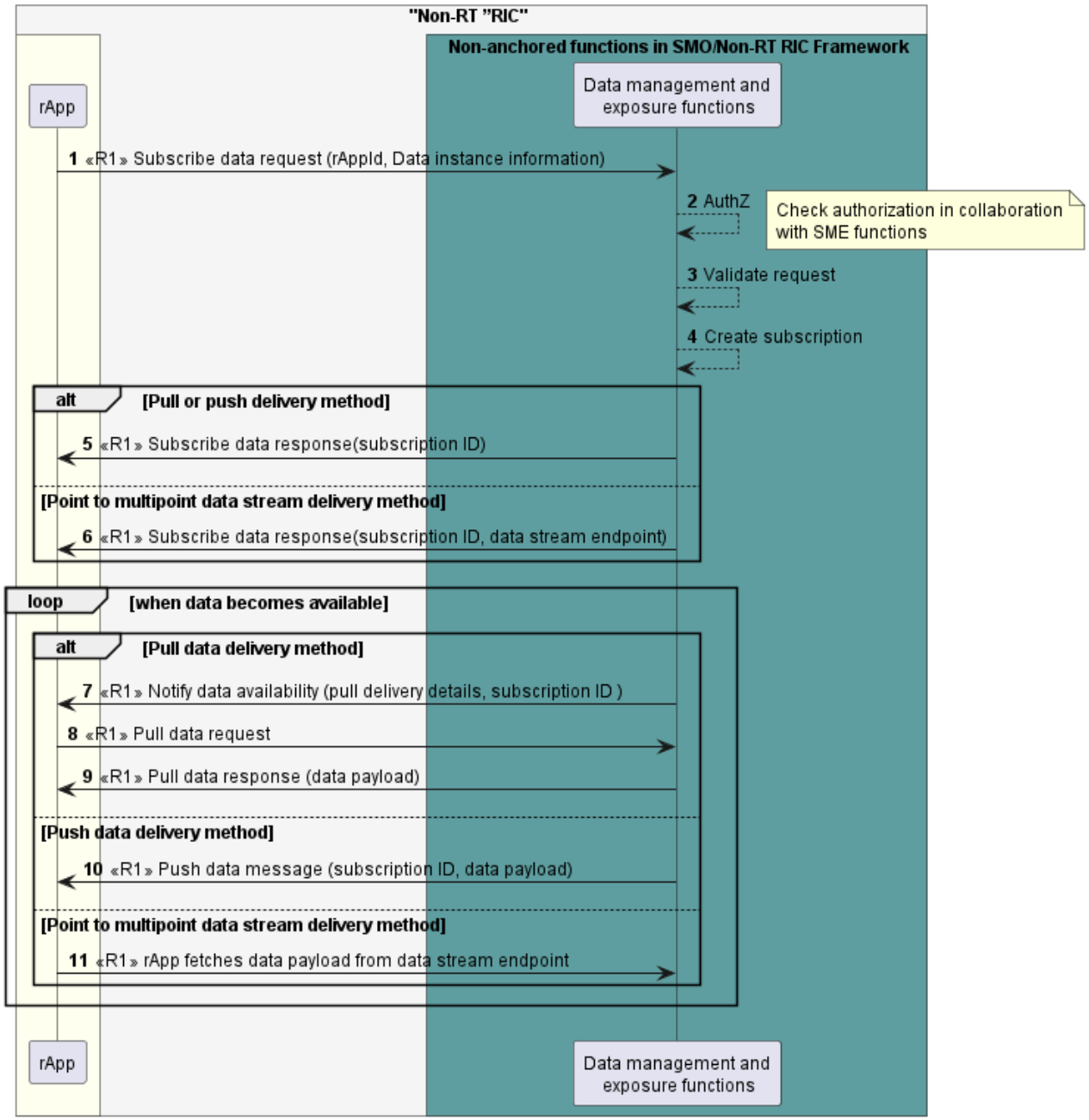

Figure 7.7.4.1-1: Subscribe data and delivery of subscribed data use case flow diagram

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	REQ-R1-DME-Data subscription-FUN2.	

```
@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
```

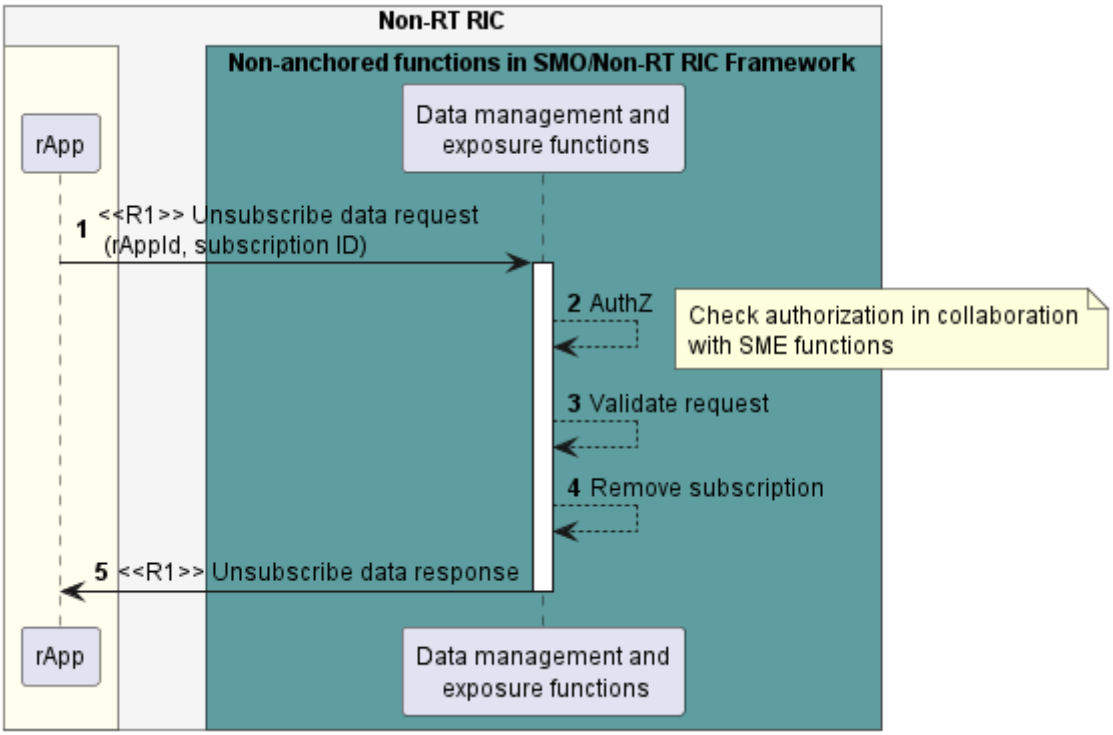

Figure 7.7.4.2-1: Terminate data subscription use case flow diagram

7.7.4.3 Update a data subscription

Table 7.7.4.3-1: Update a data subscription

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	rApp update a 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 subscribed to data instance.	
Begins when	The rApp determines to update a data subscription.	
Step 1 (M)	The rApp sends the Update data subscription request with rAppId, subscription ID and data subscription update information as parameters.	
Step 2 (M)	The Data management and exposure functions check for authorization with SME functions whether the rApp is authorized to update a data subscription.	
Step 3 (M)	The Data management and exposure functions validate the request.	
Step 4 (M)	The Data management and exposure functions updates the data subscription.	
Step 5 (M)	The Data management and exposure functions respond to the request.	
Ends when	rApp has updated the data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-DME-Data subscription-FUN3.	

```
@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>> Update data subscription request\n (rAppId, subscription ID\n, data subscription
update information)
activate DME
DME --> DME : AuthZ
note right
Check authorization in collaboration
with SME functions
end note

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

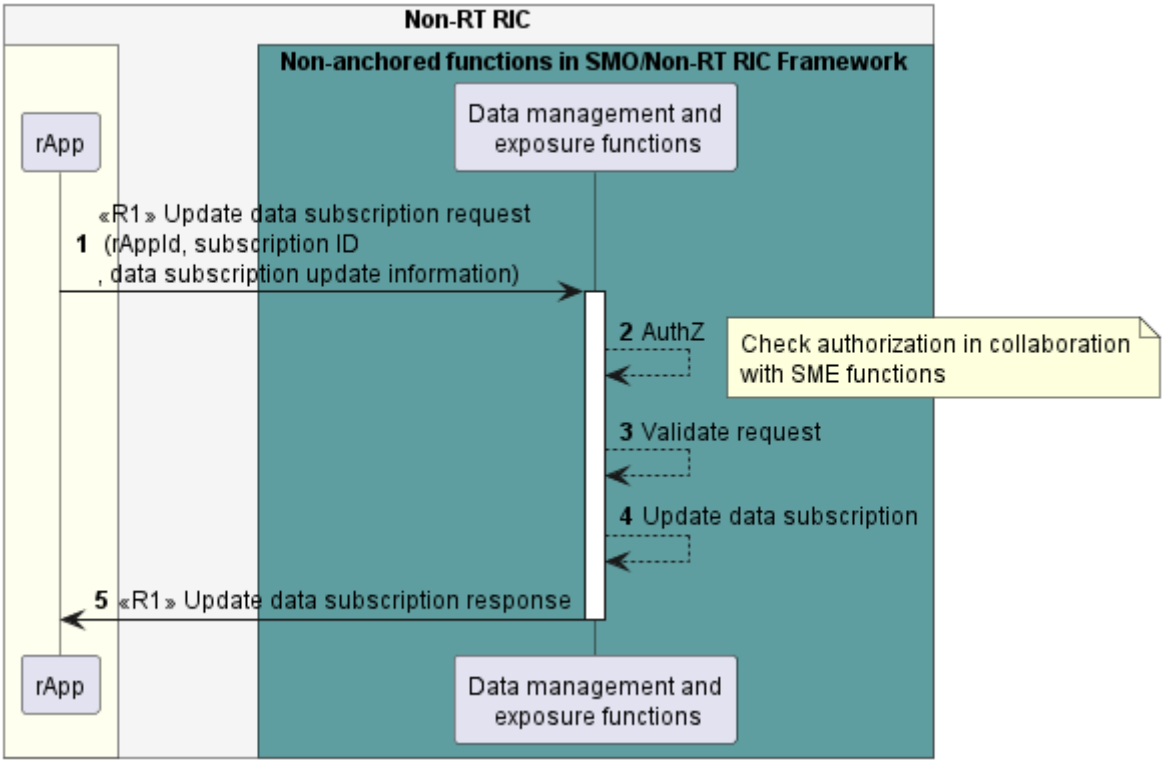

Figure 7.7.4.3-1: Update a data subscription use case flow diagram

7.7.4.4 Query a data subscription

Table 7.7.4.4-1: Query a data subscription

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	rApp query a 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 subscribed to data instance.	
Begins when	The rApp determines to query a data subscription.	
Step 1 (M)	The rApp sends the Query data subscription request with rAppId, 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 query a data subscription.	
Step 3 (M)	The Data management and exposure functions validate the request.	
Step 4 (M)	The Data management and exposure functions respond to the request with the data subscription information.	
Ends when	rApp has queried the data subscription.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-DME-Data subscription-FUN4.	

```
@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>> Query data subscription request(rAppId, subscription ID)
activate DME
DME --> DME : AuthZ
note right
Check authorization in collaboration
with SME functions
end note

DME --> DME : Validate request
rApp <- DME: <<R1>> Query data subscription response\n (data subscription information)
deactivate DME
@enduml
```

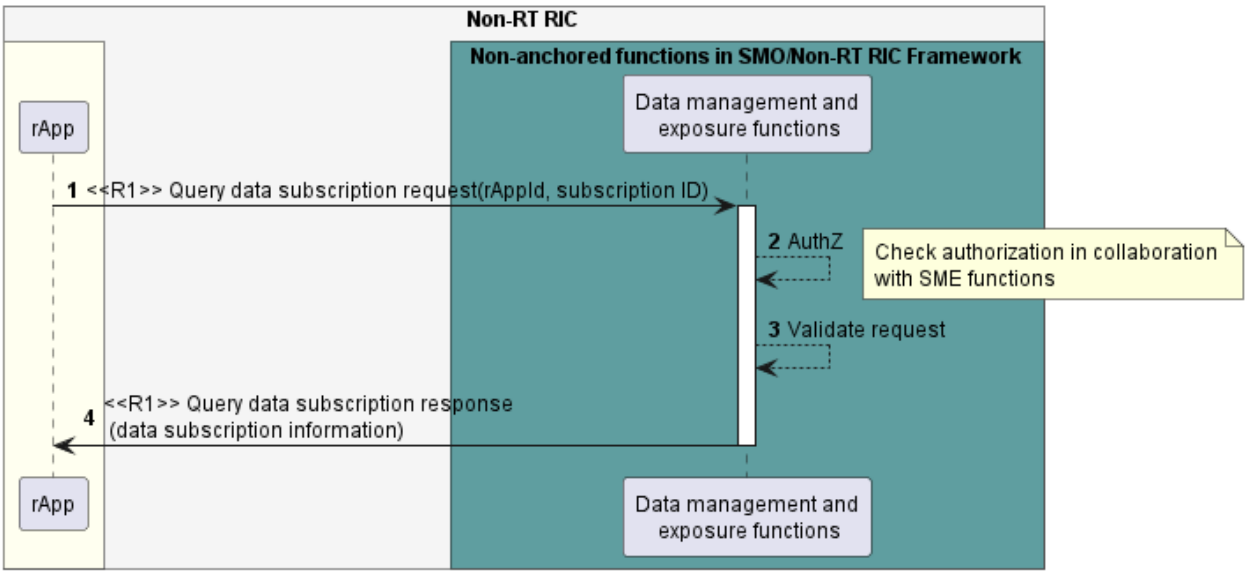

Figure 7.7.4.4-1: Query a data subscription use case flow diagram

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. If point to multipoint data stream is used as delivery method, the Data management and exposure functions provide the data stream endpoint. The rApp requests the Data management and exposure functions with rAppId and subscription identifier as parameters to terminate the data subscription.

In the update data subscription request, the rApp provides the rAppId, subscription ID and data subscription update information.

In the query data subscription request, the rApp provides the rAppId and subscription identifier. In the query data subscription response, the Data management and exposure functions provide the data subscription information.

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.

When point to multipoint data stream is used, rApp fetches the data from data stream endpoint.

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
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 "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
```

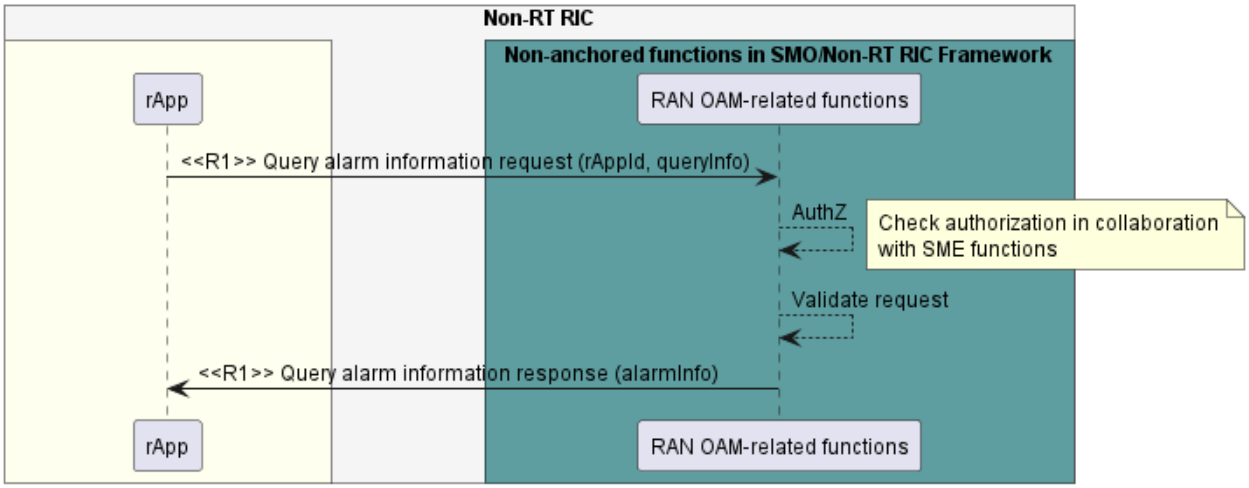

Figure 8.1.4.1-1: Query alarm information use case flow diagram

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) receive the request to change the acknowledgement state of an alarm;

   b) store the alarm acknowledgement state information;

   c) provide 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 an alarm.	
Actors and Roles	- rApp in the role of Service Consumer that requests to acknowledge an alarm. - RAN OAM-related functions in the role of Fault management service Producer that handles the request to acknowledge 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 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 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
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 "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
```

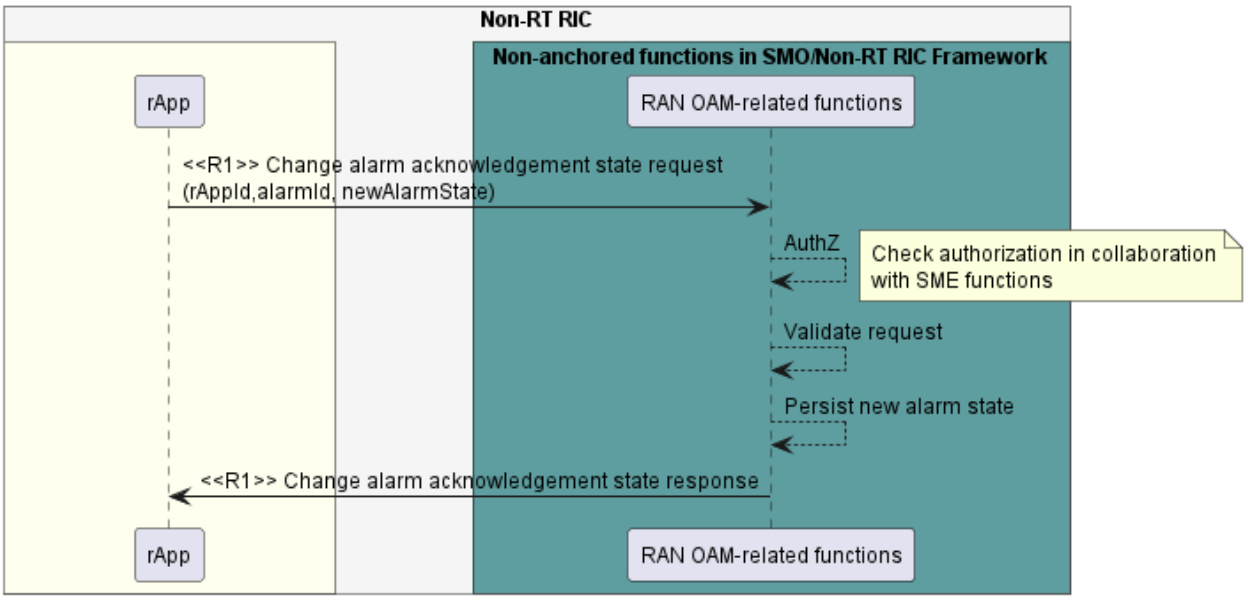

Figure 8.2.4.1-1: Change alarm acknowledgement state use case flow diagram

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: If and how the RAN OAM-related functions ensure that the alarm IDs from multiple network elements do not collide is not defined in the present document.

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.

8.3.3 Entities/resources involved in the use-case

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

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

   b) provide 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 information 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
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 "RAN OAM-related functions" as fmsp
  endbox
endbox
rapp -> fmsp: <<R1>> Query performance information request(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 response(performanceInformation)
@enduml
```

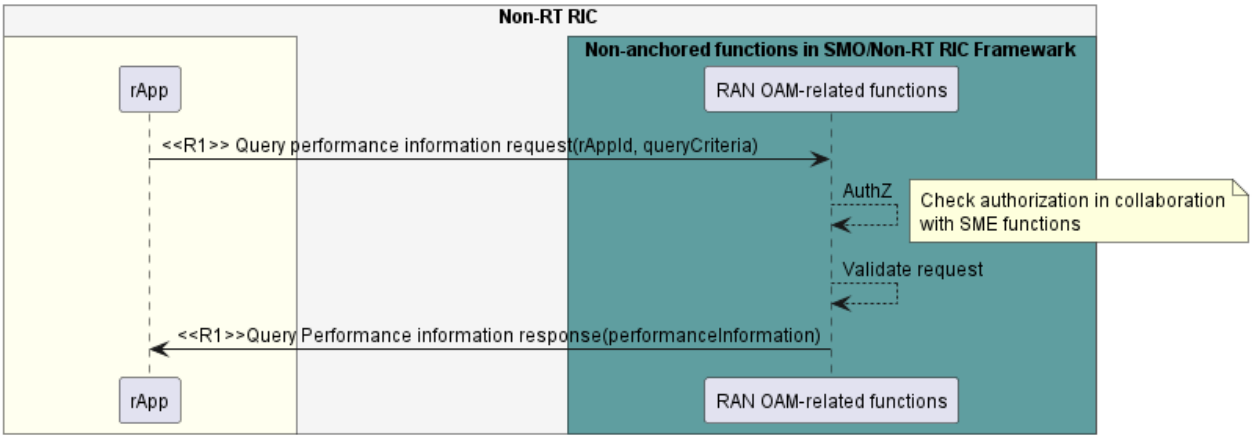

Figure 8.3.4.1-1: Query performance information use case flow diagram

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 not defined in the present document.

8.4 RAN OAM-Related use case 4: Retrieve configuration schemas

8.4.1 Overview

This use-case allows an rApp acting as a CM service Consumer to retrieve information pertaining to the configuration schemas of one or more managed entities.

8.4.2 Background and goal of the use-case

An rApp acting as a CM service Consumer can retrieve information pertaining to one or more managed entities from the Configuration management service Producer.

8.4.3 Entities/resources involved in the use-case

1) RAN OAM-related functions as Configuration management service Producer:

   a) receive the Get schemas request to retrieve configuration schema information related to one or more managed entities;

   b) provide the response of success or failure result to the Get schema request.

2) rApp:

   a) supports functionality to initiate Get schemas procedure to retrieve the configuration schema information.

8.4.4 Solutions

8.4.4.1 Retrieve configuration schema information

Table 8.4.4.1-1: Retrieve configuration schema information use-case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp obtains configuration schema information from the Configuration management service Producer.	
Actors and Roles	- rApp in the role of CM service Consumer that request the configuration schema information pertaining to one or more managed entities. - RAN OAM-related functions in the role of Configuration management service Producer that provide the requested configuration schema information.	
Assumptions	n/a	
Preconditions	The rApp is deployed and authorized to request the configuration schema information.	
Begins when	The rApp determines the need to request the configuration schema information.	
Step 1 (M)	The rApp requests "Get schemas" to obtain the configuration schema information from the RAN OAM-related functions by providing the rAppId, optional query criteria and information about the managed entities, along with the desired configuration schema information, that determines the requested result set.	
Step 2 (M)	The RAN OAM-related functions check whether the rApp is authorized to request Get Schemas to retrieve configuration schema information.	
Step 3 (M)	The RAN OAM-related functions validate the information provided with Get Schema request, 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 configuration schema information.	
Ends when	The rApp was able to receive the configuration schema information.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-OAM-CmService-FUN1.	

```
@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 "RAN OAM-related functions" as cmsp
  endbox
endbox
rapp -> cmsp: <<R1>> Get schemas request(rAppId, queryCriteria)
cmsp --> cmsp: AuthZ
note right
Check authorization in
Collaboration with SME functions
end note
cmsp --> cmsp: Validate request
cmsp -> rapp: <<R1>>Get schemas response (configurationSchema)
@enduml
```

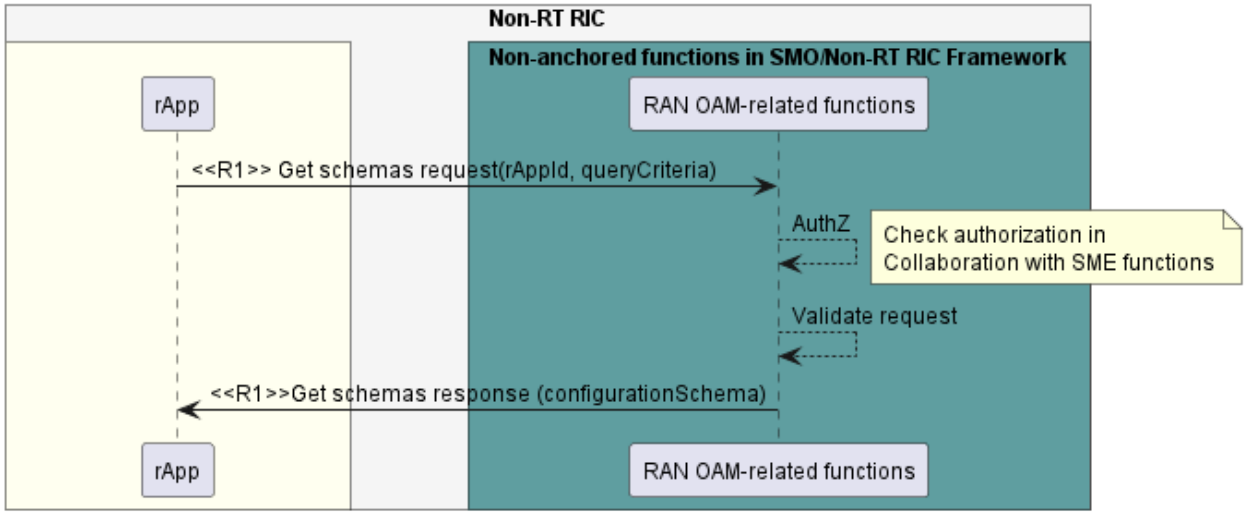

Figure 8.4.4.1-1: Retrieve configuration schema information use case flow diagram

8.4.5 Required data

The Get schemas request for retrieval of configuration schemas information contains the rAppId, and query criteria (including information about the related managed entities and about the requested configuration schemas information). The Get schemas response is scoped by the information provided in the request.

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.

8.5 RAN OAM-Related use case 5: Read configuration data

8.5.1 Overview

This use-case allows an rApp acting as a CM service Consumer to retrieve information pertaining to the configuration data of one or more managed entities.

8.5.2 Background and goal of the use-case

An rApp acting as a CM service Consumer can retrieve configuration data pertaining to one or more managed entities from the Configuration management service Producer.

8.5.3 Entities/resources involved in the use-case

1) RAN OAM-related functions as Configuration management service Producer:

   a) receives the request to read configuration data related to one or more managed entities;

   b) provides the response of success or failure result to the read configuration data request.

2) rApp:

   a) supports functionality to initiate procedure to read configuration data.

8.5.4 Solutions

8.5.4.1 Read configuration data

Table 8.5.4.1-1: Read configuration data use-case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp obtains configuration data from the Configuration management service Producer.	
Actors and Roles	- rApp in the role of Service Consumer that request the configuration data pertaining to one or more managed entities. - RAN OAM-related functions in the role of Configuration Management service Producer that provide the requested configuration data information.	
Assumptions	rApp can optionally query configuration schema before requesting to read configuration data.	
Preconditions	The rApp is deployed and authorized to request the configuration data.	
Begins when	The rApp determines the need to request the configuration data.	
Step 1 (M)	The rApp requests "Read configuration" to obtain the configuration data from the RAN OAM-related functions by providing the rAppId, optional query criteria and information about the managed entities, along with information about the desired configuration data, that determines the requested result set.	
Step 2 (M)	The RAN OAM-related functions check whether the rApp is authorized to Read configuration data.	
Step 3 (M)	The RAN OAM-related functions validate information provided with the Read configuration request.	
Step 4 (M)	The RAN OAM-related functions respond to the rApp with a success result along with the desired configuration data.	
Ends when	The rApp was able to receive the configuration data.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-OAM-CmService-FUN2.	

```
@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 "RAN OAM-related functions" as cmsp
  endbox
endbox
rapp -> cmsp: <<R1>> Read configuration request(rAppId, queryCriteria)
cmsp --> cmsp: AuthZ
note right
Check authorization in Collaboration
with SME functions
end note
cmsp --> cmsp: Validate request
cmsp -> rapp: <<R1>> Read configuration response (configurationData)
@enduml
```

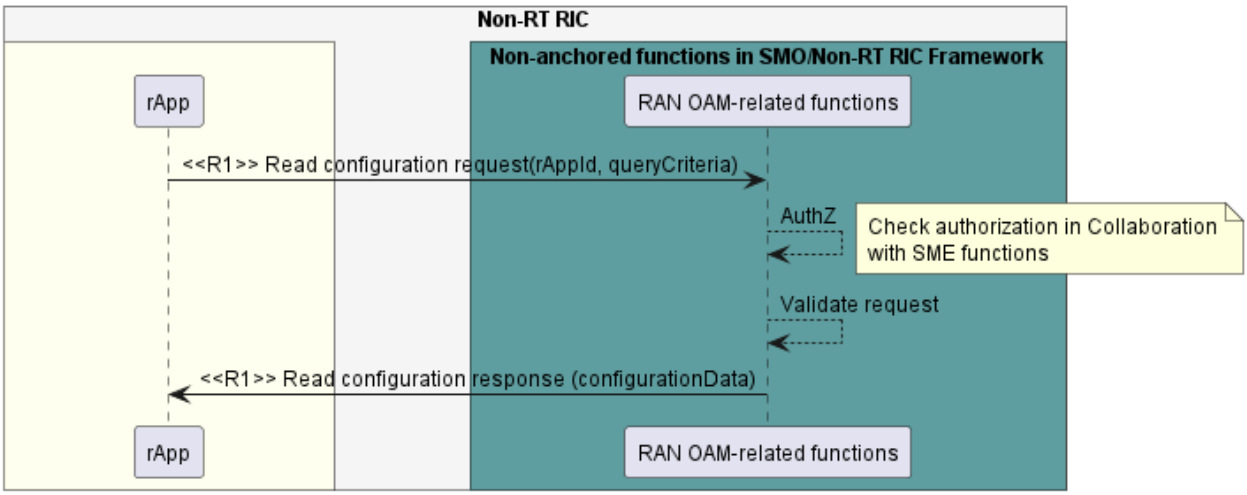

Figure 8.5.4.1-1: Read configuration data use case flow diagram

8.5.5 Required data

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

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.

8.6 RAN OAM-Related use case 6: Writing configuration changes

8.6.1 Overview

This use-case allows an rApp acting as a CM service Consumer to write information pertaining to the configuration changes of one or more managed entities.

8.6.2 Background and goal of the use-case

An rApp acting as a CM service Consumer can write information pertaining to the configuration changes for one or more managed entities through the CM service Producer.

8.6.3 Entities/resources involved in the use-case

1) RAN OAM-related functions as CM service Producer:

   a) receives the Write configuration changes request to write configuration change information related to one or more managed entities;

   b) provides the response of success or failure result to the Service Consumer that initiated the Write configuration request.

2) rApp:

   a) supports functionality to initiate Write configuration changes request to write the configuration change information.

8.6.4 Solutions

8.6.4.1 Write configuration changes

Table 8.6.4.1-1: Write Configuration changes use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use case
Goal	The rApp writes configuration change information to the CM service Producer.	
Actors and Roles	- rApp in the role of CM service Consumer that requests writing the configuration change information pertaining to one or more managed entities. - RAN OAM-related functions in the role of CM service Producer that initiate provisioning the requested configuration changes.	
Assumptions	rApp optionally query configuration schema before preparing the write configuration change information.	
Preconditions	The rApp is deployed and authorized to write the configuration change information. Optionally the rApp is subscribed to receive information related to write configuration change status from RAN OAM-related function through Notification or polling.	
Begins when	The rApp determines the need to write the configuration change information.	
Step 1 (M)	The rApp sends a Write configuration changes request to write the configuration change information to the RAN OAM-related functions by providing the rAppId, along with the desired configuration changes information for one or more managed entities.	
Step 2 (M)	The RAN OAM-related functions check whether the rApp is authorized to initiate the Write configuration changes request.	
Step 3 (M)	The RAN OAM-related functions validate the information provided in the Write configuration changes request.	
Step 4 (M)	The RAN OAM-related functions create a Write configuration job from the information provided in the Write configuration changes request.	
Step 5 (M)	RAN OAM-related functions respond to the rApp with information about the job created such as job identifier. The rApp can use this information for querying or receiving notification about the status / result of the requested configuration change.	
Ends when	The rApp was able to receive the Write configuration response.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-OAM-CmService-FUN3.	

```
@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 "RAN OAM-related functions" as cmsp
  endbox
endbox
rapp -> cmsp: <<R1>> Write configuration changes request\n(rAppId, configuration changes
information)
cmsp --> cmsp: AuthZ
note right
Check authorization in
Collaboration with SME functions
end note
cmsp --> cmsp: Validate request
cmsp --> cmsp: Create job
```

```
cmsp -> rapp: <<R1>> Write configuration changes response\n (Job information)
@enduml
```

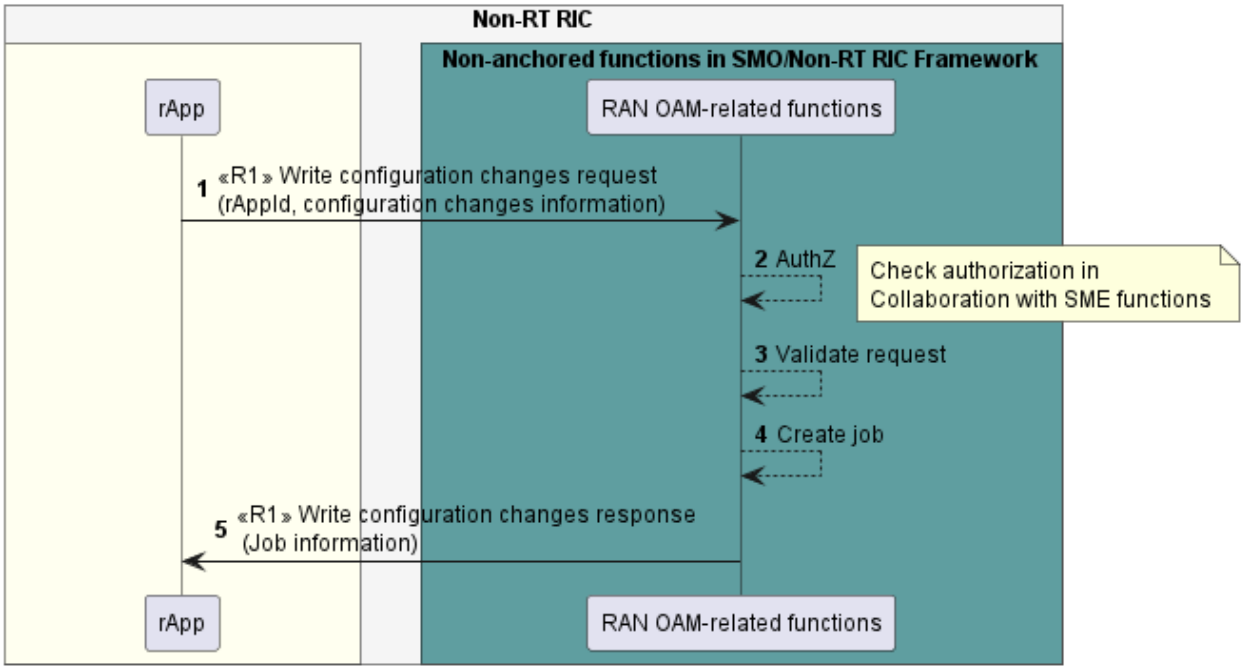

Figure 8.6.4.1-1: Write configuration changes use case flow diagram

8.6.5 Required data

The request for writing configuration changes information contains the rAppId and write configuration information for one or more managed entities (e.g. list of attributes and desired values).

The RAN OAM-related functions respond to the rApp with configuration job information created, e.g. job identifier. The rApp can use this job information for querying or receiving notifications about the status / result of the requested configuration change which includes e.g.:

- a success result if all the requested configuration changes were written in the related managed entities; or
- a partial success result if some but not all requested configuration changes were written in the related mentioned managed entities, optionally with further information about the reason of the errors; or
- a failure result if one of the requested configuration changes were written in the related managed entities, optionally with further information about the reason of the errors.

9 Use cases for O2-Related Services

NOTE: Use cases for O2-Related services are not defined in the present document.

10 Use cases for A1-Related Services

10.1 Void

10.2 Void

10.3 A1-Related use case 1: Query A1 policy type identifiers

10.3.1 Overview

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

10.3.2 Background and goal of the use case

Query A1 policy type identifiers procedure is defined as part of A1 policy management service in R1GAP [1].

10.3.3 Entities/resources involved in the use case

1) A1 Policy functions:
   a) support functionality to allow rApps to query the A1 policy type identifiers.
2) rApp:
   a) initiates the Query A1 policy type identifiers procedure.

10.3.4 Solutions

10.3.4.1 Query A1 policy type identifiers

Table 10.3.4.1-1: Query A1 policy type identifiers

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves the list of identifiers for all the available A1 policy types.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy functions in the role of A1 policy management service Producer.	
Assumptions	rApp is authorized to consume the A1 policy management service.	
Preconditions	n/a	
Begins when	The rApp determines the need to query the identifiers for all the available A1 policy types.	
Step 1 (M)	The rApp queries the available A1 policy type identifiers, providing the rAppId and optional selection criteria (Near-RT RIC identifier and/or typename (see 3 in A1AP [5], clause 6.2.3.1)).	
Step 2 (M)	The A1 policy management functions validate if the rApp is authorized to query A1 policy type identifiers.	
Step 3 (M)	The A1 policy management functions respond with the list of identifiers for all the available A1 policy types that matches with the query criteria. Further, the response includes for each A1 policy type identifier a list of Near-RT RIC identifiers of those Near-RT RICs that support the related A1 policy type.	
Ends when	The rApp has the identifiers of all the available A1 policy types.	
Exceptions	n/a	
Post Conditions	The rApp has the identifiers of all the available A1 policy types.	
Traceability	REQ-R1-A1P-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 "A1 policy functions " as A1P
            endbox
    rApp -> A1P : <<R1>> Query A1 policy type identifiers request (rAppId)
    activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
               with SME functions
    end note
     A1P -> rApp : <<R1>> Query A1 policy type identifiers response (list of A1 policy type
identifiers)
    Deactivate A1P
@enduml
```

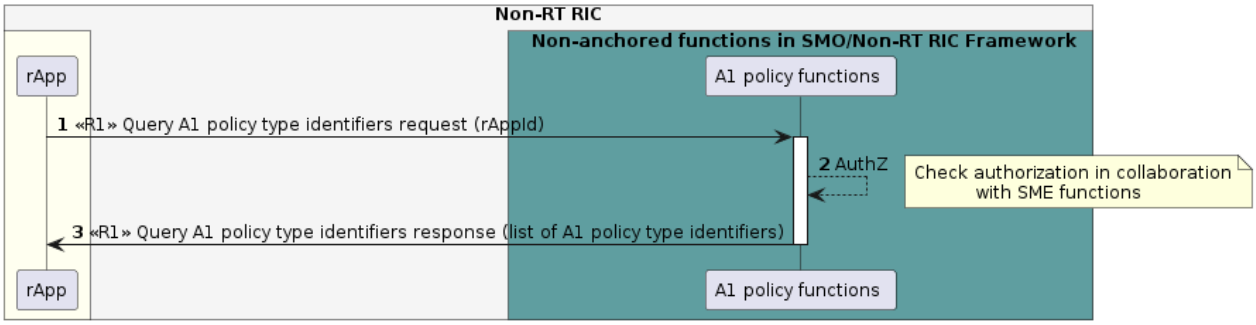

Figure 10.3.4.1-1: Query A1 policy type identifiers use case flow diagram

10.3.5 Required data

For querying the A1 policy type identifiers, the rApp sends a query request to the A1 policy functions and optionally provides a Near-RT RIC identifier and/or a typename (see in A1AP [5], clause 6.2.3.1.3) as a selection criterion.

If the rApp queries with no selection criteria, the A1 policy functions respond with a list of A1 policy type identifiers. Further, the response includes for each A1 policy type identifier a list of Near-RT RIC identifiers of those Near-RT RICs that support the related A1 policy type.

If the rApp queries with selection criteria, the A1 policy functions respond with a list of A1 policy type identifiers which match with the query criteria. Further, the response includes for each A1 policy type identifier a list of Near-RT RIC identifiers of those Near-RT RICs that support the related A1 policy type.

10.4 A1-Related use case 2: Query an A1 policy type

10.4.1 Overview

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

10.4.2 Background and goal of the use case

Query A1 policy type procedure is defined as part of A1 policy management service in R1GAP [1].

10.4.3 Entities/resources involved in the use case

1) A1 Policy functions:

   a) support functionality to allow rApps to query the information of an A1 policy type.

2) rApp:

   a) initiates the Query A1 policy type procedure.

10.4.4 Solutions

10.4.4.1 Query A1 policy type

Table 10.4.4.1-1: Query A1 policy type

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves information about A1 policy type.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy 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 A1 policy type identifier.	
Begins when	The rApp determines the need to query the information of an A1 policy type.	
Step 1 (M)	The rApp queries the information of an A1 policy type with the A1 policy functions by providing rAppId and an A1 policy type identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to access the information of an A1 policy type.	
Step 3 (M)	The A1 policy functions retrieve the policy type information.	
Step 4 (M)	The A1 policy functions respond with the requested A1 policy type information.	
Ends when	The rApp have received the information about the requested A1 policy type.	
Exceptions	n/a	
Post Conditions	The rApp have the information about the requested A1 policy type.	
Traceability	REQ-R1-A1P-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 "A1 policy functions " as A1P
            endbox
    rApp -> A1P : <<R1>> Query A1 policy type request (rAppId, A1 policy type identifier)
    activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
                with SME functions
    end note
A1P --> A1P: retrieve policy type information
     A1P -> rApp : <<R1>> Query A1 policy type response (A1 policy type information)
    Deactivate A1P
@enduml
```

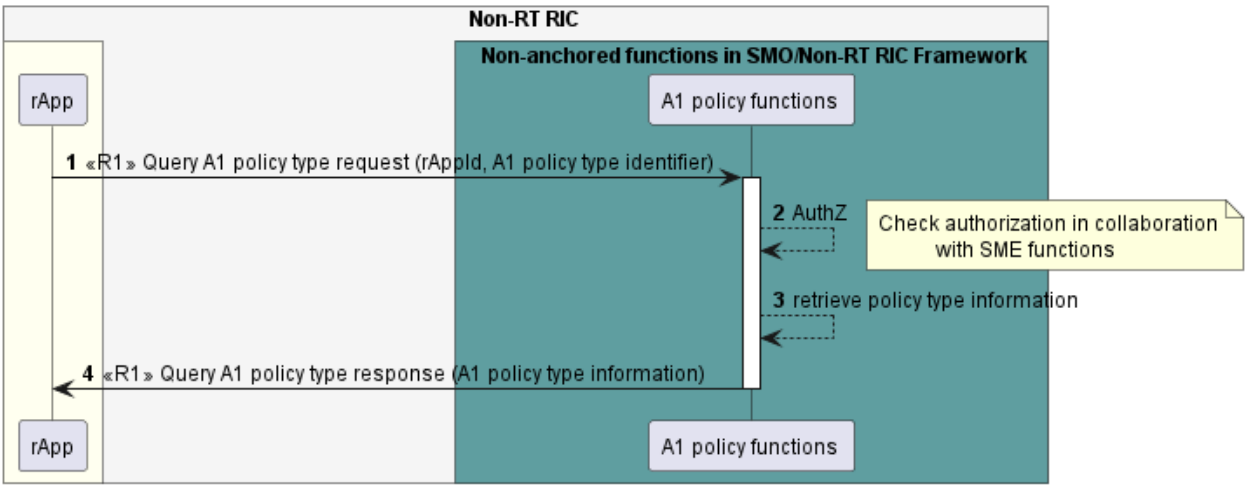

Figure 10.4.4.1-1: Query A1 policy type use case flow diagram

10.4.5 Required data

For querying an A1 policy type, the rApp provides the rAppId and an A1 policy type identifier. The A1 policy functions respond with the information about the A1 policy type.

10.5 A1-Related use case 3: Query A1 policy identifiers

10.5.1 Overview

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

10.5.2 Background and goal of the use case

Query A1 policy identifiers procedure is defined as part of A1 policy management service in R1GAP [1].

10.5.3 Entities/resources involved in the use case

1) A1 Policy functions:

   a) support functionality to allow rApps to query the A1 policy identifiers.

2) rApp:

   a) initiates the Query A1 policy identifiers procedure.

10.5.4 Solutions

10.5.4.1 Query A1 policy identifiers

Table 10.5.4.1-1: Query A1 policy identifiers

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves the existing A1 policy identifiers.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy functions in the role of A1 policy management service Producer.	
Assumptions	rApp is authorized to consume the A1 policy management service.	
Preconditions	n/a	
Begins when	The rApp determines the need to query the A1 policy identifiers.	
Step 1 (M)	The rApp queries the A1 policy identifiers with the A1 policy functions and may provide an A1 policy type identifier and/or a Near RT RIC identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to query A1 policy identifiers.	
Step 3 (M)	The A1 policy functions respond with the list of A1 policy identifiers that matches with the query criteria.	
Ends when	The rApp has the list of A1 policy identifiers.	
Exceptions	n/a	
Post Conditions	The rApp has the list of A1 policy identifiers.	
Traceability	REQ-R1-A1P-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 "A1 policy functions " as A1P
            endbox
    rApp -> A1P : <<R1>> Query A1 policy identifers request (rAppId)
    activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
               with SME functions
    end note
     A1P -> rApp : <<R1>> Query A1 policy identifers response (list of A1 policy identifers)
    Deactivate A1P
@enduml
```

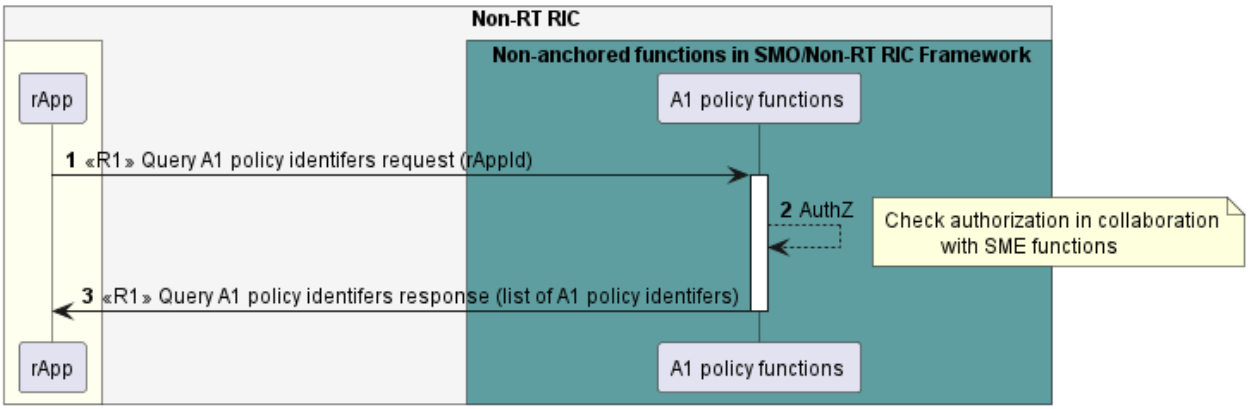

Figure 10.5.4.1-1: Query A1 policy identifiers use case flow diagram

10.5.5 Required data

For querying the A1 policy identifiers, the rApp optionally provides an A1 policy type identifier and/or a Near RT RIC identifier as a query parameter. The A1 policy functions respond with the list of A1 policy identifiers.

10.6 A1-related use case 4: Create, Query, Update and Delete an A1 policy

10.6.1 Overview

This use case enables rApps to create an A1 policy, query an A1 policy, update an A1 policy, and delete an A1 policy.

10.6.2 Background and goal of the use case

Create A1 policy, Query A1 policy, Update A1 policy, and Delete A1 policy procedures are defined as part of A1 policy management service in R1GAP [1].

10.6.3 Entities/resources involved in the use case

1) A1 policy functions:

   a) support functionality to allow rApps to request creating7, querying, updating and deleting an A1 policy.

2) rApp:

   a) initiates the procedures for creating an A1 policy, querying an A1 policy, updating an A1 policy, and deleting an A1 policy.

10.6.4 Solutions

10.6.4.1 Create a single A1 policy in an identified Near-RT RIC

Table 10.6.4.1-1: Create a single A1 policy in an identified Near-RT RIC

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp initiates creation of an A1 policy in an identified Near-RT RIC.	
Actors and Roles	- rApp in the role of the A1 policy management service Consumer. - A1 policy functions in the role of the A1 policy management service Producer.	
Assumptions	rApp is authorized to consume the A1 policy management service.	
Preconditions	The rApp is aware about the A1 policy type.	
Begins when	The rApp determines to create a new A1 policy.	
Step 1 (M)	The rApp requests the A1 policy functions to create a new A1 policy of certain type by providing a policy type identifier, policy information, rApp identifier, and a Near-RT RIC identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to create A1 policies.	
Step 3 (M)	The A1 policy functions generate the A1 policy identifier.	
Step 4 (M)	The A1 policy functions create a single A1 policy in the Near-RT RIC corresponding to the given Near-RT RIC identifier.	See A1UCR [6], clause 6.3
Step 5 (M)	The A1 policy functions store the mapping of A1 policy identifier to the Near-RT RIC identifier.	
Step 6 (M)	The A1 policy functions respond to rApp with the A1 policy identifier of the created A1 policy.	
Ends when	The rApp has created a new A1 policy.	
Exceptions	n/a	
Post Conditions	An A1 policy exists and the rApp will be able to query, update and delete the A1 policy.	
Traceability	REQ-R1-A1P-FUN9.	

```
@startuml
 'https://plantuml.com/sequence-diagram'
    !pragma teoz true
        skinparam ParticipantPadding 5
        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 "A1 policy functions " as A1P
            endbox
    rApp -> A1P :1 <<R1>> Create A1 policy request\n (policy type identifier, policy information,
Near-RT RIC identifier, rAppId)
    activate A1P
    A1P --> A1P :2 AuthZ
    note right
    Check authorization in collaboration
                with SME functions
    end note
A1P --> A1P : 3 Generate A1 policy identifier

ref over A1P
    4 create single policy(see A1 UCR clause 6.3)
end ref

A1P --> A1P :5 Store A1 policy identifier

     A1P -> rApp : 6 <<R1>> Create A1 policy response (policy identifier)
    Deactivate A1P
@enduml
```

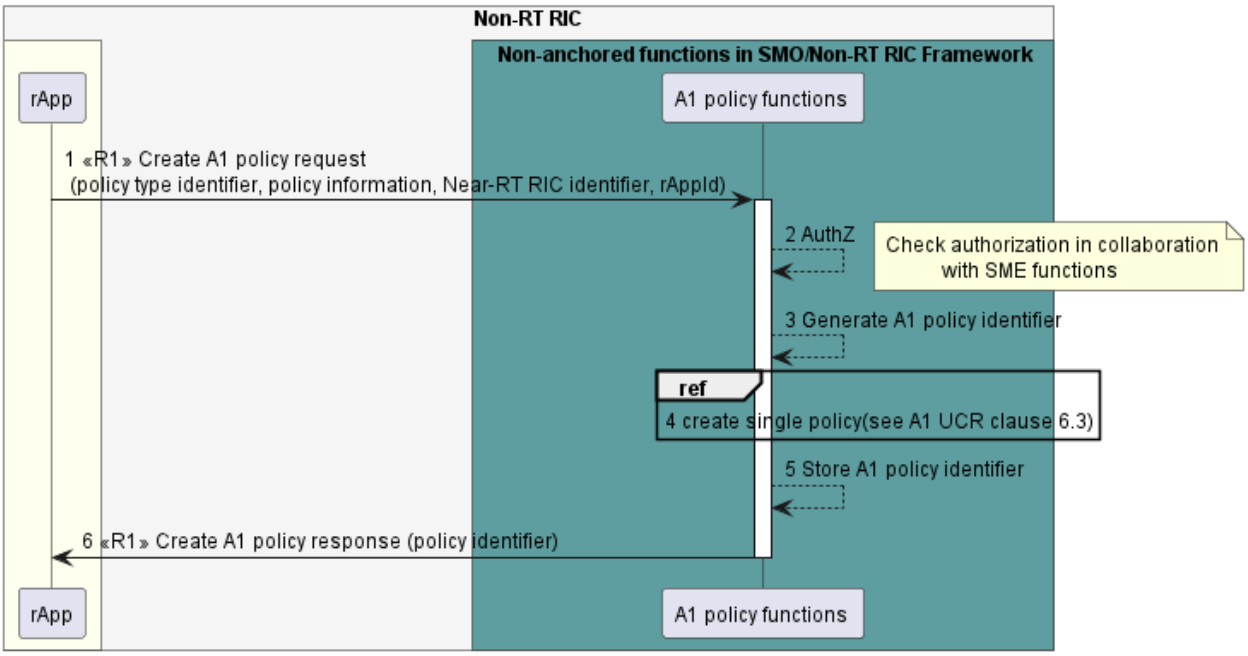

Figure 10.6.4.1-1: Create a single A1 policy in an identified Near-RT RIC flow diagram

10.6.4.2 Query an A1 policy

Table 10.6.4.2-1: Query an A1 policy

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp queries an existing A1 policy.	
Actors and Roles	- rApp in the role of an A1 policy management service Consumer. - A1 policy functions in the role of an A1 policy management service Producer.	
Assumptions	The rApp is authorized to consume the A1 policy management service.	
Preconditions	- The rApp is aware about the A1 policy identifier of the A1 policy that it wants to query. - A1 policy functions are aware of the Near-RT RIC identifier corresponding to this A1 policy identifier so that it can query the A1 policy at the relevant Near-RT RIC instance.	
Begins when	The rApp determines to query an existing A1 policy.	
Step 1 (M)	The rApp requests the A1 policy functions to query an existing A1 policy by providing the A1 policy identifier, and its rApp identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to query an A1 policy.	
Step 3 (M)	A1 policy functions respond with the result of the A1 policy query including policy information.	See note
Ends when	The rApp has the A1 policy information.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-A1P-FUN9.	
NOTE: The A1 policy functions may use A1UCR [6], clause 6.4.3.2 Query single policy to retrieve the A1 policy information from the relevant Near-RT RIC whenever an rApp queries an A1 policy. It may also choose to respond to the rApp query with the latest policy information previously received from the relevant Near-RT RIC.		

```
@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-RT RIC FWK/SMO FWK" #cadetBlue
           participant "A1 policy functions " as A1P
rApp -> A1P :<<R1>>Query  A1 policy request (A1 policy identifier, \n rAppId)
activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
               with SME functions
    end note
A1P -> rApp :<<R1>>Query A1 policy response (policy information)

    Deactivate A1P

@enduml
```

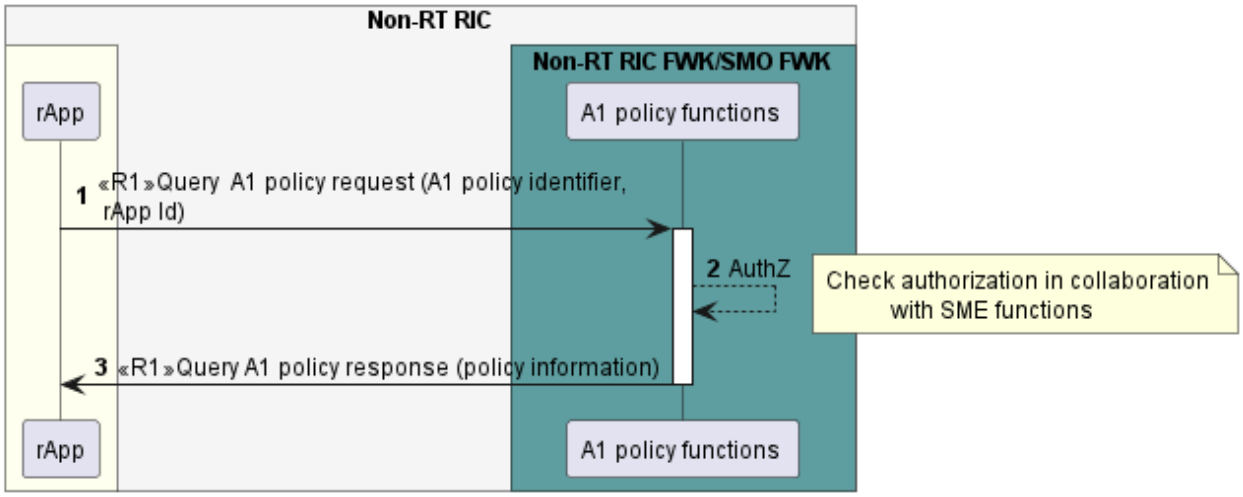

Figure 10.6.4.2-1: Query A1 policy use case flow diagram

10.6.4.3 Update an A1 policy

Table 10.6.4.3-1: Update an A1 policy

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp updates an existing A1 policy.	
Actors and Roles	- rApp in the role of an A1 policy management service Consumer. - A1 policy functions in the role of an A1 policy management service Producer.	
Assumptions	The rApp is authorized to consume the A1 policy management service.	
Preconditions	- The rApp is aware of the identifier of the A1 policy that it wants to update. - A1 policy functions are aware of the existing policy to be updated was created by the rApp. - A1 policy functions are aware of the Near-RT RIC identifier corresponding to this A1 policy identifier so that it can update the A1 policy at the relevant Near-RT RIC instance.	
Begins when	The rApp determines to update an existing A1 policy.	
Step 1 (M)	The rApp requests the A1 policy functions to update an existing A1 policy by providing the A1 policy identifier, policy update information, rApp identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to update an A1 policy.	
Step 3 (M)	The A1 policy functions update a single A1 policy in the Near-RT RIC.	See A1UCR [6], clause 6.5
Step 4 (M)	The A1 policy functions respond with the result of the A1 policy update.	
Ends when	The rApp has updated the A1 policy.	
Exceptions	n/a	
Post Conditions	The content of the A1 policy has changed.	
Traceability	REQ-R1-A1P-FUN9.	

```
@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-RT RIC FWK/SMO FWK" #cadetBlue
           participant "A1 policy functions " as A1P
rApp -> A1P :<<R1>>Update A1 policy request (A1 policy identifier, \n policy information,  rApp Id)
activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
               with SME functions
    end note
ref over A1P
  3 Update single policy(see A1 UCR clause 6.5)
end ref
autonumber 4
A1P -> rApp :<<R1>>Update A1 policy response
deactivate A1P
@enduml
```

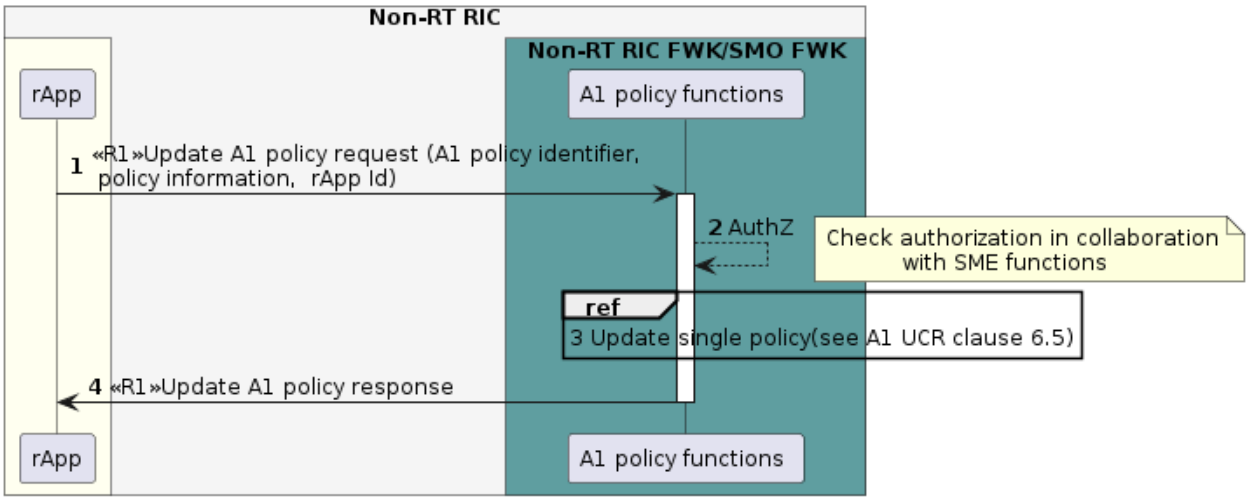

Figure 10.6.4.3-1: Update A1 policy use case flow diagram

10.6.4.4 Delete an A1 policy

Table 10.6.4.4-1: Delete an A1 policy

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp deletes an existing A1 policy.	
Actors and Roles	- rApp in the role of an A1 policy management service Consumer. - A1 policy functions in the role of an A1 policy management service Producer.	
Assumptions	The rApp is authorized to consume A1 policy management service.	
Preconditions	- The rApp is aware of the identifier of the A1 policy that it wants to delete. - A1 policy functions are aware of the existing policy to be deleted was created by the rApp. - A1 policy functions are aware of the Near-RT RIC identifier corresponding to this A1 policy identifier so that it can delete the A1 policy at the relevant Near-RT RIC instance.	
Begins when	The rApp determines to delete an existing A1 policy.	
Step 1 (M)	The rApp requests the A1 policy functions to delete an existing A1 policy by providing the A1 policy identifier and rApp identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to delete an A1 policy.	
Step 3 (M)	The A1 policy functions delete a single A1 policy in the Near-RT RIC.	See A1UCR [6], clause 6.6
Step 4 (M)	The A1 policy functions respond with the result of A1 policy delete.	
Ends when	The rApp has deleted an A1 policy.	
Exceptions	n/a	
Post Conditions	The A1 policy has ceased to exist.	
Traceability	REQ-R1-A1P-FUN9.	

```
@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-RT RIC FWK/SMO FWK" #cadetBlue
       participant "A1 policy functions " as A1P
        endbox
```

```
rApp -> A1P :<<R1>>Delete A1 policy request (A1 policy identifier, \n rAppId)
activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
               with SME functions
    end note
ref over A1P
  3 Delete single policy(see A1 UCR clause 6.6)
end ref
autonumber 4
A1P -> rApp :<<R1>>Delete A1 policy response
Deactivate A1P
@enduml
```

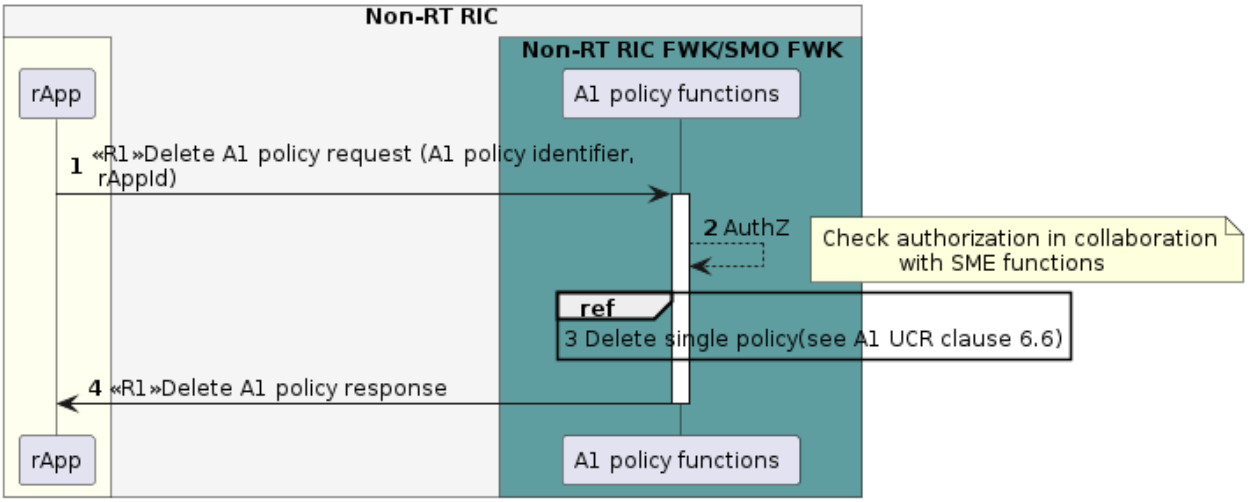

Figure 10.6.4.4-1: Delete A1 policy use case flow diagram.

10.6.5 Required data

For creating an A1 policy of a certain type in a Near-RT RIC, the rApp provides an A1 policy type identifier, A1 policy information, its rApp identifier (rAppId) and, a Near-RT RIC identifier. The A1 policy functions respond with the A1 policy identifier of the created policy.

For querying an A1 policy, the rApp provides an A1 policy identifier, and its rApp identifier (rAppId). The A1 policy functions respond with A1 policy information.

For updating an A1 policy, the rApp provides an A1 policy identifier, A1 policy updated information, and its rApp identifier (rAppId).

For deleting an A1 policy, the rApp provides an A1 policy identifier and its rApp identifier (rAppId).

10.7 A1-Related use case 5: Query the status of an A1 policy

10.7.1 Overview

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

10.7.2 Background and goal of the use case

Query A1 policy status procedure is defined as part of A1 policy management service in R1GAP [1].

10.7.3 Entities/resources involved in the use case

1) A1 policy 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.

10.7.4 Solutions

10.7.4.1 Query the status of an A1 policy

Table 10.7.4.1-1: 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 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 functions by providing an rAppId and A1 policy identifier.	
Step 2 (M)	The A1 policy functions validate if the rApp is authorized to query the status of an A1 policy.	
Step 3 (M)	The A1 policy functions respond with the status of the corresponding A1 policy.	See note
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	REQ-R1-A1P-FUN5.	
NOTE: The A1 policy functions may use A1UCR [6], clause 6.7.3.1 Query policy status, or A1UCR [6], clause 6.7.3.2 Notify policy status change to respond with the latest status of the corresponding A1 policy.		

```
@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 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
```

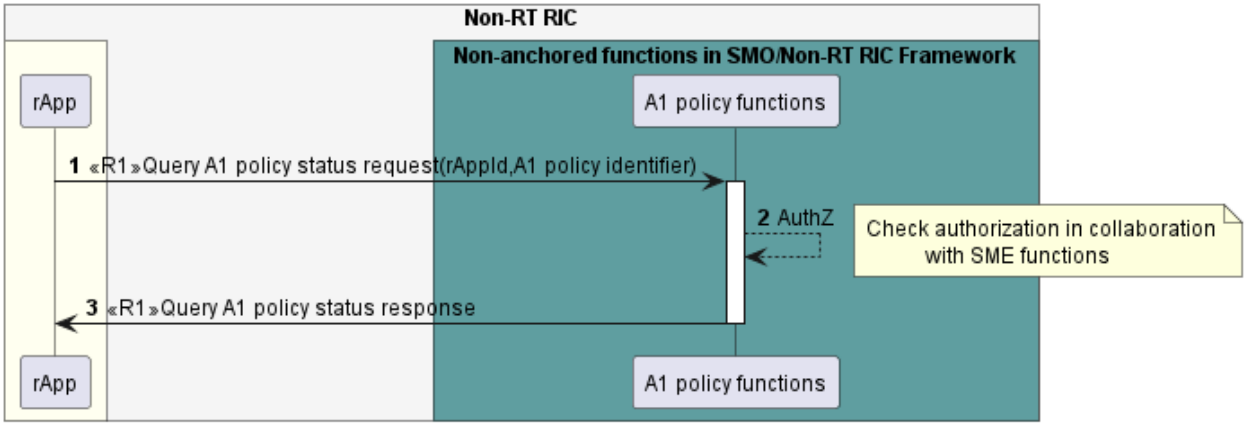

Figure 10.7.4.1-1: Query status of A1 policy use case flow diagram

10.7.5 Required data

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

The A1 policy functions respond with the status of the A1 policy. The status information of A1 policy is defined in A1 TD [3].

10.8 A1-Related use case 6: Subscribe to status of an A1 policy

10.8.1 Overview

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

10.8.2 Background and goal of the use case

Subscribe to A1 policy status changes procedure is defined as part of A1 policy management service in R1GAP [1].

10.8.3 Entities/resources involved in the use case

1) A1 Policy functions in the role of A1 policy management service Producer:
   a) support functionality to allow rApps to subscribe to and unsubscribe from A1 policy status changes;
   b) support notifying the rApp regarding the changes in the status of an A1 policy;
   c) provide response of success or failure to subscribe to A1 policy status changes and unsubscribe to A1 policy status changes.
2) rApp:
   a) supports functionality to initiate requests to subscribe to and unsubscribe from A1 policy status changes;
   b) management service Producer.

10.8.4 Solutions

10.8.4.1 Subscribe to A1 policy status change

Table 10.8.4.1-1: subscribe to A1 policy status change

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp subscribes to notifications about changes of the status of an A1 policy.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy functions in the role of A1 policy management service Producer.	
Assumptions	rApp is aware of the A1 policy identifier of the A1 policy that it is interested in subscribing to status change notifications for.	
Preconditions	rApp is authorized to consume the A1 policy management service.	
Begins when	The rApp determines the need to subscribe to notifications, regarding changes in the status of an A1 policy.	
Step 1 (M)	The rApp request the A1 policy functions to subscribe to notifications regarding changes in the status of an A1 policy by providing the rAppId and A1 policy identifier.	
Step 2 (M)	The A1 policy functions check whether the rApp is authorized to send a subscription request.	
Step 3 (M)	The A1 policy functions respond with subscription identifier.	
Ends when	The rApp was able to subscribe to notifications.	
Exceptions	n/a	
Post Conditions	The rApp can receive notifications when there are any changes in the status of the A1 policy. The rApp can unsubscribe from status change notifications of the A1 policy.	
Traceability	REQ-R1-A1P-FUN6.	

```
@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 functions "as A1P
            endbox
    rApp -> A1P : <<R1>> subscribe A1 policy status change request\n(rAppId, A1 policy identifier)
    activate A1P
    A1P --> A1P : AuthZ
    note right
    Check authorization in collaboration
                with SME functions
    end note
     A1P -> rApp : <<R1>> subscribe A1 policy status change response (subscription identifier)
    Deactivate A1P
@enduml
```

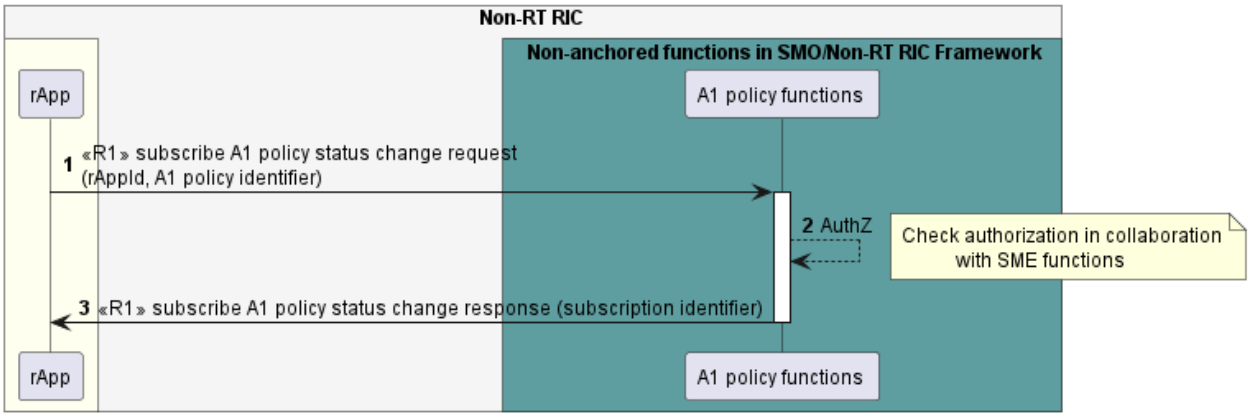

Figure 10.8.4.1-1: Subscribe to A1 policy status change use case flow diagram

10.8.4.2 Notify A1 policy status change

Table 10.8.4.2-1: Notify A1 policy status change use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp receives notifications about changes in the status of an A1 policy.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy functions in the role of A1 policy management service Producer.	
Assumptions	n/a	
Preconditions	The rApp has subscribed to notifications on changes in the status of an A1 policy.	
Begins when	The A1 policy functions determine to send notifications, regarding changes in the status of the A1 policy to the subscribed rApp.	
Step 1 (M)	The A1 policy functions detect changes in status (i.e. an A1 policy status becomes enforced or not enforced).	
Step 2 (M)	The A1 policy functions send notifications regarding the changes to the subscribed rApp with subscription identifier and information about status changes of the A1 policy.	
Ends when	The rApp was able to process the changes in the status of the A1 policy.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-A1P-FUN7.	

```
@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 functions " as A1P
endbox
autonumber
A1P→A1P: detect the status change
A1P-> rApp : <<R1>> notify changes \n(subscription identifier)
@enduml
```

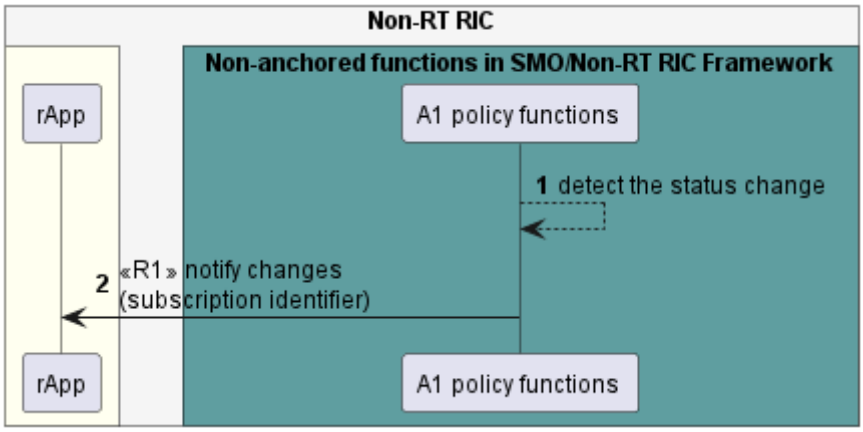

Figure 10.8.4.2-1: Notify status change use case flow diagram

10.8.4.3 Unsubscribe from A1 policy status change

Table 10.8.4.3-1: Unsubscribe from A1 policy status change

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp unsubscribes from A1 policy status change notifications.	
Actors and Roles	- rApp in the role of A1 policy management service Consumer. - A1 policy functions in the role of A1 policy management service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to unsubscribe from A1 policy status change notifications. - The rApp has subscribed to A1 policy status change notifications.	
Begins when	The rApp determines to unsubscribe from A1 policy status change.	
Step 1 (M)	The rApp requests the A1 policy functions to unsubscribe from A1 policy status changes with rAppId and subscription identifier.	
Step 2 (M)	The A1 policy functions check whether the rApp is authorized to unsubscribe.	
Step 3 (M)	The A1 policy functions send a response.	
Ends when	The rApp was able to unsubscribe from A1 policy status change.	
Exceptions	n/a	
Post Conditions	The rApp is not subscribed to A1 policy status change.	
Traceability	REQ-R1-A1P-FUN9.	

```
@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 functions " as A1P
 endbox
autonumber
rApp -> A1P : <<R1>> Unsubscribe from A1 policy status change request(rAppId,subscription
identifier)
activate A1P
   A1P -> A1P : AuthZ
note right

  Check authorization in collaboration
              with SME functions

end note
```

```
   A1P -> rApp : <<R1>> Unsubscribe from A1 policy status change response
deactivate A1P
@enduml
```

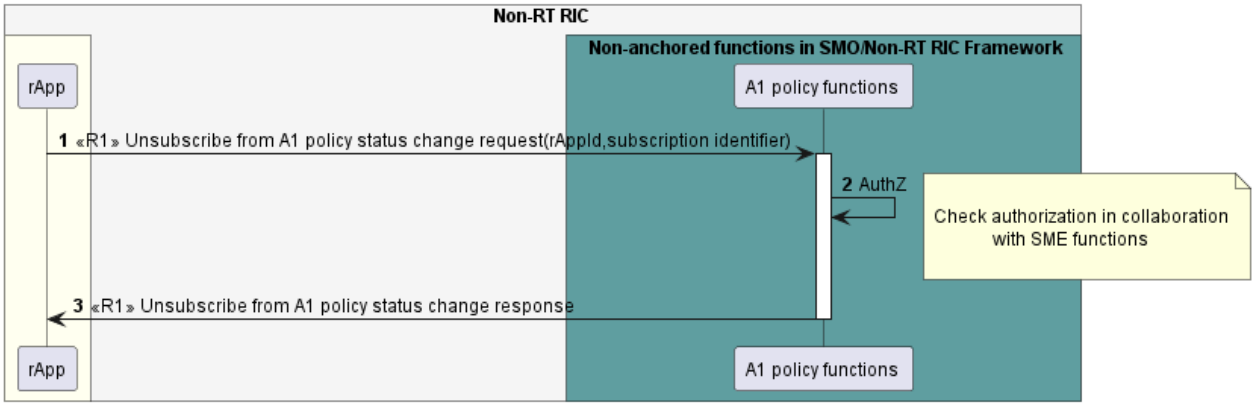

Figure 10.8.4.3-1: Unsubscribe from A1 policy status change use case flow diagram

10.8.5 Required data

To receive notifications regarding changes in status change of an A1 policy, the rApp as A1 policy service Consumer subscribes to notifications by providing rAppId and A1 policy identifier. When the A1 policy functions have successfully processed the request, they respond by providing a subscription identifier.

The A1 policy functions send notifications to the subscribed rApp when there are changes in status changes of the A1 policy by providing subscription identifier, and information about the changes (i.e. A1 policy status becomes enforced or not enforced).

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

11 Use cases for AI/ML Workflow Services

11.1 AI/ML workflow-related use case 1: AI/ML model registration and deregistration

11.1.1 Overview

This use case enables an rApp as AI/ML model management and exposure services Consumer to register AI/ML models, query AI/ML model registration, update AI/ML model registrations and deregister AI/ML models,

11.1.2 Background and goal of the use case

An AI/ML model producer rApp can register an AI/ML model, query and update the registration of an AI/ML model and deregister an AI/ML model with the AI/ML workflow functions.

11.1.3 Entities/resources involved in the use case

1) AI/ML workflow functions as AI/ML model management and exposure services Producer:

   a) support the functionality to allow an rApp to register, query and update the registration of, and deregister an AI/ML model;

   b) provide the response of success or failure results to the register AI/ML model, query, and update AI/ML model registration, and deregister AI/ML model request.

2) rApp as AI/ML model management and exposure services Consumer:

   a) support to initiate the procedure to register an AI/ML model, query and update the registration of an AI/ML model and deregister an AI/ML model.

11.1.4 Solutions

11.1.4.1 Register AI/ML model

Table 11.1.4.1-1: Register AI/ML model

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML model producer rApp registers an AI/ML model.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure services Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure services Producer.	
Assumptions	The AI/ML model that they intend to register in AI/ML workflow functions has not been registered before.	
Preconditions	The rApp is authorized to access the AI/ML model management and exposure services for registering an AI/ML model. The AI/ML model is certified and onboarded in SMO run time library.	
Begins when	The AI/ML model producer rApp determines the need to register an AI/ML model.	
Step 1 (M)	The rApp requests the AI/ML workflow functions to register the AI/ML model by providing the rAppId and AI/ML model-related information.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to register an AI/ML model.	
Step 3 (M)	The AI/ML workflow functions validate the register AI/ML model request.	
Step 4 (M)	The AI/ML workflow functions register the AI/ML model.	
Step 5 (M)	The AI/ML workflow functions respond to rApp with successful AI/ML model registration along with AI/ML model registration identifier.	
Ends when	The AI/ML model producer rApp was able to register the AI/ML model.	
Exceptions	n/a	
Post Conditions	The AI/ML model is discoverable, and the rApp can query, update, or delete the model registration.	
Traceability	REQ-R1-AIML-Registermodel-FUN1.	

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

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

src -> ML: <<R1>> Register AI/ML model request (rAppId, AI/ML model-related information)
ML -> ML: Authz
note right
Check authorization in
collaboration with SME functions
end note
ML -> ML: Validate
```

```
ML -> ML: Register AI/ML model
ML -> src: <<R1>> Register AI/ML model response (AI/ML model registration identifier)
@enduml
```

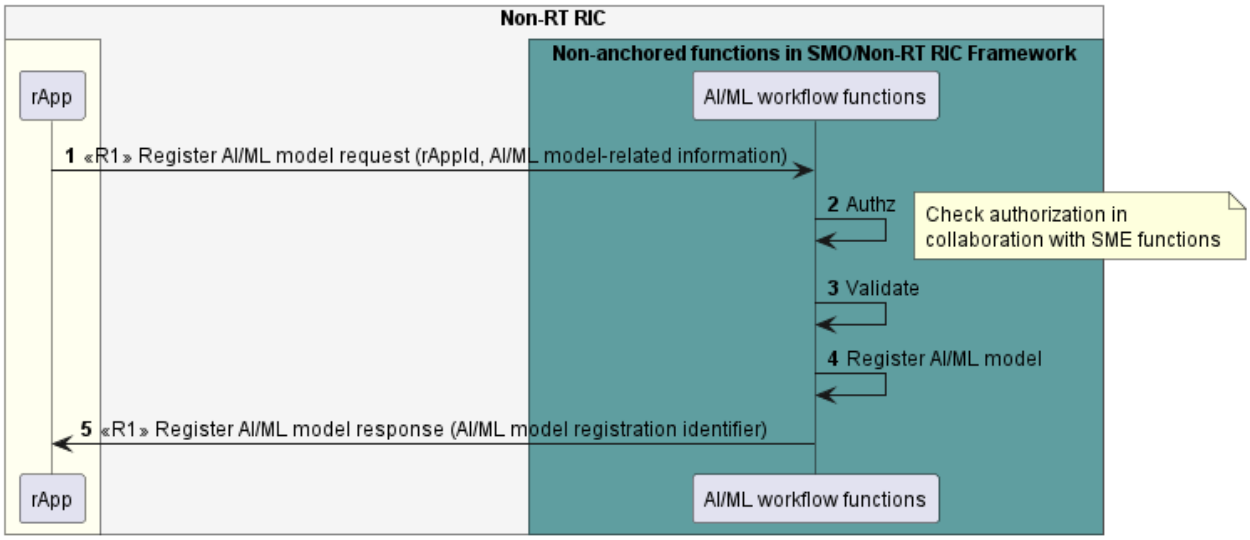

Figure 11.1.4.1-1: Register AI/ML model use case flow diagram

11.1.4.2 Query AI/ML model registration

Table 11.1.4.2-1: Query AI/ML model registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML model producer rApp queries an AI/ML model registration.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure services Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure services Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure services for registering an AI/ML model. - The AI/ML model has been registered by the rApp.	
Begins when	The AI/ML model producer rApp determines the need to query the registration of a registered AI/ML model.	
Step 1 (M)	The AI/ML model producer rApp requests to query the registration of an AI/ML model by providing rAppId and registration identifier.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to query the AI/ML model registration.	
Step 3 (M)	The AI/ML workflow functions look up the information of queried AI/ML model.	
Step 4 (M)	The AI/ML workflow functions respond to rApp with AI/ML model-related information.	
Ends when	The AI/ML model producer rApp was able to query the AI/ML model registration.	
Exceptions	n/a	
Post Conditions	The AI/ML model is discoverable, and the rApp can query, update or delete the AI/ML model registration.	
Traceability	REQ-R1-AIML-Registermodel-FUN1.	

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

```
autonumber

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

src -> ML: <<R1>> Query AI/ML model request (rAppId, AI/ML model registration identifier)
ML -> ML: Authz
note right
Check authorization in
collaboration with SME functions
end note
ML -> ML: Look up AI/ML model registration
ML -> src: <<R1>> Query AI/ML model response (AI/ML model-related information)
@enduml
```

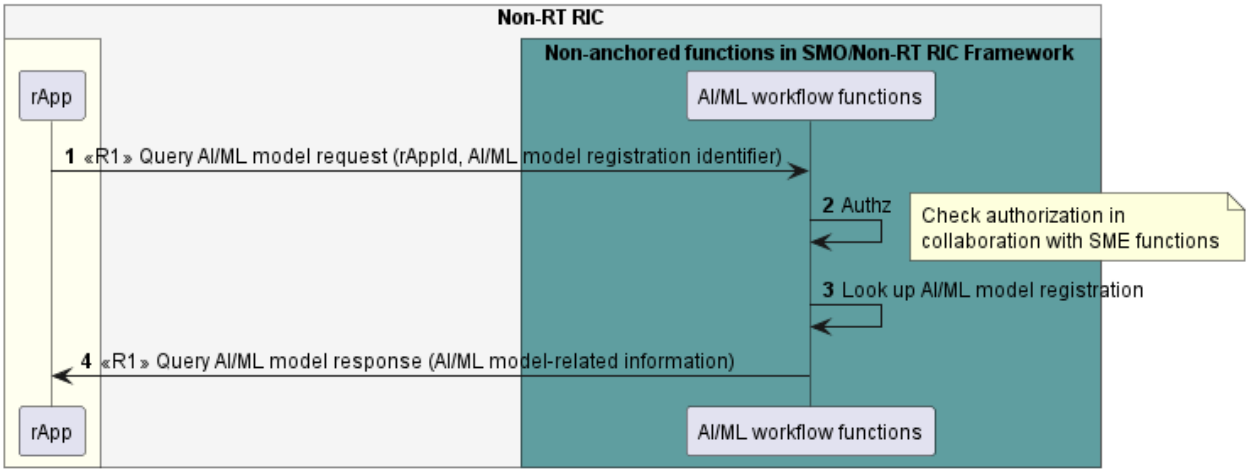

Figure 11.1.4.2-1: Query AI/ML model registration use case flow diagram

11.1.4.3 Update AI/ML model registration

Table 11.1.4.3-1: Update AI/ML model registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML model producer rApp updates the registration of an AI/ML model.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure services Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure services Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure services for updating an AI/ML model registration. - The AI/ML model has been registered by the rApp.	
Begins when	The AI/ML model producer rApp determines the need to update the registration of an AI/ML model.	
Step 1 (M)	The AI/ML model producer rApp requests to update the registration of an AI/ML model by providing rAppId, updated model registration information, etc.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to update the registration.	
Step 3 (M)	The AI/ML workflow functions validate the update AI/ML model registration request.	
Step 4 (M)	The AI/ML workflow functions update the registration of the AI/ML model.	
Step 5 (M)	The AI/ML workflow functions respond to the model producer rApp with successful update AI/ML model registration response.	
Ends when	The AI/ML model producer rApp was able to update the registration of the AI/ML model.	
Exceptions	n/a	
Post Conditions	The rApp can query, update, or delete the AI/ML model registration. The updated AI/ML model is discoverable and is notified to all subscribed model consumer rApps.	
Traceability	REQ-R1-AIML-Registermodel-FUN1.	

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

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

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "AI/ML workflow functions" as aif
  endbox
endbox

autonumber
app -> aif: <<R1>> Update model registration request \n (rAppId, AI/ML model registration
identifier, \n updated model registration information)
activate aif
aif --> aif: AuthZ
note right
Check authorization in
collaboration with SME functions
end note
aif --> aif: Validate
aif --> aif: Update model \n registration information
app <- aif: <<R1>> Update model registration response
deactivate aif
@enduml
```

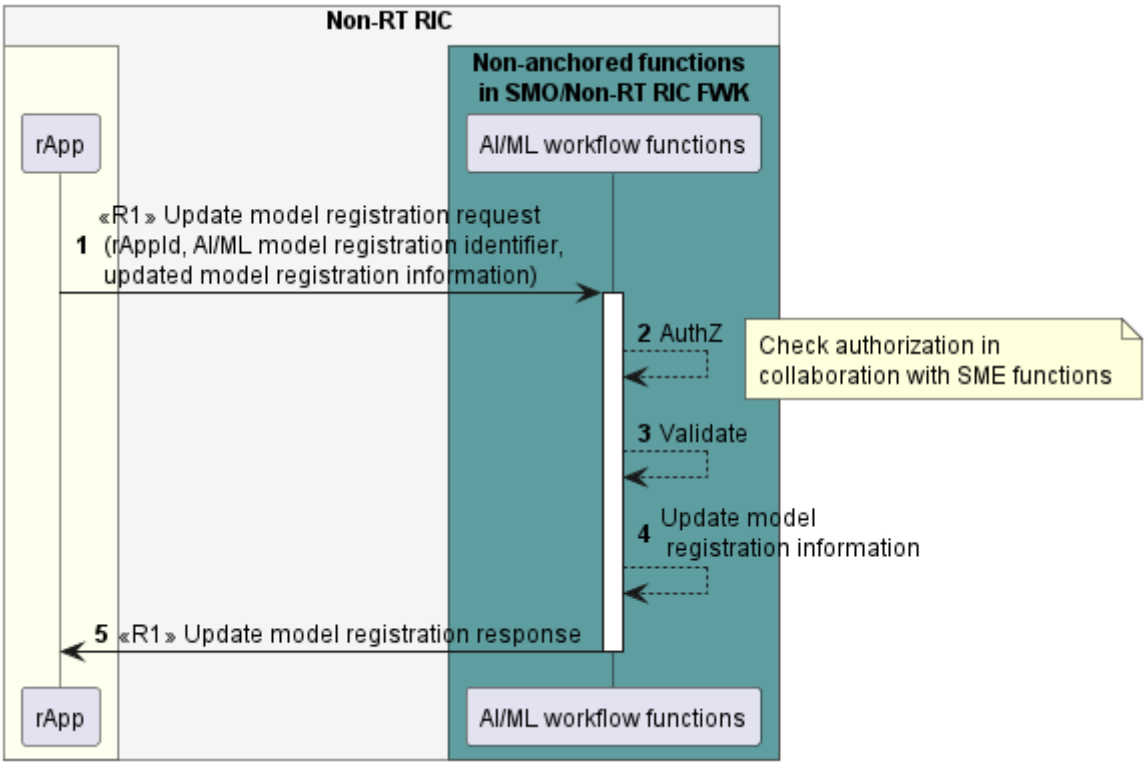

Figure 11.1.4.3-1: Update registration of an AI/ML model use case flow diagram

11.1.4.4 Deregister AI/ML model

Table 11.1.4.4-1: Deregister AI/ML model

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML model producer rApp deregisters an AI/ML model.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure services Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure services Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure services. - The AI/ML model has been registered by the rApp.	
Begins when	The AI/ML model producer rApp determines the need to deregister an AI/ML model it has previously registered.	
Step 1 (M)	The rApp requests AI/ML workflow functions to deregister the AI/ML model by providing rAppId and AI/ML model registration identifier.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to deregister the AI/ML model.	
Step 3 (M)	The AI/ML workflow functions delete the registration of the AI/ML model.	
Step 4 (M)	The AI/ML workflow functions respond to the rApp with successful deregister AI/ML model response.	
Ends when	The AI/ML model producer rApp was able to deregister the AI/ML model.	
Exceptions	n/a	
Post Conditions	The AI/ML model is not discoverable any longer.	
Traceability	REQ-R1-AIML-Registermodel-FUN1.	

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

```
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
autonumber

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox

endbox

src -> ML: <<R1>> Deregister AI/ML model request \n(rAppId, AI/ML model registration identifier)
ML -> ML: Authz
note right
Check authorization in
collaboration with SME functions
end note
ML -> ML: Deregister AI/ML model
ML -> src: <<R1>> Deregister AI/ML model response
@enduml
```

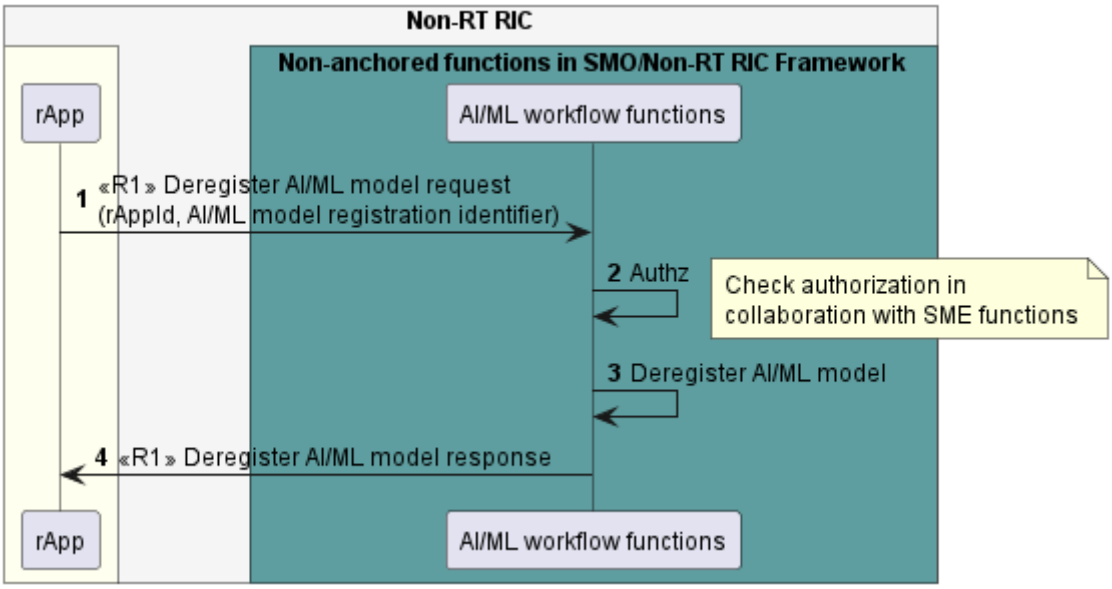

Figure 11.1.4.4-1: Deregister AI/ML model use case flow diagram

11.1.5 Required data

The AI/ML model-related information includes the AI/ML model identifier, AI/ML model description, input, and output datatype, etc.

For registering an AI/ML model, the rApp provides the rAppId and AI/ML model-related information. On successful registration, AI/ML workflow functions provide an AI/ML model registration identifier for the registration of the AI/ML model.

For querying the registration of an AI/ML model, the rApp needs to provide the rAppId and the AI/ML model registration identifier.

For updating the registration of an AI/ML model, the rApp needs to provide the rAppId, the AI/ML model registration identifier and the updated AI/ML model-related information or the modified part of the AI/ML model-related information.

For deregistering an AI/ML model, the rApp provides the rAppId and the AI/ML model registration identifier.

11.2 AI/ML workflow-related use case 2: Discovery of AI/ML Model

11.2.1 Overview

This use case enables an rApp to retrieve information about registered AI/ML Models and their associated information.

11.2.2 Background and goal of the use case

The discover AI/ML model procedure is defined as part of the AI/ML workflow services in R1GAP [1].

11.2.3 Entities/resources involved in the use case

1) AI/ML workflow functions in the role of AI/ML model management and exposure service Producer:
   a) support functionality allowing rApps to discover the registered AI/ML Models;
2) rApp in the role of AI/ML model management and exposure Consumer:
   a) initiates the procedure to discover the registered AI/ML models.

11.2.4 Solutions

11.2.4.1 Discover AI/ML models

Table 11.2.4.1-1: Discover AI/ML models

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp discovers registered AI/ML models.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML workflow services. - At least one AI/ML model is registered with the AI/ML workflow functions.	
Begins when	The rApp determines the need to discover the registered AI/ML models.	
Step 1 (M)	The rApp requests the AI/ML workflow functions for the information on the AI/ML Models by providing rAppId and optional selection criteria.	
Step 2 (M)	The AI/ML workflow functions validate if the rApp is authorized to discover the registered AI/ML models.	
Step 3 (M)	The AI/ML workflow functions provide the information about registered AI/ML models. The list contains the information of those AI/ML models that match the filtering criteria if those were provided by the rApp, or of all AI/ML models otherwise. For each AI/ML model, the model identifier and model metadata are provided.	
Ends when	The rApp was able to discover the registered AI/ML models.	
Exceptions	n/a	
Post Conditions	The rApp can use the AI/ML model identifier in other AI/ML workflow services.	
Traceability	REQ-R1-AI/ML-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 " AI/ML workflow functions " as aif
     endbox
rApp ->aif:<<R1>> Discover AI/ML model request\n(rAppId,Selection criteria)
activate aif
aif --> aif:AuthZ
note right
Check authorization in collaboration
with SME functions
end note
aif -> rApp :<<R1>>Discover AI/ML model response\n(model identifiers,model metadata)
deactivate aif
@enduml
```

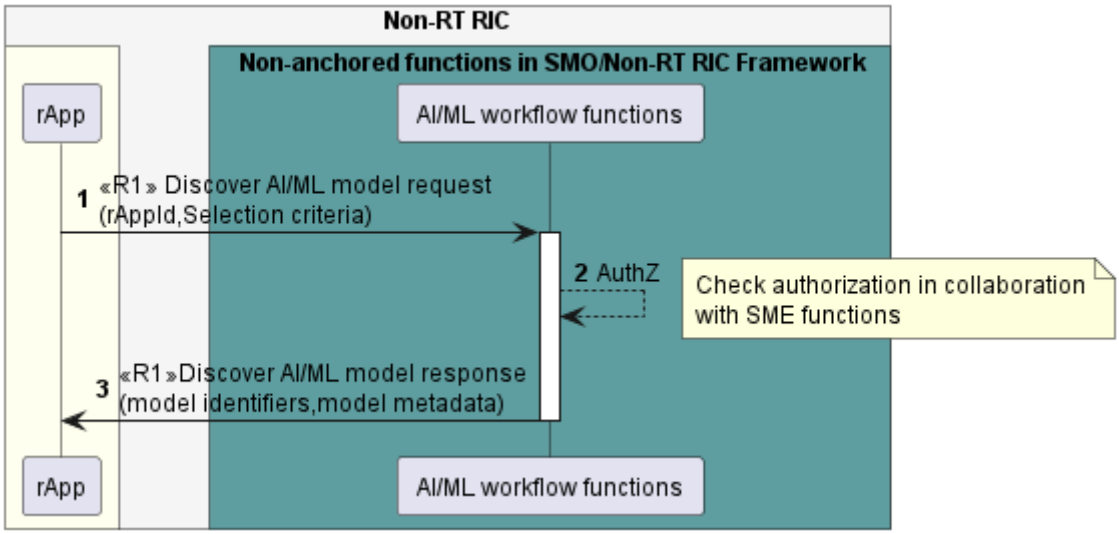

Figure 11.2.4.1-1: Discover AI/ML model use case flow diagram

11.2.5 Required data

For the Discover AI/ML model request, the rApp provides the rAppId and optional selection criteria.

The AI/ML workflow functions respond with information which includes the model identifiers and metadata information. The metadata information contains AI/ML model related information and training related information.

If selection criteria information was provided in the related request, the response only contains information about those AI/ML models information that match the filtering criteria.

11.3 AI/ML workflow-related use case 3: AI/ML training - AI/ML workflow functions producing AI/ML training services

11.3.1 Overview

This use case defines how an rApp as an AI/ML training services Consumer requests AI/ML training, queries AI/ML training job status, cancels AI/ML training, and receives notification about AI/ML training job status change, when the AI/ML workflow functions in the SMO/Non-RT RIC framework acts as the AI/ML training services Producer.

11.3.2 Background and goal of the use case

An AI/ML training services Consumer rApp can request the AI/ML workflow functions in the Non-RT RIC framework to train an AI/ML model. For on-going AI/ML training, the AI/ML training services Consumer rApp can query the training job status or cancel the AI/ML training. If the training job status is changed, the AI/ML workflow functions can notify the AI/ML training service Consumer rApp about the training job status change.

11.3.3 Entities/resources involved in the use case

1) AI/ML workflow functions as AI/ML training services Producer:

   a) support the functionality to allow an AI/ML training services Consumer rApp to request training, query training job status, and cancel training;

   b) provide the response of success or failure results to the request training, query training job status, and cancel training request;

   c) support the functionality to notify an AI/ML training services Consumer rApp about the training job status change.

2) AI/ML training services Consumer rApp:

   a) support to initiate the procedure to request AI/ML training, query AI/ML training job status, and cancel AI/ML training;

   b) support the functionality to receive notification of AI/ML training job status change.

11.3.4 Solutions

11.3.4.1 Request AI/ML training

Table 11.3.4.1-1: Request AI/ML training use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training services Consumer rApp requests training of an AI/ML model.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services.	
Begins when	The AI/ML training services Consumer rApp determines the need to train an AI/ML model.	
Step 1 (M)	The AI/ML training services Consumer rApp requests the AI/ML workflow functions to train an AI/ML model providing rAppId, information for training, and optionally a notification URI, etc.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the AI/ML training services Consumer rApp is authorized to request training.	
Step 3 (M)	The AI/ML workflow functions validate the request.	
Step 4 (M)	The AI/ML workflow functions create the training job.	
Step 5 (M)	The AI/ML workflow functions respond to the AI/ML training services Consumer rApp with training job identifier as a parameter.	
Ends when	The AI/ML training services Consumer rApp is able to create the training job at the AI/ML workflow functions.	
Exceptions	n/a	
Post Conditions	The AI/ML workflow functions can retrieve model to be trained from model repository and consume training data from DME.	
Traceability	REQ-R1-AI/ML-training-FUN1.	

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

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "AI/ML training services \n Consumer rApp" as con
  endbox

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "AI/ML workflow functions" as aif
  endbox
endbox

autonumber
con -> aif: <<R1>> Request training request \n (rApp id, information for training, \n notification
URI, etc.)
activate aif
aif --> aif: AuthZ
note right
Check authorization in
collaboration with SME functions
end note
aif --> aif: Validate
aif --> aif: Create training job
con <- aif: <<R1>> Request training response \n (training job id)
deactivate aif
@enduml
```

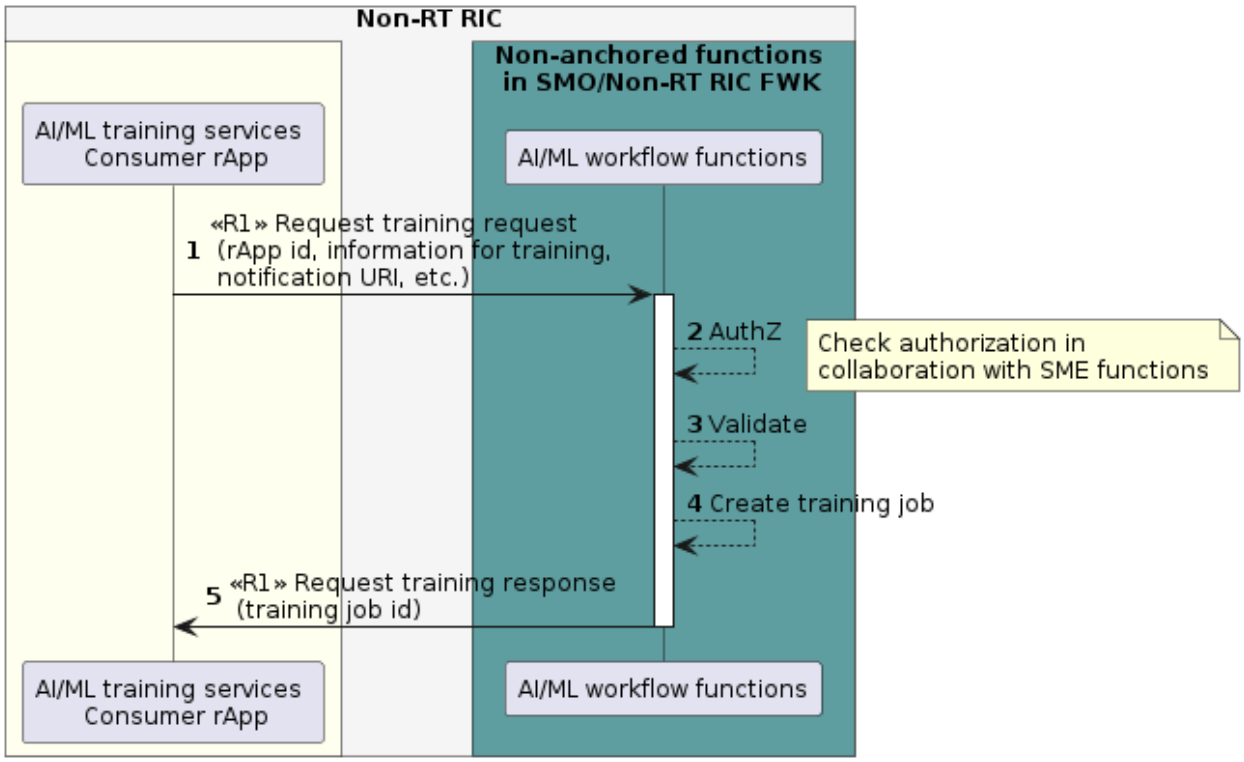

Figure 11.3.4.1-1: Request AI/ML training use case flow diagram

11.3.4.2 Query AI/ML training job status

Table 11.3.4.2-1: Query AI/ML training job status use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training services Consumer rApp query training job status of a created training job.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services.	
Begins when	The AI/ML training services Consumer rApp determines the need to query the training job status.	
Step 1 (M)	The AI/ML training services Consumer rApp queries the AI/ML workflow functions about the status of a created training job by providing rAppId and training job identifier.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the AI/ML training services Consumer rApp is authorized to query the training job status.	
Step 3 (M)	The AI/ML workflow functions validate the request.	
Step 4 (M)	The AI/ML workflow functions respond to the AI/ML training services Consumer rApp with training job status.	
Ends when	The AI/ML training services Consumer rApp is able to obtain the training job status.	
Exceptions	n/a	
Post Conditions	The training job status is known to the AI/ML training services Consumer rApp.	
Traceability	REQ-R1-AI/ML-training-FUN2.	

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

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "AI/ML training services \n Consumer rApp" as con
  endbox

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "AI/ML workflow functions" as aif
  endbox
endbox

autonumber
con -> aif: <<R1>> Query training status request \n (rApp id, training id)
activate aif
aif --> aif: AuthZ
note right
Check authorization in
collaboration with SME functions
end note
aif --> aif: Validate
con <- aif: <<R1>> Query training status response \n (training job status)
deactivate aif
@enduml
```

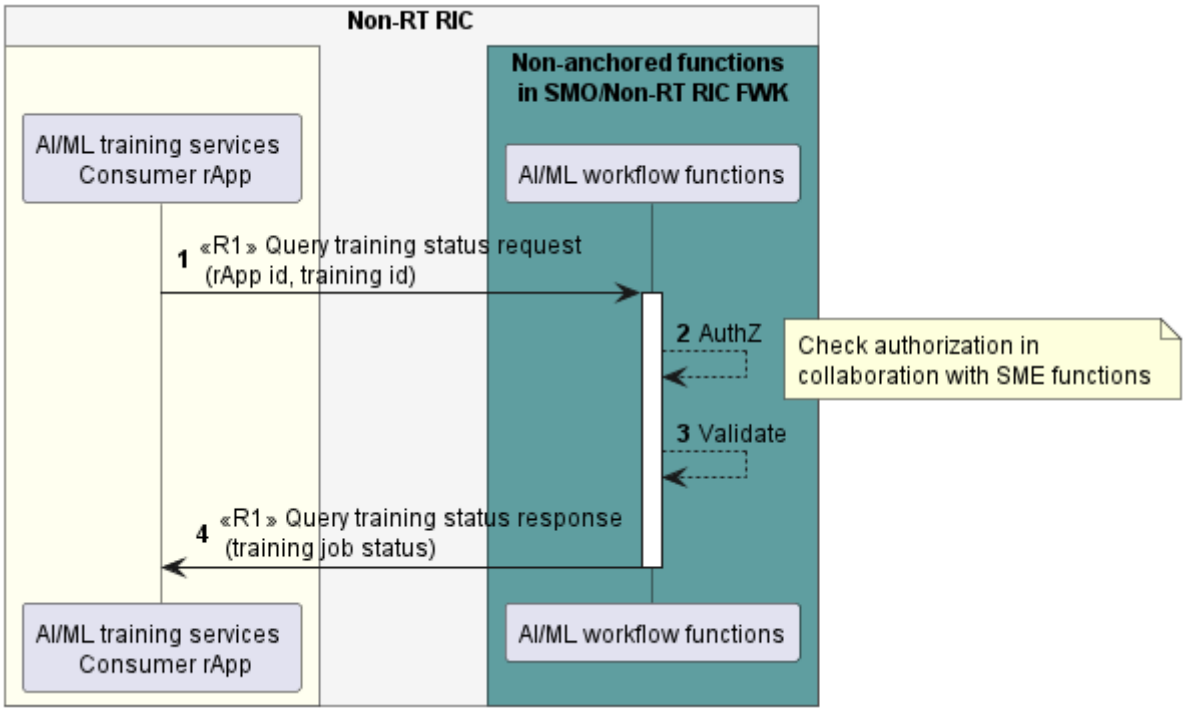

Figure 11.3.4.2-1: Query AI/ML training job status use case flow diagram

11.3.4.3 Cancel AI/ML training

Table 11.3.4.3-1: Cancel AI/ML training use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training services Consumer rApp cancels training of an AI/ML model.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services.	
Begins when	The AI/ML training services Consumer rApp determines to cancel the training of an AI/ML model.	
Step 1 (M)	The AI/ML training services Consumer rApp cancels the training by providing rAppId and training job identifier.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the AI/ML training services Consumer rApp is authorized to cancel the training.	
Step 3 (M)	The AI/ML workflow functions validate the request.	
Step 4 (M)	The AI/ML workflow functions stop the training job.	
Step 5 (M)	The AI/ML workflow functions respond to the AI/ML training services Consumer rApp.	
Ends when	The AI/ML workflow functions are able to cancel training of an AI/ML model.	
Exceptions	n/a	
Post Conditions	The training is cancelled, and the training job is terminated.	
Traceability	REQ-R1-AI/ML-training-FUN1.	

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

```
box Non-RT RIC #whitesmoke
  box #ivory
    participant AI/ML training services \n Consumer rApp as con
  endbox

  box Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "AI/ML workflow functions" as aif
  endbox
endbox

autonumber
con -> aif: <<R1>> Cancel training request \n (rApp id, training job id.)
activate aif
aif → aif: AuthZ
note right
Check authorization in
collaboration with SME functions
end note
aif --> aif: Validate
aif --> aif: Stop training job
con <- aif: <<R1>> Cancel training response
deactivate aif

@enduml
```

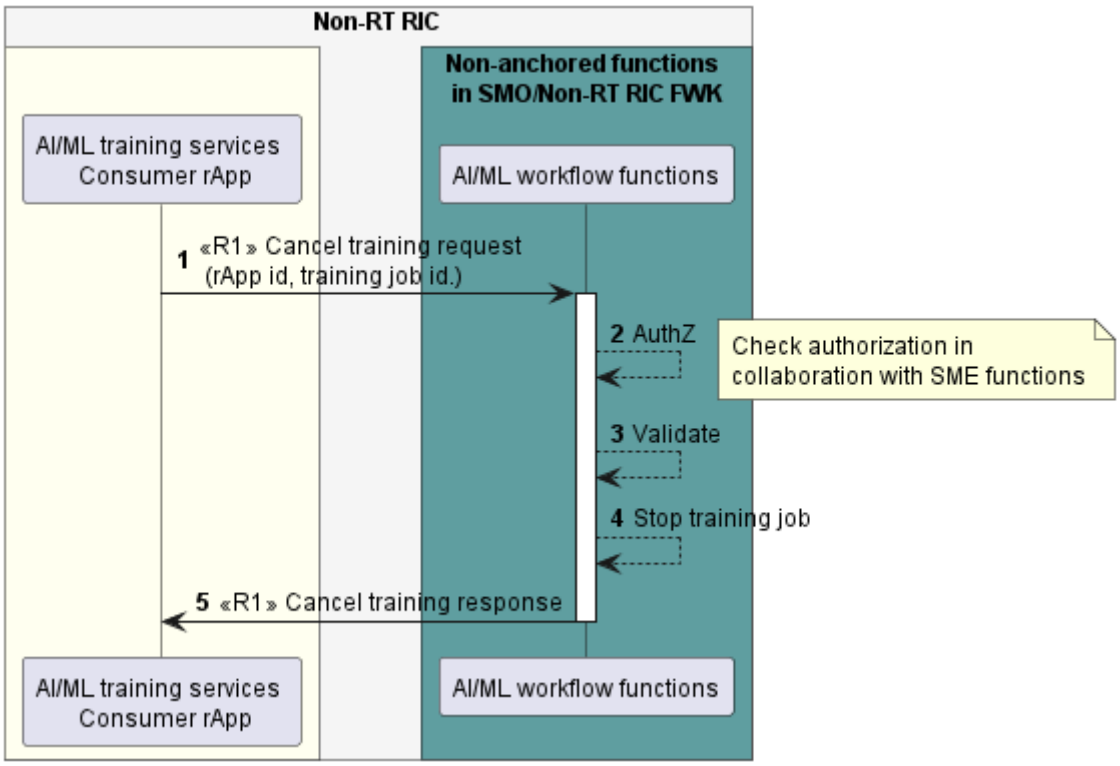

Figure 11.3.4.3-1: Cancel AI/ML training use case flow diagram

11.3.4.4 Notify AI/ML training job status change

Table 11.3.4.4-1: Notify AI/ML training job status change use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML workflow functions notify the AI/ML training services Consumer rApp about the training job status change of a created training job.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services. - The AI/ML training services Consumer rApp had provided a notification URI to the AI/ML workflow functions.	
Begins when	The AI/ML workflow functions determine the need to notify training job status change to the AI/ML training services Consumer rApp.	
Step 1 (M)	The AI/ML workflow functions notify the training job status change to AI/ML training services Consumer rApp.	
Ends when	The AI/ML training services Consumer rApp is notified about the updated training job status.	
Exceptions	n/a	
Post Conditions	The updated training job status is known to the AI/ML training services Consumer rApp.	
Traceability	REQ-R1-AI/ML-training-FUN3.	

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

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "AI/ML training services \n Consumer rApp" as con
  endbox

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "AI/ML workflow functions" as aif
  endbox
endbox

autonumber
con <- aif: <<R1>> Notify training job status change \n (training job id, training job status, model
id)
@enduml
```

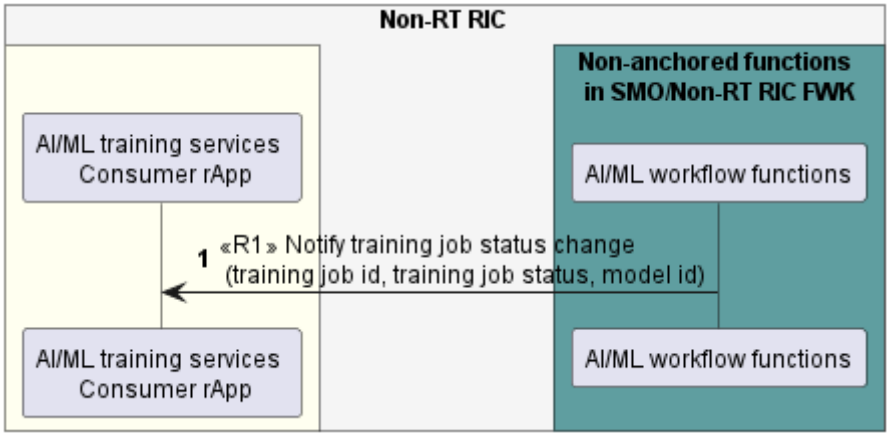

Figure 11.3.4.4-1: Notify AI/ML training job status change use case flow diagram

11.3.5 Required data

For requesting training, the AI/ML training services Consumer rApp needs to provide its rAppId, information for training (e.g. information about training dataset, model identifier, training criteria.), and optionally a notification URI to receive notification about training job status change. If the AI/ML workflow functions accept the training request, they assign a training job identifier for the created training job.

For querying training job status, the AI/ML training services Consumer rApp needs to provide its rAppId and the training job identifier. The AI/ML workflow functions respond with the status of the training job identified by the training job identifier.

For cancelling training, the AI/ML training services Consumer rApp needs to provide its rAppId and the training job identifier.

For notifying training job status change, the AI/ML workflow functions provide the training job status, the training job identifier, and optionally model identifier.

11.4 AI/ML workflow-related use case 4: AI/ML training - rApp producing AI/ML training services

11.4.1 Overview

This use case defines how an rApp as an AI/ML training services Consumer requests AI/ML training, queries AI/ML training job status, cancels AI/ML training, and receives notification about AI/ML training job status change, when another rApp acts as the AI/ML training services Producer.

11.4.2 Background and goal of the use case

An AI/ML training services Consumer rApp can request the AI/ML training service Producer rApp to train an AI/ML model. For on-going AI/ML training, the AI/ML training services Consumer rApp can query the training job status or cancel the AI/ML training. If the training job status is changed, the AI/ML training services Producer rApp can notify the AI/ML training service Consumer rApp about the training job status change.

11.4.3 Entities/resources involved in the use case

1) AI/ML training services Producer rApp:

   a) supports the functionality to allow an AI/ML training services Consumer rApp to request training, query training job status, and cancel training;

   b) provides the response of success or failure results to the request training, query training job status, and cancel training request;

   c) supports the functionality to notify an AI/ML training services Consumer rApp about the training job status change.

2) AI/ML training services Consumer rApp:

   a) supports to initiate the procedure to request AI/ML training, query AI/ML training job status, and cancel AI/ML training;

   b) supports the functionality to receive notification of AI/ML training job status change.

11.4.4 Solutions

11.4.4.1 Request AI/ML training

Table 11.4.4.1-1: Request AI/ML training use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training services Consumer rApp requests training of an AI/ML model.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - AI/ML training service Producer rApp in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services. - The AI/ML training services Producer rApp is deployed, authenticated, and authorized to produce AI/ML training services.	
Begins when	The AI/ML training services Consumer rApp determines the need to train an AI/ML model.	
Step 1 (M)	The AI/ML training services Consumer rApp requests the AI/ML training services Producer rApp to train an AI/ML model providing rAppId, information for training, and optionally a notification URI, etc.	
Step 2 (M)	The AI/ML training services Producer rApp checks with SME functions whether the AI/ML training services Consumer rApp is authorized to request training.	
Step 3 (M)	The AI/ML training services Producer rApp validates the request.	
Step 4 (M)	The AI/ML training services Producer rApp creates the training job.	
Step 5 (M)	The AI/ML training services Producer rApp responds to the AI/ML training services Consumer rApp with training job identifier as a parameter.	
Ends when	The AI/ML training services Consumer rApp is able to create the training job at the AI/ML training services Producer rApp.	
Exceptions	n/a	
Post Conditions	The AI/ML training service Producer rApp can retrieve model to be trained from model repository and consume training data from DME.	
Traceability	REQ-R1-AI/ML-training-FUN1.	

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

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "AI/ML training services \n Consumer rApp" as con
    participant "AI/ML training services \n Producer rApp" as aif
  endbox

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "SME functions" as sme
  endbox
endbox

autonumber
con -> aif: Request training request \n (rApp id, information for training, \n notification URI,
etc.)
activate aif
aif <-> sme: <<R1>> Check authorization in \n collaboration with SME functions
aif --> aif: Validate
aif --> aif: Create training job
con <- aif: Request training response \n (training job id)
deactivate aif
@enduml
```

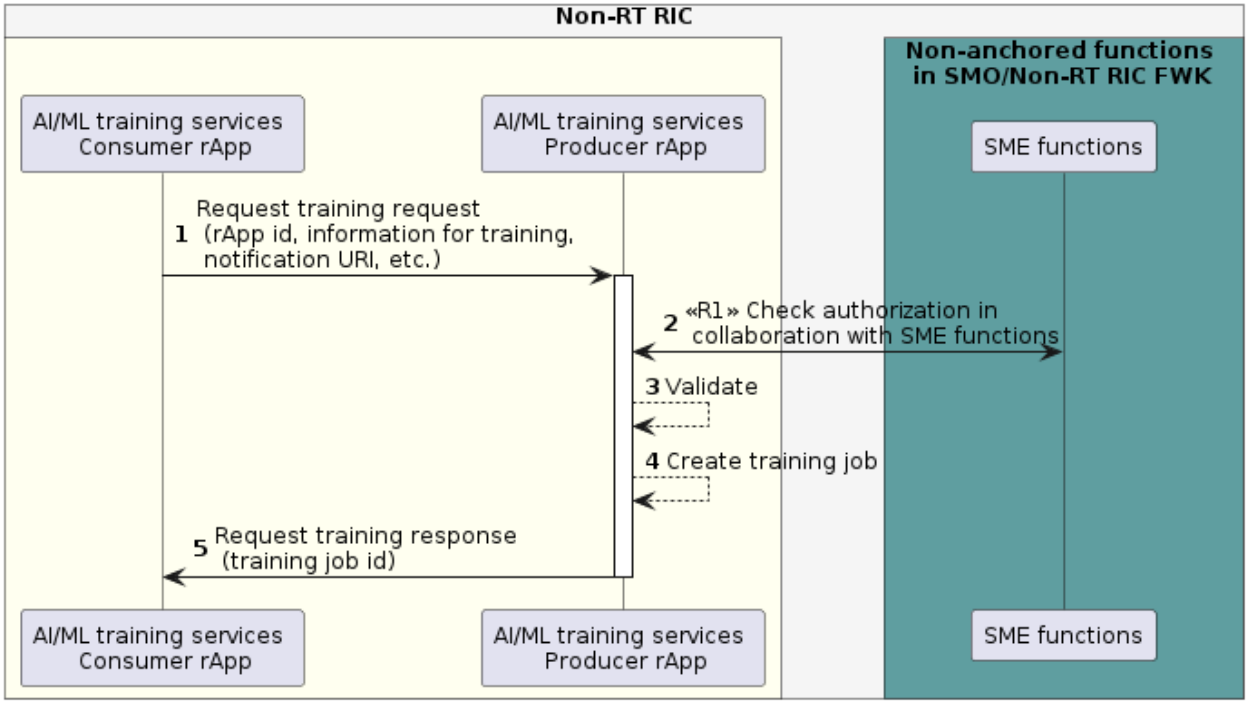

Figure 11.4.4.1-1: Request AI/ML training use case flow diagram

11.4.4.2 Query AI/ML training job status

Table 11.4.4.2-1: Query AI/ML training job status use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training services Consumer rApp query training job status of a created training job.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - AI/ML training service Producer rApp in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services. - The AI/ML training services Producer rApp is deployed, authenticated, and authorized to produce AI/ML training services.	
Begins when	The AI/ML training services Consumer rApp determines the need to query the training job status.	
Step 1 (M)	The AI/ML training services Consumer rApp queries the AI/ML training service Producer rApp about the status of a created training job by providing rAppId and training job identifier.	
Step 2 (M)	The AI/ML training service Producer rApp checks with SME functions whether the AI/ML training services Consumer rApp is authorized to query the training job status.	
Step 3 (M)	The AI/ML training service Producer rApp validates the request.	
Step 4 (M)	The AI/ML training service Producer rApp responds to the AI/ML training services Consumer rApp with training job status.	
Ends when	The AI/ML training services Consumer rApp is able to obtain the training job status.	
Exceptions	n/a	
Post Conditions	The training job status is known to the AI/ML training services Consumer rApp.	
Traceability	REQ-R1-AI/ML-training-FUN2.	

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

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "AI/ML training services \n Consumer rApp" as con
    participant "AI/ML training services \n Producer rApp" as aif
  endbox

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "SME functions" as sme
  endbox
endbox

autonumber
con -> aif: Query training job status request (rApp id, training id)
activate aif
aif <-> sme: <<R1>> Check authorization in \n collaboration with SME functions
aif --> aif: Validate
con <- aif: Query training job status response (training job status)
deactivate aif
@enduml
```

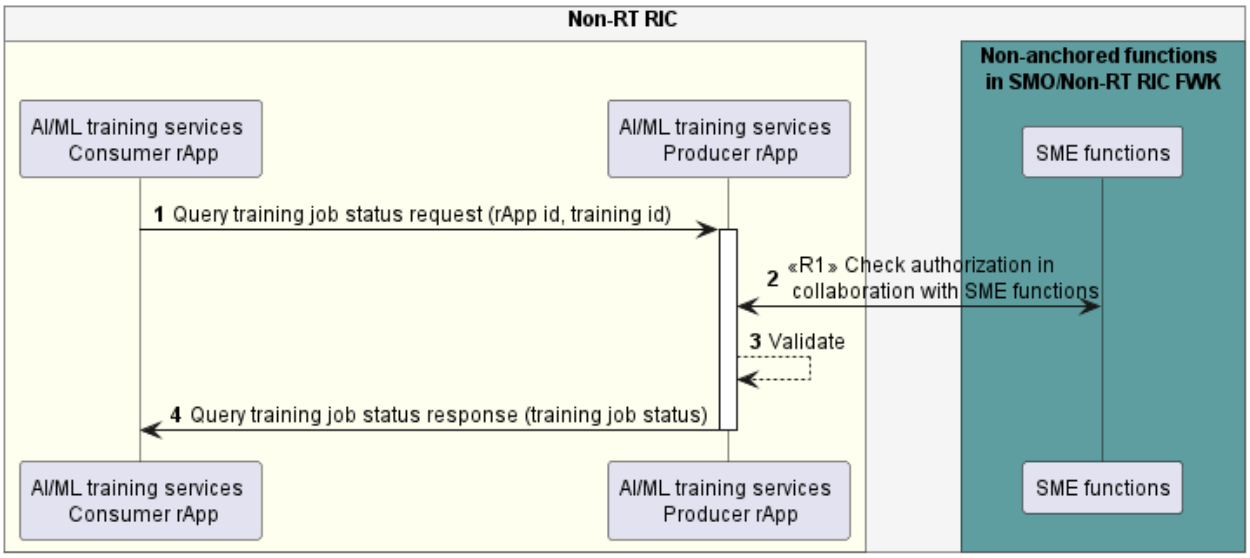

Figure 11.4.4.2-1: Query AI/ML training job status use case flow diagram

11.4.4.3 Cancel AI/ML training

Table 11.4.4.3-1: Cancel AI/ML training use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training services Consumer rApp cancels training of an AI/ML model.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - AI/ML training service Producer rApp in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services. - The AI/ML training services Producer rApp is deployed, authenticated, and authorized to produce AI/ML training services.	
Begins when	The AI/ML training services Consumer rApp determines to cancel the training of an AI/ML model.	
Step 1 (M)	The AI/ML training services Consumer rApp cancels the training by providing rAppId and training job identifier.	
Step 2 (M)	The AI/ML training service Producer rApp checks with SME functions whether the AI/ML training services Consumer rApp is authorized to cancel the training.	
Step 3 (M)	The AI/ML training service Producer rApp validates the request.	
Step 4 (M)	The AI/ML training service Producer rApp stops the training job.	
Step 5 (M)	The AI/ML training service Producer rApp responds to the AI/ML training services Consumer rApp.	
Ends when	The AI/ML training service Producer rApp is able to cancel training of an AI/ML model.	
Exceptions	n/a	
Post Conditions	The training is cancelled, and the training job is terminated.	
Traceability	REQ-R1-AI/ML-training-FUN1.	

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

box "Non-RT RIC" #whitesmoke
  box #ivory
    participant "AI/ML training services \n Consumer rApp" as con
    participant "AI/ML training services \n Producer rApp" as aif
  endbox

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "SME functions" as sme
  endbox
endbox

autonumber
con -> aif: Cancel training request \n (rApp id, training job id.)
activate aif
aif <-> sme: <<R1>> Check authorization in \n collaboration with SME functions
aif --> aif: Validate
aif --> aif: Stop training job
con <- aif: Cancel training response
deactivate aif
@enduml
```

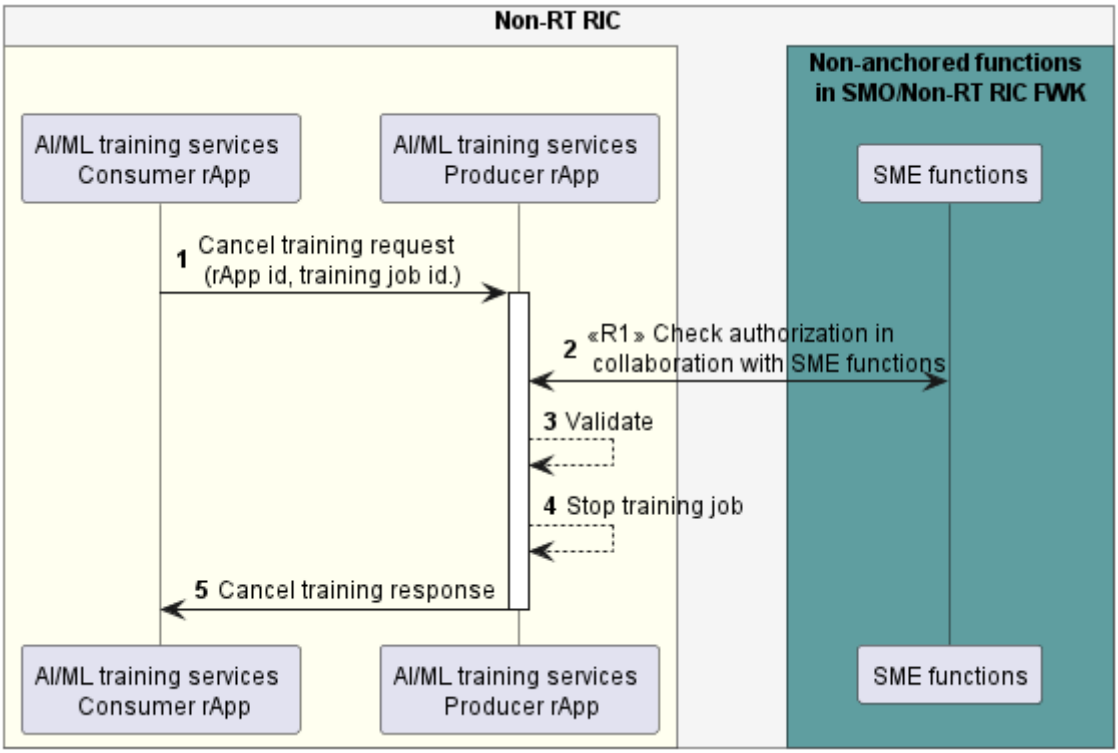

Figure 11.4.4.3-1: Cancel AI/ML training use case flow diagram

11.4.4.4 Notify AI/ML training job status change

Table 11.4.4.4-1: Notify AI/ML training job status change use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML training service Producer notifies the AI/ML training services Consumer rApp about the training job status change of a created training job.	
Actors and Roles	- AI/ML training services Consumer rApp in the role of Service Consumer. - AI/ML training service Producer rApp in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML training services Consumer rApp is deployed, authenticated, and authorized to consume AI/ML training services. - The AI/ML training services Consumer rApp had provided a notification URI to the AI/ML training services Producer rApp. - The AI/ML training services Producer rApp is deployed, authenticated, and authorized to produce AI/ML training services.	
Begins when	The AI/ML training service Producer determines the need to notify training job status change to the AI/ML training services Consumer rApp.	
Step 1 (M)	The AI/ML training service Producer notifies the training job status change to AI/ML training services Consumer rApp.	
Ends when	The AI/ML training services Consumer rApp is notified about the updated training job status.	
Exceptions	n/a	
Post Conditions	The updated training job status is known to the AI/ML training services Consumer rApp.	
Traceability	REQ-R1-AI/ML-training-FUN3.	

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

```
skinparam SequenceGroupBodyBackgroundColor Transparent

box "Non-RT RIC" #whitesmoke
  participant "AI/ML training services \n Consumer rApp" as con
  participant "AI/ML training services \n Producer rApp" as aif
endbox
autonumber
con <- aif: Notify training job status change \n (training job id, training job status, model id)
@enduml
```

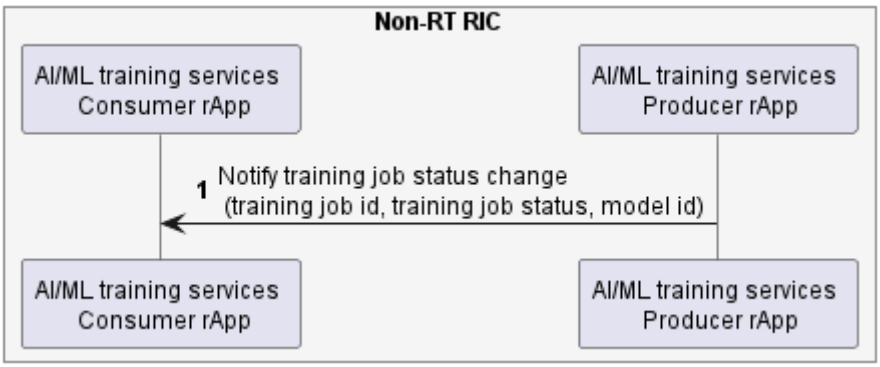

Figure 11.4.4.4-1: Notify AI/ML training job status change use case flow diagram

11.4.5 Required data

For requesting training, the AI/ML training services Consumer rApp needs to provide its rAppId, information for training (e.g. information about training dataset, model identifier, training criteria.), and optionally a notification URI to receive notification about training job status change. If the AI/ML training service Producer rApp accepts the training request, it assigns a training job identifier for the created training job.

For querying training job status, the AI/ML training services Consumer rApp needs to provide its rAppId and the training job identifier. The AI/ML training service Producer rApp responses with the status of the training job identified by the training job identifier.

For cancelling training, the AI/ML training services Consumer rApp needs to provide its rAppId and the training job identifier.

For notifying training job status change, the AI/ML training service Producer rApp provides the training job status, the training job identifier, and optionally model identifier.

11.5 AI/ML workflow-related use case 5: AI/ML model Retrieve

11.5.1 Overview

This use case defines how an rApp retrieves model location details as a consumer of registered AI/ML model(s).

11.5.2 Background and goal of the use case

The AI/ML model retrieve procedures are defined as part of the AI/ML workflow services as part of AI/ML workflow services in R1GAP [1].

11.5.3 Entities/resources involved in the use case

1) AI/ML workflow functions:

   a) support model management and exposure functionality to allow an rApp to retrieve the location details of AI/ML model(s).

2) rApp:

a) initiates the procedure to retrieve the location details of AI/ML model(s).

11.5.4 Solutions

11.5.4.1 Retrieve model

Table 11.5.4.1-1: Model retrieve

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp retrieves AI/ML model(s).	
Actors and Roles	- rApp in the role of Model retrieve service Consumer. - AI/ML workflow functions in the role of Model retrieve service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML Model retrieve service. - The rApp has the model identifier(s).	
Begins when	The rApp determines the need to retrieve the AI/ML model.	
Step 1 (M)	The rApp requests the AI/ML workflow functions to retrieve an AI/ML model by providing the rApp ID, and the AI/ML model identifier(s).	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the rApp is authorized to retrieve AI/ML models.	
Step 3 (M)	The AI/ML workflow functions fetch the model location details of registered AI/ML model(s).	
Step 4 (M)	The AI/ML workflow functions respond to rApp AI/ML model identifier, model location details.	
Ends when	The rApp has the location details.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-Retrievemodel-FUN1.	

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

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

group Model Consumer rApp model retrieve
rApp -> ML: <<R1>> Retrieve AI/ML model information request \n (rAppId, AI/ML model identifier(s))
ML -> ML: AuthZ
ML -> ML: fetch the model location details \n of registered AI/ML model(s)
ML -> rApp: <<R1>> Retrieve AI/ML model information response \n (AI/ML model identifier(s), model
location details)
end
@enduml
```

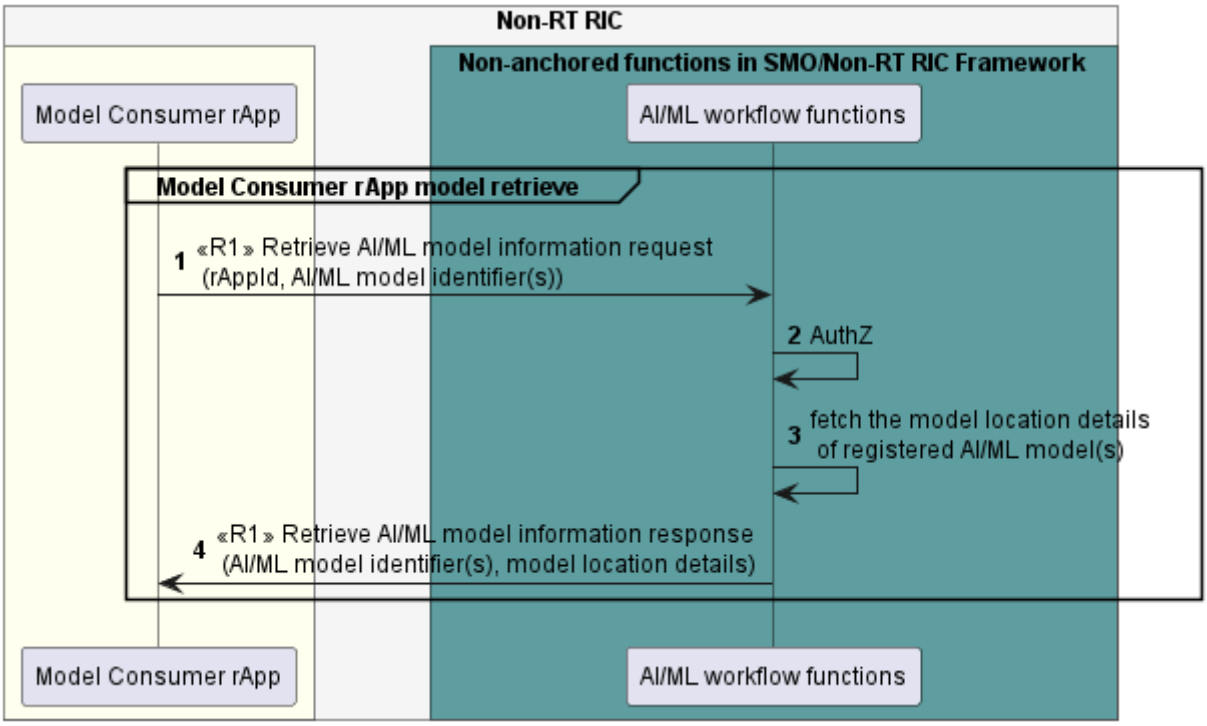

Figure 11.5.4.1-1: Retrieve model use case flow diagram

11.5.5 Required data

For Retrieve AI/ML model information request, the rApp provides the rAppId, and model identifier(s).

For Retrieve AI/ML model information response, the AI/ML workflow functions provide model identifier(s) and model location details.

11.6 AI/ML workflow-related use case 6: AI/ML model deployment

11.6.1 Overview

This use case defines how an rApp requests the deployment of an AI/ML model as an AI/ML model Consumer.

11.6.2 Background and goal of the use case

The AI/ML model deployment procedures are defined as part of the AI/ML workflow services as part of AI/ML workflow services in R1GAP [1].

11.6.3 Entities/resources involved in the use case

1) AI/ML workflow functions:

   a) support model management and exposure functionality to allow an rApp to request the deployment of an AI/ML model.

2) rApp:

   a) initiates the procedure to deploy the updated artifact version of an AI/ML model being consumed by the rApp.

11.6.4 Solutions

11.6.4.1 Retrieve model

Table 11.6.4.1-1: Model deployment

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp requests to deploy an AI/ML model being consumed by the rApp.	
Actors and Roles	- rApp in the role of Model deployment service Consumer. - AI/ML workflow functions in the role of Model deployment service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the Model deployment service. - The rApp is authorized to access the AI/ML model referenced in the deployment request. - The AI/ML model referenced in the deployment request is being consumed by the rApp requesting the deployment.	
Begins when	The rApp determines the need to deployment an AI/ML model.	
Step 1 (M)	The rApp requests the AI/ML workflow functions to deploy an AI/ML model by providing the rAppId, the AI/ML model identifier and the target deployment location of the AI/ML model.	
Step 2 (M)	The AI/ML workflow functions authorize the rApps deployment request.	
Step 3 (M)	The AI/ML workflow functions send the model deployment response to the rApp. The AI/ML workflow functions trigger the rApp deployment procedure. See note.	
Step 4 (M)	The AI/ML workflow functions notify the rApp the model deployment status.	
Ends when	n/a	
Exceptions	n/a	
Post Conditions	The model deployment status is known to the rApp.	
Traceability	REQ-R1-AIML-Deploymodel-FUN1.	
NOTE: The rApp deployment procedure is not specified in the present document.		

```
@startuml!pragma teoz trueskinparam ParticipantPadding 5skinparam BoxPadding 10skinparam
defaultFontSize 12skinparam lifelineStrategy solid skinparam SequenceGroupBackgroundColor
Transparentskinparam SequenceGroupBodyBackgroundColor Transparentautonumber
box "Non-RT RIC" #whitesmoke    box #ivory  participant "Model Consumer rApp" as rApp   endbox
    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue   participant  "AI/ML
workflow functions" as ML   endboxendbox


group Model deploymentrApp -> ML : <<R1>> Model deployment request (rAppId, AI/ML Model
Identifier,\n  the target deployment location of the AI/ML model)ML -> ML: AuthZnote right  Check
authorization in collaboration  with SME functionsend note
ML -> rApp : <<R1>> Model deployment response
note over ML    The rApp deployment procedure is    not specified in this specification end note
ML -> rApp : <<R1>> Model deployment status notification (AI/ML model identifier, deployment status)
@enduml
```

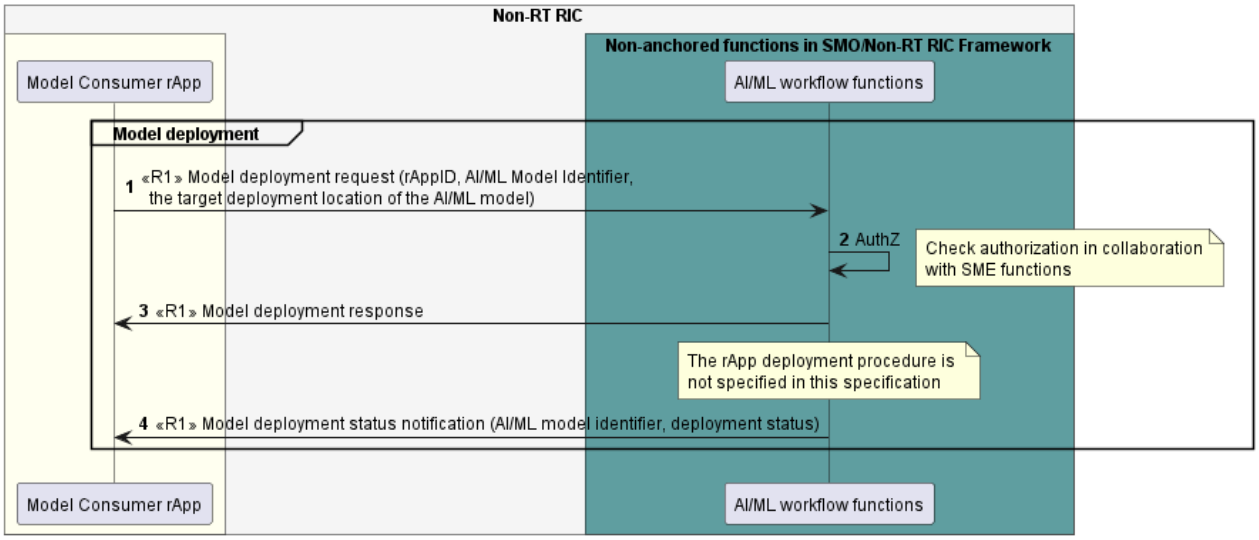

Figure 11.6.4.1-1: Model deployment use case flow diagram

11.6.5 Required data

For a model deployment request, the rApp provides the rAppId, AI/ML model identifier, and the target deployment location of the AI/ML model.

For a model deployment status notification, AI/ML workflow functions provide AI/ML model identifier and the deployment status.

11.7 AI/ML workflow-related use case 7: AI/ML model performance monitoring - AIML workflow functions consuming performance monitoring service

11.7.1 Overview

This use case enables the AI/ML workflow functions as AI/ML model performance monitoring service consumer to subscribe to AI/ML model performance and receive notifications of AI/ML model performance information.

11.7.2 Background and goal of the use case

An AI/ML model performance monitoring service consumer can subscribe to AI/ML model performance and receive notifications of AI/ML model performance information.

11.7.3 Entities/resources involved in the use case

1) rApp as AI/ML model performance monitoring service Producer:

   a) supports the functionality to allow an authorized consumer to subscribe to, and receive notifications of, AI/ML model performance information.

2) AI/ML workflow functions as AI/ML model performance monitoring service Consumer:

   a) support the functionality to initiate the procedure to subscribe to, and receive notifications of, AI/ML model performance information.

11.7.4 Solutions

11.7.4.1 Subscribe AI/ML model performance

Table 11.7.4.1-1: Subscribe AI/ML model performance

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML workflow functions subscribe to AI/ML model performance.	
Actors and Roles	- rApp in the role of AI/ML model performance monitoring service Producer. - AI/ML workflow functions in the role of AI/ML model performance monitoring service Consumer.	
Assumptions	- The supported AI/ML model performance information are part of the AI/ML model registration information.	
Preconditions	- The rApp has deployed a registered AI/ML model.	
Begins when	The AI/ML workflow functions determine the need to monitor the AI/ML model performance of a deployed AI/ML model.	
Step 1 (M)	The AI/ML workflow functions subscribe to AI/ML model performance with AI/ML model identifier, required AI/ML model performance information (optional), periodicity (optional), event notification conditions (optional), and notification URI as parameters.	
Step 2 (M)	The rApp validates the subscription request.	
Step 3 (M)	The rApp creates the subscription.	
Step 4 (M)	The rApp responds to the request with subscription ID as a parameter.	
Ends when	The subscription is created, and AI/ML workflow functions have the subscription ID.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-Performancer-FUN1.	

```
@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 "AI/ML Workflow functions" as AIML
end box

AIML -> rApp : <<R1>> Subscribe AI/ML model performance request (AI/ML model identifier, \n AI/ML
model performance information (optional), Periodicity (optional),\n EventNotificationConditions
(optional), Notification URI)
rApp --> rApp : Validate request
rApp --> rApp : Create subscription
rApp -> AIML: <<R1>> Subscribe AI/ML model performance response (Subscription ID)

@enduml
```

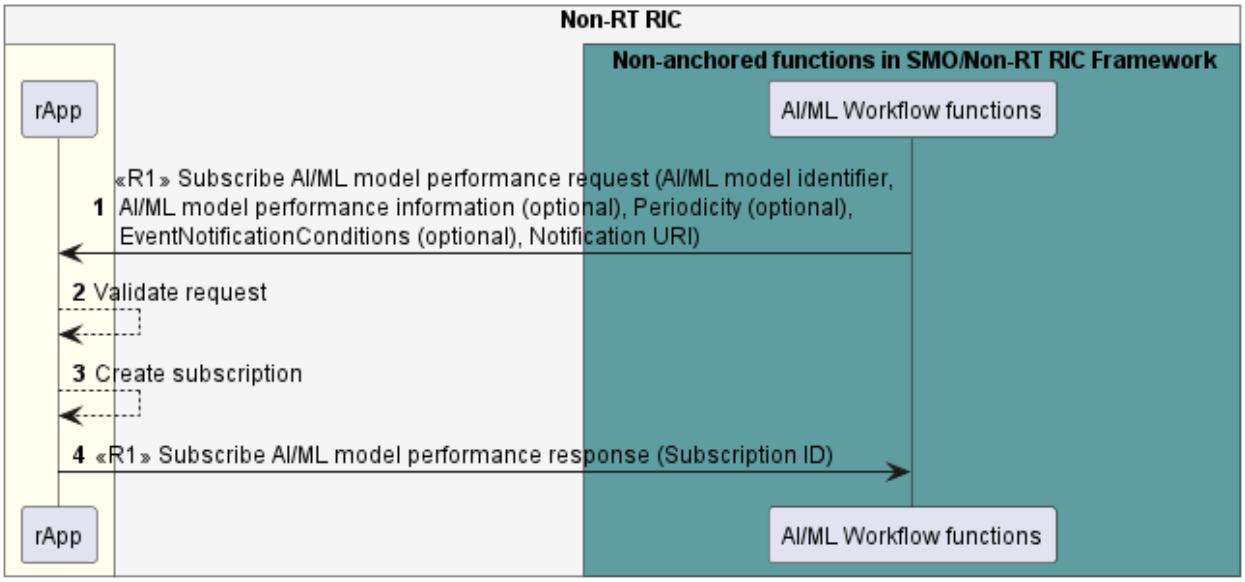

Figure 11.7.4.1-1: Subscribe AI/ML model performance use case flow diagram

11.7.4.2 Notify AI/ML model performance

Table 11.7.4.2-1: Notify AI/ML model performance

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp notifies subscribed consumers of AI/ML model performance information.	
Actors and Roles	- rApp in the role of AI/ML model performance monitoring service Producer. - AI/ML workflow functions in the role of AI/ML model performance monitoring service Consumer.	
Assumptions	- The supported AI/ML model performance information are part of the AI/ML model registration information.	
Preconditions	- The rApp has deployed a registered AI/ML model. - The AI/ML workflow functions subscribed to AI/ML model performance.	
Begins when	The rApp determines the need to notify subscribed consumers of the performance information of a deployed AI/ML model.	
Alternative procedure	In the case of event-based notifications, the rApp notifies subscribed consumers of the AI/ML model performance information every time an event is triggered in Step 1.	
Step 1 (M)	The rApp checks the occurrence of the notification events specified in the subscription request.	
Step 2 (M)	The rApp notifies the subscribed consumers of the AI/ML model performance information providing the requested AI/ML model performance information and the event triggered as parameters. If no AI/ML model performance information is specified in the subscription request, all supported AI/ML model performance information is provided.	
Alternative procedure	In the case of periodic notifications, the rApp notifies subscribed consumers of the AI/ML model performance information as per the notification periodicity specified in the subscription.	
Step 3 (M)	The rApp notifies the subscribed consumers of the AI/ML model performance information providing the requested AI/ML model performance information. If no AI/ML model performance information is specified in the subscription request, all supported AI/ML model performance information is provided.	
Ends when	The rApp is able to terminate the subscription or the AI/ML model is no longer deployed.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-Performancer-FUN2.	

```
@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 "AI/ML model performance monitoring \n service Producer rApp" as rApp
end box
box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "AI/ML Workflow functions" as AIML
end box

Loop Periodicity of notifications specified in the subscription request
    Alt Event based notifications
    rApp --> rApp : Check if specified event is triggered
    rApp -> AIML : <<R1>> Notify AI/ML model performance (AI/ML model identifier, \n AI/ML model
performance information, EventNotificationType)
    else Periodical notifications
        rApp -> AIML : <<R1>> Notify AI/ML model performance (AI/ML model identifier, \n AI/ML model
performance information)
end
end
```

```
@enduml
```

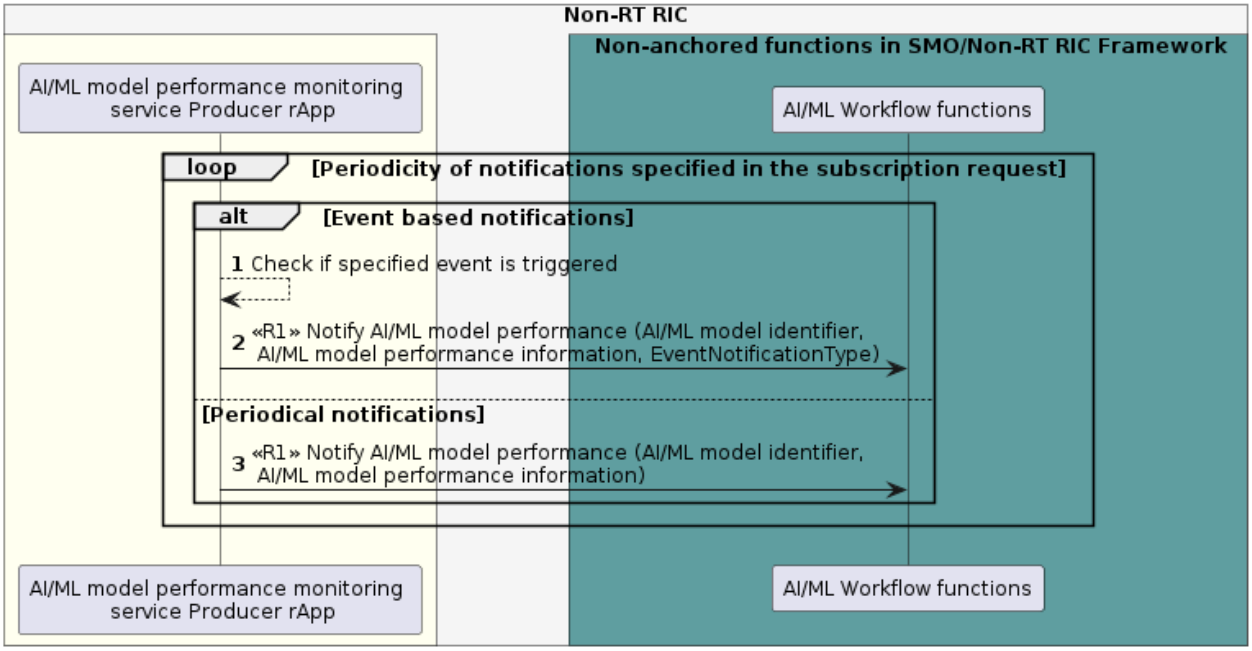

Figure 11.7.4.2-1: Notify AI/ML model performance use case flow diagram

11.7.5 Required data

For the Subscribe AI/ML model performance request, the AI/ML workflow functions provide the AI/ML model identifier and notification URI. In the Subscribe AI/ML model performance request, the AI/ML workflow functions optionally provide the required AI/ML model performance information, Notification event conditions upon which the Notify AI/ML model performance is triggered, and periodicity of the notification.

For the Subscribe AI/ML model performance response, the rApp provides the Subscription ID.

For the Notify AI/ML model performance, the rApp provides the AI/ML model performance information specified in the Subscribe AI/ML model performance request, as well as the event that triggered the notification if notification event conditions are specified in the Subscribe AI/ML model performance request. If no required AI/ML performance information is specified in the Subscribe AI/ML model performance request, all supported AI/ML model performance information is provided in the Notify AI/ML model performance.

11.8 AI/ML workflow-related use case 8: AI/ML model performance monitoring - rApp consuming performance monitoring service

11.8.1 Overview

This use case defines how an rApp as AI/ML model performance monitoring service Consumer subscribes to AI/ML model performance and receives notifications of AI/ML model performance information.

11.8.2 Background and goal of the use case

An AI/ML model performance monitoring service Consumer can subscribe to AI/ML model performance and receive notifications of AI/ML model performance information.

11.8.3 Entities/resources involved in the use case

1) rApp as AI/ML model performance monitoring service Producer:

   a) supports the functionality to allow an authorized consumer to subscribe to, and receive notifications of, AI/ML model performance information.

2) rApp as AI/ML model performance monitoring service Consumer:

   a) supports the functionality to initiate the procedure to subscribe to, and receive notifications of, AI/ML model performance information.

11.8.4 Solutions

11.8.4.1 Subscribe AI/ML model performance

Table 11.8.4.1-1: Subscribe AI/ML model performance

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	rApp subscribes to AI/ML model performance.	
Actors and Roles	- rApp in the role of AI/ML model performance monitoring service Producer. - rApp in the role of AI/ML model performance monitoring service Consumer.	
Assumptions	- The supported AI/ML model performance information are part of the AI/ML model registration information.	
Preconditions	- The AI/ML model performance monitoring service producer rApp has deployed a registered AI/ML model.	
Begins when	The AI/ML model performance monitoring service Consumer rApp determines the need to monitor the AI/ML model performance of a deployed AI/ML model.	
Step 1 (M)	The AI/ML model performance monitoring service Consumer rApp subscribe to AI/ML model performance with AI/ML model identifier, required AI/ML model performance information (optional), periodicity (optional), event notification conditions (optional), and notification URI as parameters.	
Step 2 (M)	The AI/ML model performance monitoring service Producer rApp checks with SME functions whether the AI/ML model performance monitoring service Consumer is authorized to subscribe to AI/ML model performance.	
Step 3 (M)	The AI/ML model performance monitoring service Producer rApp validates the subscription request.	
Step 4 (M)	The rApp creates the subscription.	
Step 5 (M)	The rApp responds to the request with subscription ID as a parameter.	
Ends when	The subscription is created, and The AI/ML model performance monitoring service Consumer rApp has the subscription ID.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-Performancer-FUN1.	

```
@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 "The AI/ML model performance monitoring \n service Producer rApp" as rApp
participant " The AI/ML model performance monitoring \n service Consumer rApp " as AIML
end box
box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
participant "SME functions" as SME
```

```
end box

AIML -> rApp : Subscribe AI/ML model performance request (AI/ML model identifier, \n AI/ML model
performance information (optional), Periodicity (optional),\n EventNotificationConditions
(optional), Notification URI)
rApp --> SME : <<R1>> AuthZ
note right
Check authorization in
collaboration with SME functions
end note

rApp --> rApp : Validate request
rApp --> rApp : Create subscription
rApp -> AIML: Subscribe AI/ML model performance response (Subscription ID)

@enduml
```

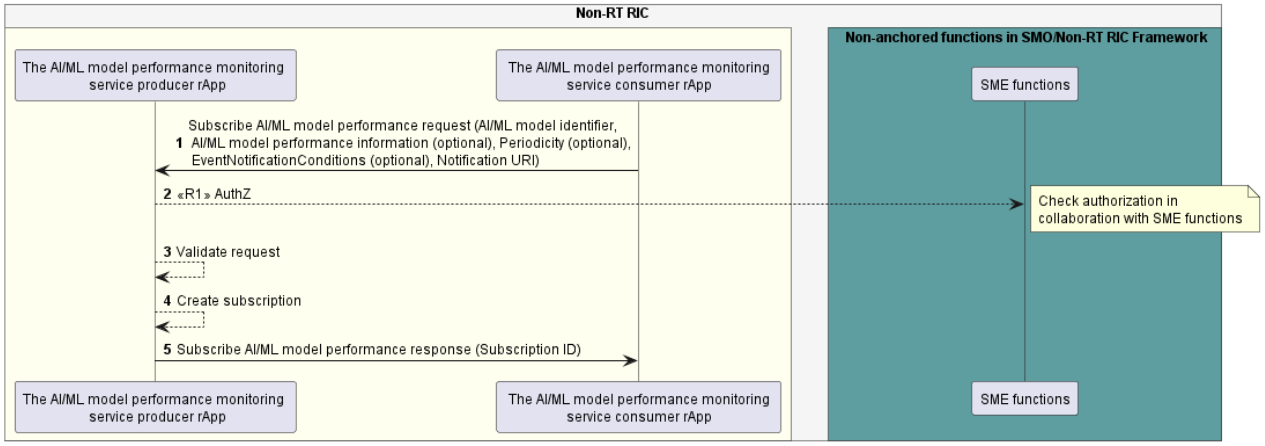

Figure 11.8.4.1-1: Subscribe AI/ML model performance use case flow diagram

11.8.4.2 Notify AI/ML model performance

Table 11.8.4.2-1: Notify AI/ML model performance

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp notifies subscribed consumers of AI/ML model performance information.	
Actors and Roles	- rApp in the role of AI/ML model performance monitoring service Producer. - rApp in the role of AI/ML model performance monitoring service Consumer.	
Assumptions	- The supported AI/ML model performance information are part of the AI/ML model registration information.	
Preconditions	- The rApp has deployed a registered AI/ML model. - The AI/ML model performance monitoring service consumer rApp subscribed to AI/ML model performance.	
Begins when	The rApp determine the need to notify subscribed consumers of the performance information of a deployed AI/ML model.	
Alternative procedure	In the case of event-based notifications, the rApp notifies subscribed consumers of the AI/ML model performance information every time an event is triggered in Step 1.	
Step 1 (M)	The rApp checks the occurrence of the notification events specified in the subscription request.	
Step 2 (M)	The rApp notifies the subscribed consumers of the AI/ML model performance information providing the requested AI/ML model performance information and the event triggered as parameters. If no AI/ML model performance information is specified in the subscription request, all supported AI/ML model performance information is provided.	
Alternative procedure	In the case of periodic notifications, the rApp notifies subscribed consumers of the AI/ML model performance information as per the notification periodicity specified in the subscription.	
Step 3 (M)	The rApp notifies the subscribed consumers of the AI/ML model performance information providing the requested AI/ML model performance information. If no AI/ML model performance information is specified in the subscription request, all supported AI/ML model performance information is provided.	
Ends when	The rApp is able to terminate the subscription or the AI/ML model is no longer deployed.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-Performancer-FUN2.	

```
@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 "AI/ML model performance monitoring \n service Producer rApp" as rApp
participant " The AI/ML model performance monitoring \n service Consumer rApp " as AIML

end box

Loop Periodicity of notifications specified in the subscription request
    Alt Event based notifications
    rApp --> rApp : Check if specified event is triggered
    rApp -> AIML : Notify AI/ML model performance (AI/ML model identifier, \n AI/ML model
performance information,EventNotificationType)
    else Periodical notifications
        rApp -> AIML : Notify AI/ML model performance (AI/ML model identifier, \n AI/ML model
performance information)
end
end

@enduml
```

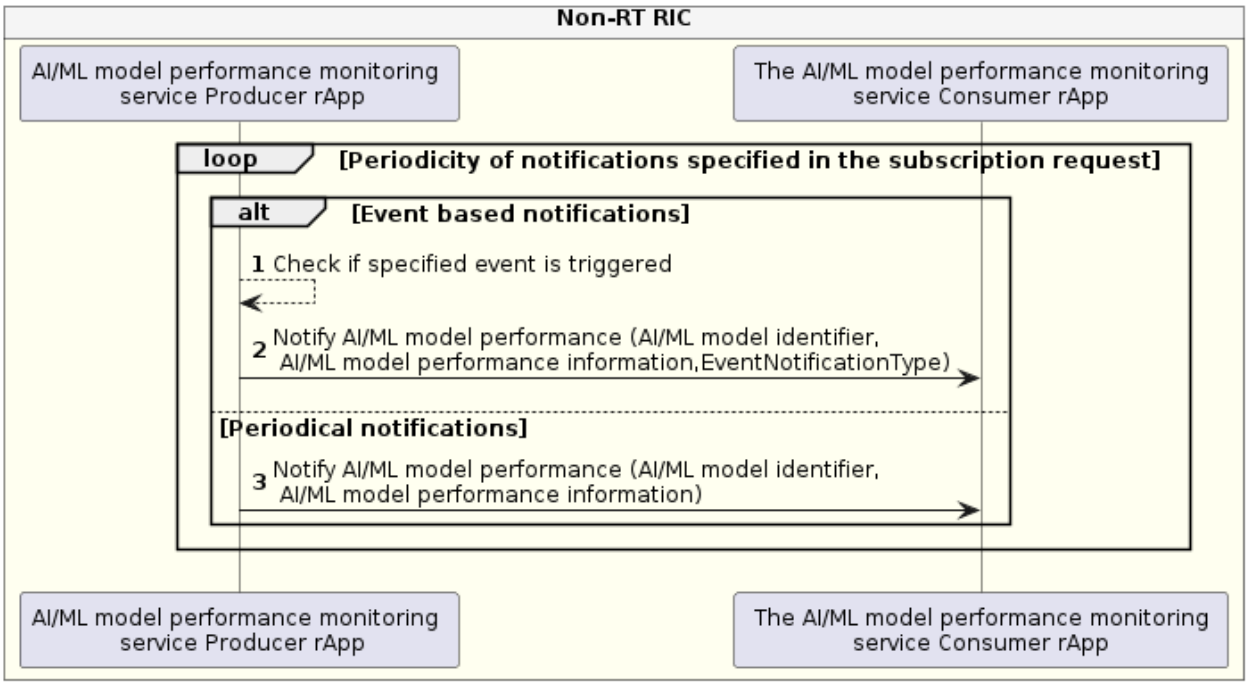

Figure 11.8.4.2-1: Notify AI/ML model performance use case flow diagram

11.8.5 Required data

For the Subscribe AI/ML model performance request, the AI/ML model performance monitoring service Consumer rApp provides the AI/ML model identifier and notification URI. In the Subscribe AI/ML model performance request, the AI/ML model performance monitoring service Consumer rApp optionally provides the required AI/ML model performance information, Notification event conditions upon which the Notify AI/ML model performance is triggered, and periodicity of the notification.

For the Subscribe AI/ML model performance response, the AI/ML model performance monitoring service Producer rApp provides the Subscription ID.

For the Notify AI/ML model performance, the AI/ML model performance monitoring service producer rApp provides the AI/ML model performance information specified in the Subscribe AI/ML model performance request, as well as the event that triggered the notification if notification event conditions are specified in the Subscribe AI/ML model performance request. If no required AI/ML performance information is specified in the Subscribe AI/ML model performance request, all supported AI/ML model performance information is provided in the Notify AI/ML model performance.

11.9 AI/ML workflow-related use case 9: AI/ML model training capability registration and deregistration

11.9.1 Overview

This use case defines how an rApp registers and deregisters AI/ML model training capability.

11.9.2 Background and goal of the use case

The register AI/ML model training capability and deregister AI/ML model training capability are optional procedures defined as part of the AI/ML model training capability registration service in R1GAP [1].

11.9.3 Entities/resources involved in the use case

1) AI/ML workflow functions:

   a) support functionality to allow an rApp to register and deregister AI/ML model training capability;

   b) support validation of AI/ML model training capability information.

2) rApp:

   a) initiates the procedure to register and deregister AI/ML model training capability.

11.9.4 Solutions

11.9.4.1 Register AI/ML model training capability

Table 11.9.4.1-1: AI/ML Model training capability registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp registers an AI/ML model training capability.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure service Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure services. - The rApp is registered as producer of AI/ML model training with SME, it intends to register its training capability in AI/ML workflow functions.	
Begins when	The rApp determines the need to register an AI/ML model training capability.	
Step 1 (M)	The rApp requests the AI/ML workflow functions to register an AI/ML model training capability by providing the rAppId and AI/ML model training capability information.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the rApp is registered as producer of AI/ML model training service.	
Step 3 (M)	The AI/ML workflow functions register the rApp's AI/ML model training capability.	
Step 4 (M)	The AI/ML workflow functions respond to rApp with successful AI/ML model training capability registration along with AI/ML model training capability registration ID.	
Ends when	The rApp was able to register an AI/ML model training capability.	
Exceptions	n/a	
Post Conditions	The AI/ML model training capability is registered. The rApp can query, update or delete the AI/ML model training capability registration.	
Traceability	REQ-R1-AI/ML-Registertraincap-FUN1.	

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

box "Non-RT RIC" #whitesmoke
    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox

    box #ivory
    participant "Training service Producer rApp" as src
```

```
    endbox
endbox

src -> ML: <<R1>> Register AI/ML model training capability request (rAppId, AI/ML model training
capability information)
ML -> ML: Authz
ML -> ML: register AI/ML model training capability
ML -> src: <<R1>> Register AI/ML model training capability response (AI/ML model training capability
registration ID)
@enduml
```

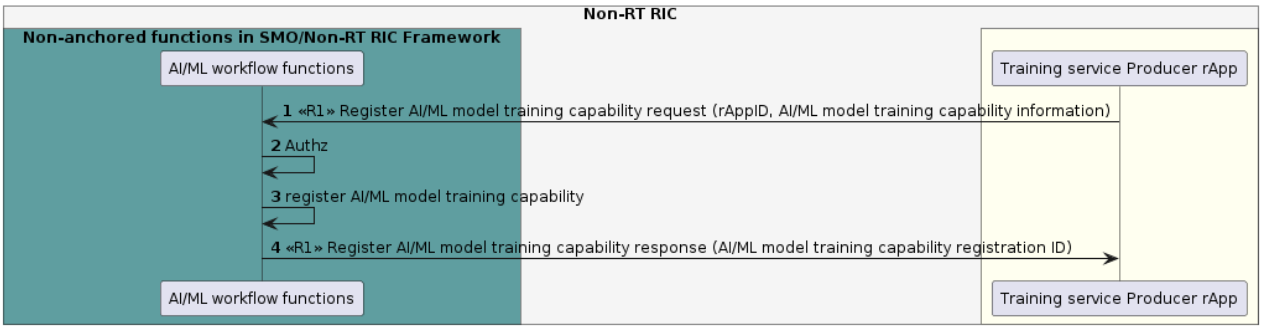

Figure 11.9.4.1-1: Register AI/ML model training capability use case flow diagram

11.9.4.2 Deregister AI/ML model training capability

Table 11.9.4.2-1: AI/ML Model training capability deregistration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp deregisters an AI/ML model training capability.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure service Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	The rApp is not producing any AI/ML model training capability.	
Preconditions	The rApp has registered an AI/ML model training capability. The rApp is authorized to access the AI/ML model management and exposure services.	
Begins when	The rApp determines the need to deregister an AI/ML model training capability that it no longer intends to produce.	
Step 1 (M)	The rApp requests AI/ML workflow functions to deregister an AI/ML model training capability by providing rAppId and AI/ML model training capability registration ID.	
Step 2 (M)	The AI/ML workflow functions remove the registration of the rApp as producer of AI/ML model training capability.	
Step 3 (M)	The AI/ML workflow functions respond to the rApp with successful deregistration.	
Ends when	The rApp was able to deregister the AI/ML model training capability.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AI/ML-Registertraincap-FUN1.	

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

box "Non-RT RIC" #whitesmoke
    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
```

```
    endbox

    box #ivory
    participant "Training service Producer rApp" as src
    endbox
endbox

src -> ML: <<R1>> Deregister AI/ML model training capability (rAppId, AI/ML model training
capability registration ID)
ML -> ML: Deregister AI/ML model training capability
ML -> src: <<R1>> Deregister AI/ML model training capability response
@enduml
```

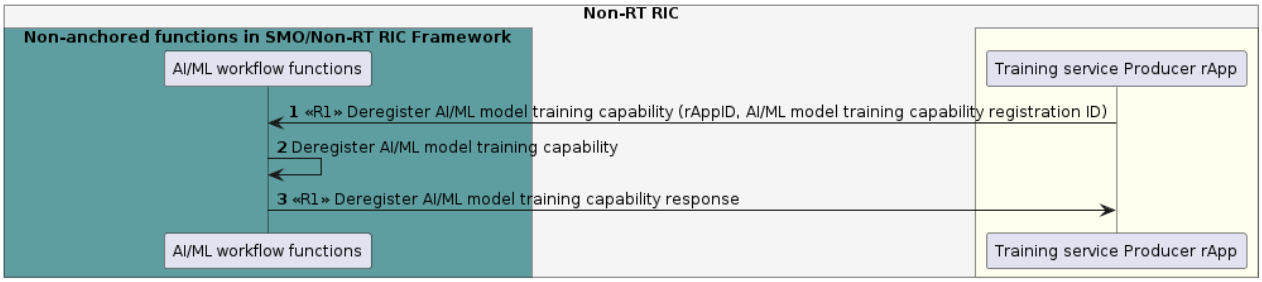

Figure 11.9.4.2-1: Deregister AI/ML model training capability use case flow diagram

11.9.4.3 Update AI/ML model training capability registration

Table 11.9.4.3-1: Update AI/ML model training capability registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp updates the registration of an AI/ML model training capability.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure services Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure services Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure services for updating an AI/ML model training capability registration. - The rApp has registered an AI/ML model training capability.	
Begins when	The AI/ML model training service producer rApp determines the need to update the registration of an AI/ML model training capability.	
Step 1 (M)	The AI/ML model training service producer rApp requests to update the registration of an AI/ML model training capability by providing rAppId, AI/ML model training capability registration ID, updated model training capability registration information, etc.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to update the registration.	
Step 3 (M)	The AI/ML workflow functions validate the update AI/ML model training capability registration request.	
Step 4 (M)	The AI/ML workflow functions update the registration of the AI/ML model training capability.	
Step 5 (M)	The AI/ML workflow functions respond to the rApp with successful update AI/ML model training capability registration response.	
Ends when	The rApp was able to update the registration of the AI/ML model training capability.	
Exceptions	n/a	
Post Conditions	The rApp can query, update or delete the AI/ML model training capability registration.	
Traceability	REQ-R1-AI/ML-Registertraincap-FUN1.	

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

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

  box "Non-anchored functions \n in SMO/Non-RT RIC FWK" #cadetBlue
    participant "AI/ML workflow functions" as aif
  endbox
endbox

autonumber
app -> aif: <<R1>> Update model training capability registration request \n (rAppId, AI/ML model
training capability registration identifier, \n updated model training capability registration
information)
activate aif
aif --> aif: AuthZ
note right
Check authorization in
collaboration with SME functions
end note
aif --> aif: Validate
aif --> aif: Update model training capability\n registration information
app <- aif: <<R1>> Update model training capability registration response
deactivate aif
@enduml
```

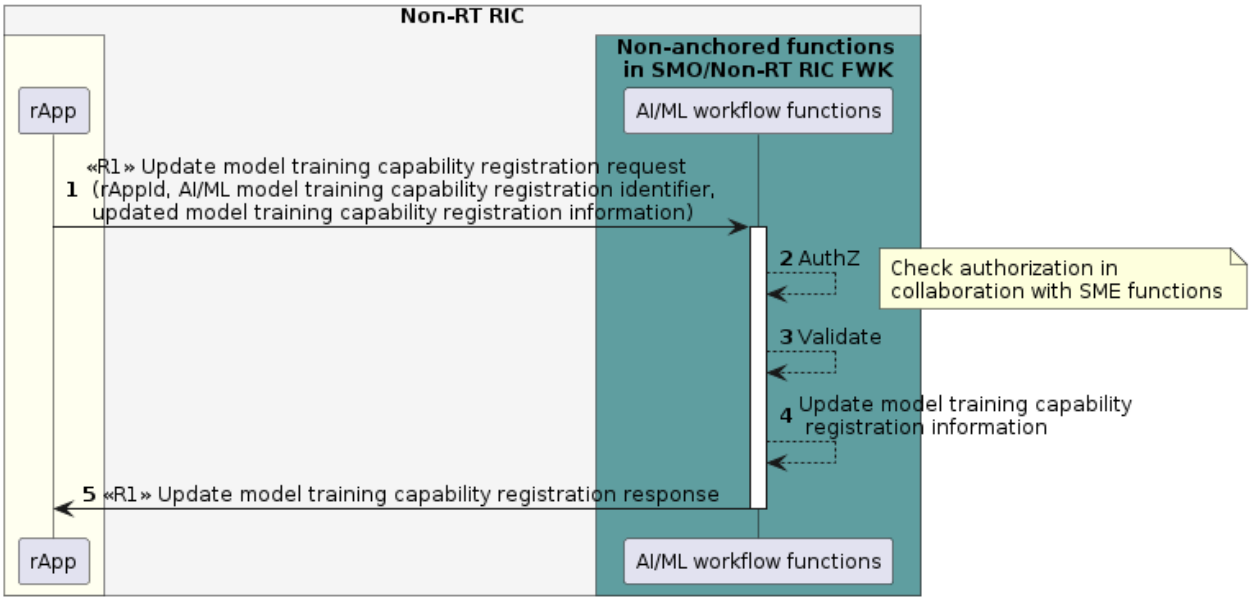

Figure 11.9.4.3-1: Update registration of an AI/ML training capability model use case flow diagram

11.9.4.4 Query AI/ML model training capability registration

Table 11.9.4.4-1: Query AI/ML model training capability registration

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp queries an AI/ML model training capability registration.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure services Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure services Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure services for registering an AI/ML model. - The AI/ML model training capability has been registered by the rApp.	
Begins when	The AI/ML model training service producer rApp determines the need to query the registration of a registered AI/ML model training capability.	
Step 1 (M)	The AI/ML model training service producer rApp requests to query the registration of an AI/ML model training capability by providing rAppId and AI/ML model training capability registration ID.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to query the AI/ML model training capability registration.	
Step 3 (M)	The AI/ML workflow functions look up the information of queried AI/ML model training capability.	
Step 4 (M)	The AI/ML workflow functions respond to rApp with AI/ML model training capability information.	
Ends when	The AI/ML model training service producer rApp was able to query the AI/ML model training capability registration.	
Exceptions	n/a	
Post Conditions	The rApp can query, update, or delete the AI/ML model training capability registration.	
Traceability	REQ-R1-AI/ML-Registertraincap-FUN1.	

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

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

src -> ML: <<R1>> Query AI/ML model training capability request (rAppId, AI/ML model training
capability registration ID)
ML -> ML: Authz
note right
Check authorization in
collaboration with SME functions
end note
ML -> ML: Look up AI/ML model training capability registration
ML -> src: <<R1>> Query AI/ML model training capability response (AI/ML model training capability
information)
@enduml
```

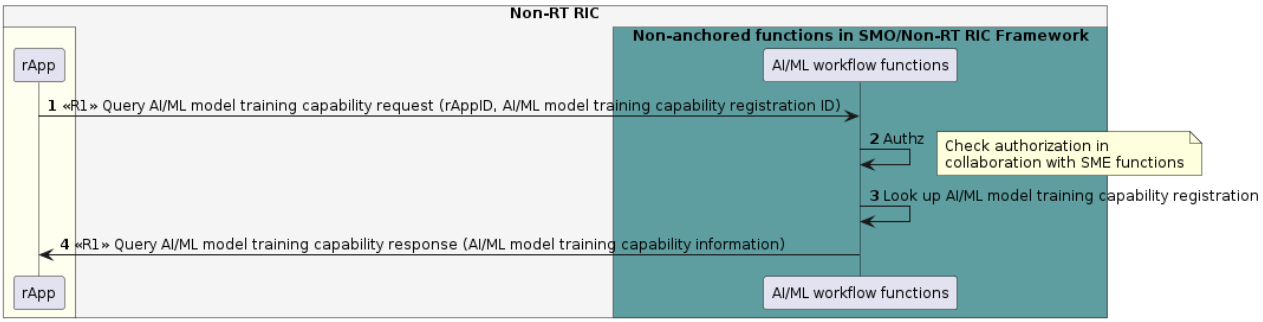

Figure 11.9.4.4-1: Update registration of an AI/ML training capability model use case flow diagram

11.9.5 Required data

The AI/ML model training capability information includes but not limited to supported training frameworks, supported library packages, supported library versions, resource information.

For registering an AI/ML model training capability, the rApp provides the rAppId and AI/ML model training capability information. On successful registration, AI/ML workflow functions provide an AI/ML model training capability registration ID for the registration of the AI/ML model training capability information.

For deregistering an AI/ML model training capability, the rApp provides the rAppId and the AI/ML model training capability registration ID.

For updating the registration of an AI/ML model training capability, the rApp needs to provide the rAppId, the AI/ML model training capability registration ID and the updated AI/ML model training capability information or the modified part of the AI/ML model training capability information.

For querying the registration of an AI/ML model training capability, the rApp needs to provide the rAppId and the AI/ML model training capability registration ID.

11.10 AI/ML workflow-related use case 10: AI/ML model inference - AI/ML workflow functions producing AI/ML inference

11.10.1 Overview

This use case enables an rApp to request and cancel an inference for an AI/ML model.

11.10.2 Background and goal of the use case

The request and cancel AI/ML model inference procedures are defined as part of the AI/ML workflow services in R1GAP [1].

11.10.3 Entities/resources involved in the use case

1) AI/ML workflow functions in the role of AI/ML model inference service Producer:

   a) support functionality allowing rApps to request, and cancel the inference of a registered AI/ML Model.

2) rApp in the role of AI/ML model inference service Consumer:

   a) initiates the procedure to request and cancel the inference of an AI/ML model.

11.10.4 Solutions

11.10.4.1 Request AI/ML model inference

Table 11.10.4.1-1: Request AI/ML model inference use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML inference services Consumer requests inference for an AI/ML model.	
Actors and Roles	- AI/ML inference services Consumer in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML inference services Consumer is deployed, authenticated, and authorized to consume AI/ML inference services.	
Begins when	The AI/ML inference services Consumer determines the need to initiate the inference an AI/ML model.	
Step 1 (M)	The AI/ML inference services Consumer requests the AI/ML workflow functions to infer an AI/ML model providing rAppId and model identifier.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the AI/ML inference services Consumer is authorized to request inference for an AI/ML model.	
Step 3 (M)	The AI/ML workflow functions validate the request.	
Step 4 (M)	The AI/ML workflow functions respond to the AI/ML inference services Consumer with inference job identifier as a parameter.	
Ends when	The AI/ML inference services Consumer is able to obtain the inference job identifier.	
Exceptions	n/a	
Post Conditions	The inference job exists, and the inference service Consumer can cancel the AI/ML model inference.	
Traceability	REQ-R1-AIML-inference-FUN1.	

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

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

rApp -> ML: <<R1>> Request AI/ML model inference request \n (rAppId, AI/ML model identifier)
ML -> ML: AuthZ

note right
Check authorization in
Collaboration with SME functions
end note

ML-> ML :validate
ML -> rApp: <<R1>> Request AI/ML model inference response \n (inference job identifier)
@enduml
```

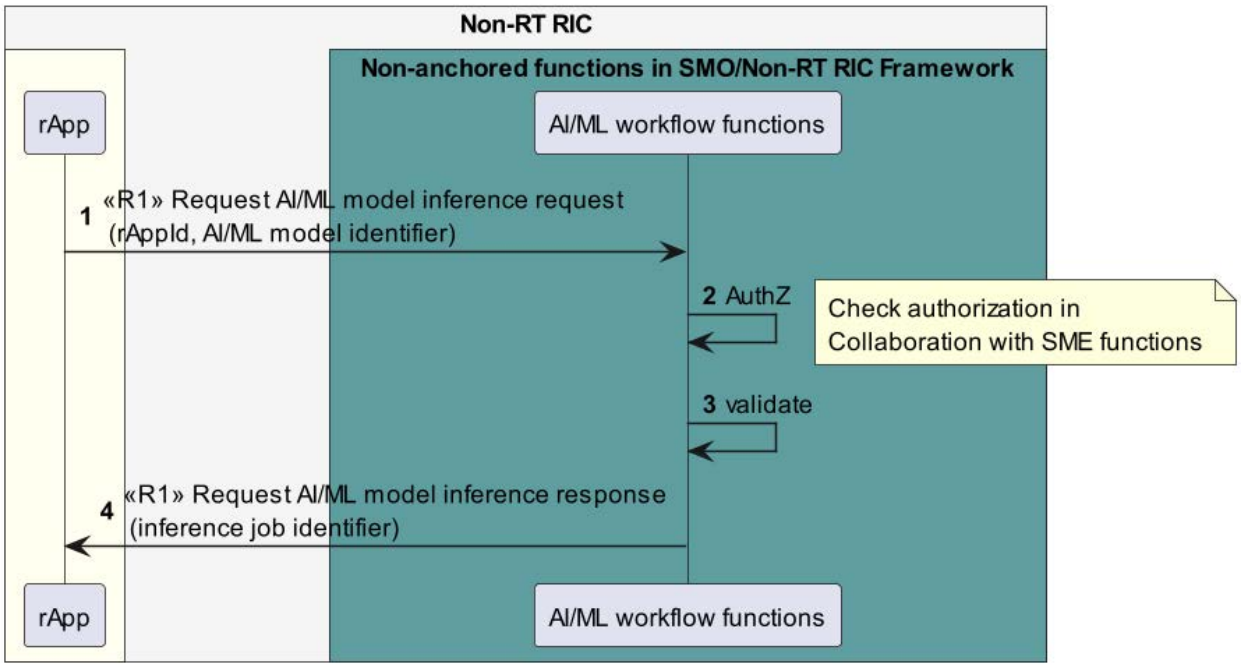

Figure 11.10.4.1-1: Request AI/ML model inference use case flow diagram

11.10.4.2 Cancel AI/ML model inference

Table 11.10.4.2-1: Cancel AI/ML model inference use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML inference services Consumer cancels the AI/ML model inference that it has previously requested.	
Actors and Roles	- AI/ML inference services Consumer in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	n/a	
Preconditions	- The AI/ML inference services Consumer is deployed, authenticated, and authorized to consume AI/ML inference services. - The inference job exists, and the AI/ML inference Consumer is aware of the job identifier.	
Begins when	The AI/ML inference services Consumer determines the need to cancel the inference of an AI/ML model.	
Step 1 (M)	The AI/ML inference services Consumer requests the AI/ML workflow functions to cancel the inference for an AI/ML model providing rAppId and inference job identifier.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the AI/ML inference services Consumer is authorized to cancel the inference job for an AI/ML model.	
Step 3 (M)	The AI/ML workflow functions cancel the inference job.	
Step 4 (M)	The AI/ML workflow functions respond to the AI/ML inference services Consumer rApp with cancellation of inference job.	
Ends when	The AI/ML inference service job has been cancelled.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-inference-FUN1.	

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

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

rApp -> ML: <<R1>> Cancel AI/ML model inference request \n (rAppId,inference job identifier)
ML -> ML: AuthZ

note right
Check authorization in
Collaboration with SME functions
end note

ML -> ML : Stop inference job
ML -> rApp: <<R1>> Cancel AI/ML model inference response
@enduml
```

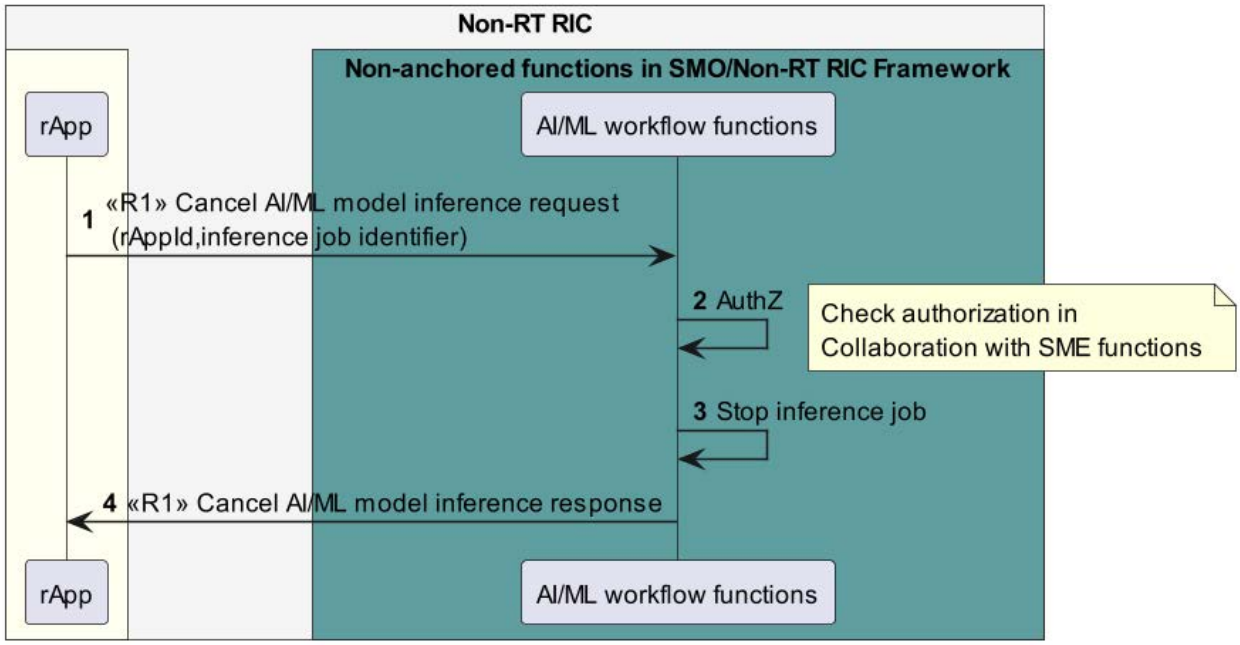

Figure 11.10.4.2-1: Cancel AI/ML model inference use case flow diagram

11.10.5 Required data

For creating AI/ML model inference job, the AI/ML inference services Consumer needs to provide its rAppId and the model identifier. The AI/ML workflow functions respond with the creation of inference job by providing the inference job identifier.

For cancellation of an AI/ML inference job, the AI/ML inference services Consumer needs to provide its rAppId and the inference job identifier.

11.11 AI/ML workflow-related use case 11: AI/ML model change subscription

11.11.1 Overview

This use case defines how an rApp subscribes to registered AI/ML models. It enables an rApp to subscribe to and unsubscribe from notifications regarding changes in the registered AI/ML models.

11.11.2 Background and goal of the use case

The subscribe AI/ML models changes procedure, unsubscribe AI/ML models changes procedure and notify AI/ML models changes procedure are defined as part of the AI/ML workflow services in R1GAP [1].

11.11.3 Entities/resources involved in the use case

1) AI/ML workflow functions:

   a) support functionality to allow an rApp to subscribe and unsubscribe the registered AI/ML model changes;

   b) support functionality to notify rApp about changes of registered AI/ML model;

   c) support validation of selection criteria.

2) rApp:

   a) initiates the procedure to subscribe and unsubscribe registered AI/ML model changes;

   b) support functionality to receive notification regarding changes of subscribed AI/ML model.

11.11.4 Solutions

11.11.4.1 Subscribe AI/ML model changes

Table 11.11.4.1-1: Subscribe AI/ML model changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp subscribes to notifications regarding changes of AI/ML models.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure service Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure service. - The rApp is aware of the AI/ML model identifier of the AI/ML model.	
Begins when	The rApp determines the need to subscribe to notifications regarding changes of registered AI/ML models.	
Step 1 (M)	The rApp requests the AI/ML workflow functions to subscribe to notifications regarding AI/ML model changes with rAppId and AI/ML model identifier selection criteria.	
Step 2 (M)	The AI/ML workflow functions receive the AI/ML model subscription request, and check whether the rApp is authorized to subscribe to identified AI/ML models.	
Step 3 (M)	The AI/ML workflow functions validate the subscribe AI/ML model request.	
Step 4 (M)	The AI/ML workflow functions establish the AI/ML model subscription.	
Step 5 (M)	The AI/ML workflow functions send back a response with a subscription identifier matching the request in Step 1 to rApp.	
Ends when	The rApp was able to subscribe to notifications regarding AI/ML model changes.	
Exceptions	n/a	
Post Conditions	- The rApp can receive notifications when changes are made to any of the AI/ML models that matches AI/ML model identifier selection criteria. - The rApp can unsubscribe from the notifications.	
Traceability	n/a	
NOTE: AI/ML model identifier includes model name, model version and artifact version. AI/ML model identifier selection criteria could be upper and/or lower bounds of model version and artifact version number, or list of individual model version and artifact version numbers, etc. Selection criteria is used to identify AI/ML models to be subscribed, which can be applied to part, or all of AI/ML model identifier components.		

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

box "Non-RT RIC" #whitesmoke
    box #ivory
    participant "rApp" as rApp
    endbox
    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

rApp -> ML: <<R1>> Subscribe AI/ML model change request (rAppId, \n AI/ML model identifier selection
criteria)
ML -> ML: Authz
note right
Check authorization in
collaboration with SME functions
```

```
end note
ML -> ML: Validate
ML -> ML: Create subscription
ML -> rApp: <<R1>> Subscribe AI/ML model change response (subscription identifier)
@enduml
```

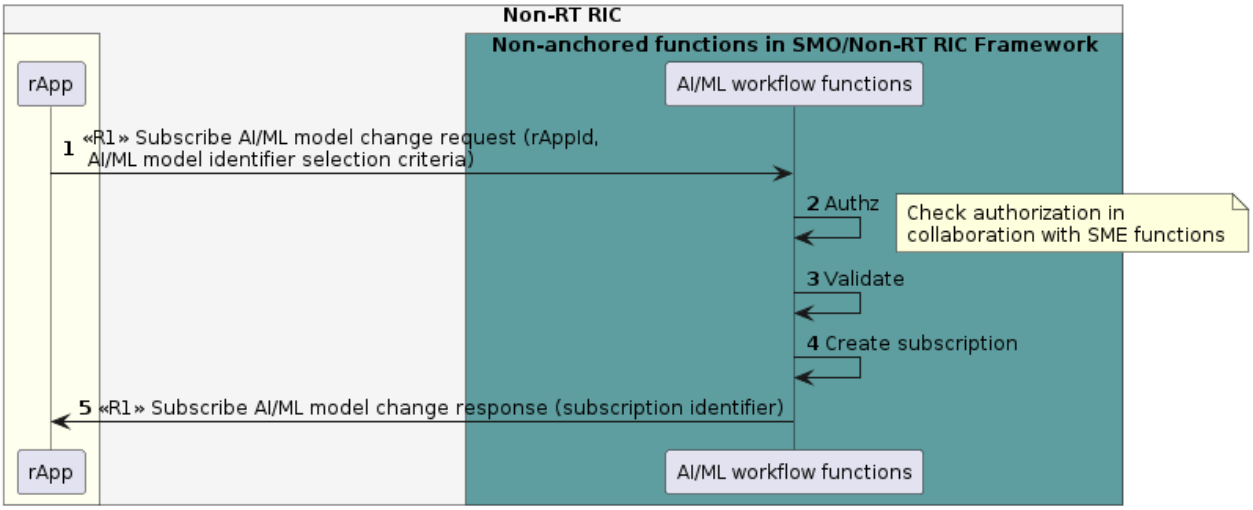

Figure 11.11.4.1-1: Subscribe AI/ML model changes use case flow diagram

11.11.4.2 Notify AI/ML model changes

Table 11.11.4.2-1: Notify AI/ML model changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp receives a notification regarding a change of a registered AI/ML model.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure service Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure service. - The rApp is subscribed to notifications.	
Begins when	The AI/ML workflow functions determine to send a notification regarding the change of a subscribed AI/ML model to the subscribing rApp.	
Step 1 (M)	The AI/ML workflow functions send a notification to the subscribing rApp with AI/ML model identifier and available change details.	
Ends when	The rApp was able to receive the notification.	
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
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
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  "AI/ML workflow functions" as ML
    endbox
endbox

ML -> rApp: <<R1>> Notify AI/ML model changes (AI/ML model identifier, available change details)
@enduml
```

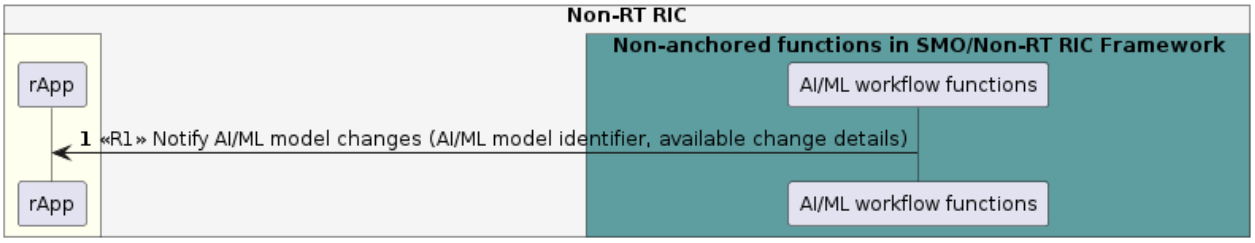

Figure 11.11.4.2-1: Notify AI/ML model changes use case flow diagram

11.11.4.3 Unsubscribe AI/ML model changes

Table 11.11.4.3-1: Unsubscribe AI/ML model changes

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp unsubscribes from notifications.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure service Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	n/a	
Preconditions	- The rApp is authorized to access the AI/ML model management and exposure service. - The rApp is subscribed to notifications regarding AI/ML model changes.	
Begins when	The rApp determines the need to unsubscribe from notifications regarding changes of registered AI/ML model.	
Step 1 (M)	The rApp requests the AI/ML workflow functions to unsubscribe from notifications regarding AI/ML model changes by providing the rAppId and subscription identifier.	
Step 2 (M)	The AI/ML workflow functions check whether the rApp is authorized to unsubscribe from notifications regarding AI/ML model changes.	
Step 3 (M)	The AI/ML workflow functions cancel the subscription.	
Step 4 (M)	The AI/ML workflow functions respond to the request.	
Ends when	The rApp was able to unsubscribe from notifications regarding changes of registered AI/ML model.	
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
skinparam SequenceGroupBackgroundColor Transparent
skinparam SequenceGroupBodyBackgroundColor Transparent
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  "AI/ML workflow functions" as ML
    endbox
endbox
```

```
rApp -> ML: <<R1>> Unsubscribe AI/ML model change request (rAppId, subscription identifier)
ML -> ML: Authz
note right
Check authorization in
collaboration with SME functions
end note
ML -> ML: Cancel subscription
ML -> rApp: <<R1>> Unsubscribe AI/ML model change response
@enduml
```

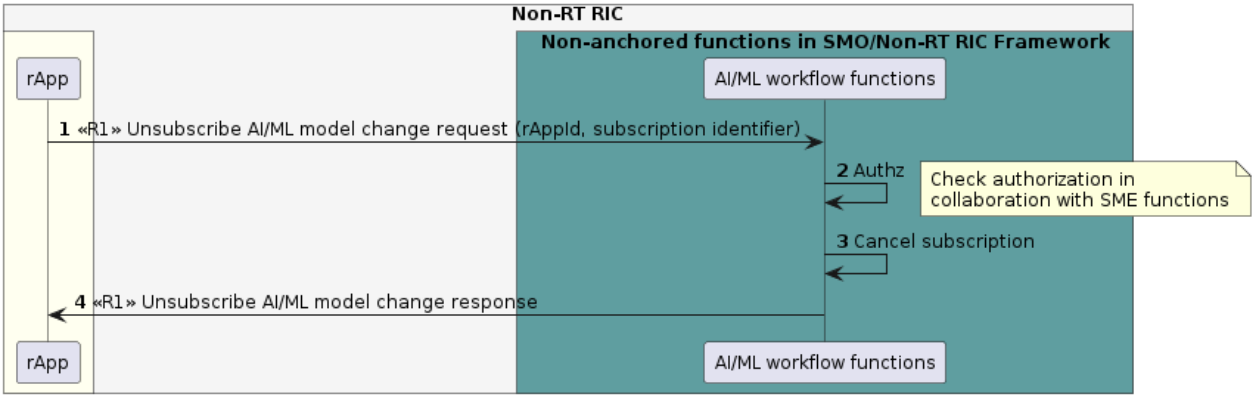

Figure 11.11.4.3-1: Unsubscribe AI/ML model changes use case flow diagram

11.11.5 Required data

For subscription to notifications regarding changes of registered AI/ML model, the rApp provides the rAppId and the AI/ML model identifier selection criteria. The AI/ML workflow functions send back a subscription identifier in response.

The AI/ML workflow functions send notifications to the subscribing rApp by providing the AI/ML model identifier(s) and available change details of AI/ML model(s).

For unsubscribing from notifications regarding changes of registered AI/ML model, the rApp needs to send the rAppId and AI/ML model subscription identifier.

11.12 AI/ML workflow-related use case 12: Query of AI/ML Model inference capabilities - AI/ML workflow functions producing AI/ML inference

11.12.1 Overview

This use case enables an rApp to query the inference capability information of an AI/ML model.

11.12.2 Background and goal of the use case

The query AI/ML model inference capabilities procedure are defined as part of the AI/ML workflow services in R1GAP [1].

11.12.3 Entities/resources involved in the use case

1) AI/ML workflow functions in the role of AI/ML model inference service Producer:

    a) Support functionality allowing rApps to query the inference capability of an AI/ML Model.

2) rApp in the role of AI/ML model inference service Consumer:

   a) Initiates the procedure to query the inference capability of an AI/ML model.

11.12.4 Solutions

11.12.4.1 Query AI/ML model inference capability

Table 11.12.4.1-1: Query AI/ML model inference capability use case

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The AI/ML inference services Consumer retrieves the AI/ML model inference capability information.	
Actors and Roles	- AI/ML inference services Consumer in the role of Service Consumer. - The AI/ML workflow functions in the role of Service Producer.	
Assumptions	Service Consumer is authorized to request inference capability information of a registered AI/ML model.	
Preconditions	- The AI/ML inference services Consumer is deployed, authenticated, and authorized to consume AI/ML inference services.	
Begins when	The AI/ML inference services Consumer determines the need to query the inference capability information of a registered AI/ML model.	
Step 1 (M)	The AI/ML inference services Consumer requests the AI/ML workflow functions to provide inference capability information by passing on the query parameters such as model identifier and associated model information.	
Step 2 (M)	The AI/ML workflow functions check with SME functions whether the AI/ML inference services Consumer is authorized to request the inference capability information for a registered AI/ML model.	
Step 3 (M)	The AI/ML workflow functions respond to the AI/ML inference services Consumer rApp with the requested inference capability information that matches the query criteria.	
Ends when	The AI/ML model inference capability information has been retrieved.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AIML-inference-FUN2.	

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

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

    box " Non-anchored functions in SMO/Non-RT RIC Framework " #cadetBlue
    participant  "AI/ML workflow functions" as ML
    endbox
endbox

rApp -> ML: <<R1>> Query AI/ML model inference capability request \n (rAppId, query criteria)
ML -> ML: AuthZ

note right
Check authorization in
Collaboration with SME functions
end note

ML -> rApp: <<R1>> Query AI/ML model inference capability response \n (inference capability
information)
@enduml
```

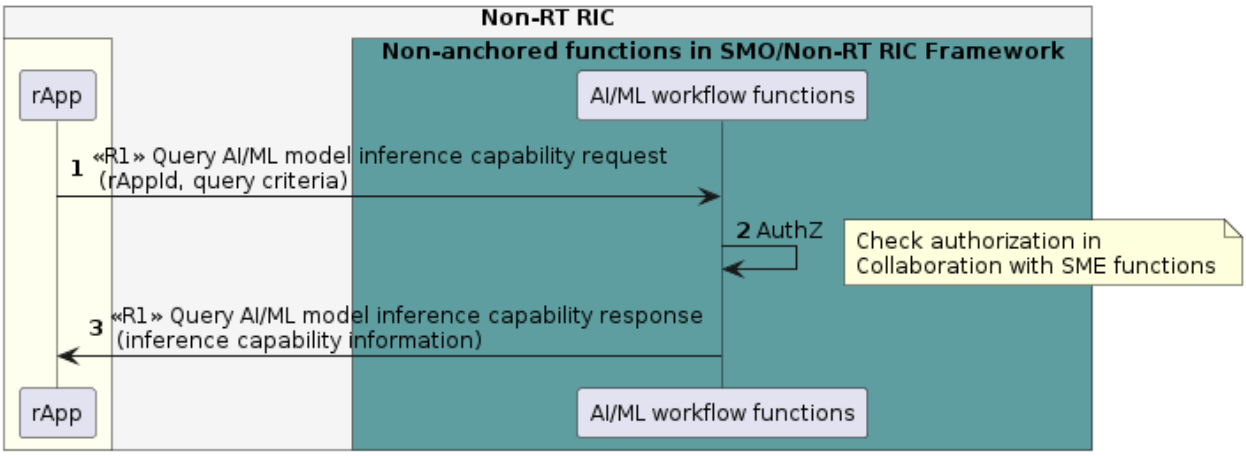

Figure 11.12.4.1-1: Query AI/ML model inference capability use case flow diagram

11.12.5 Required data

For querying of an AI/ML model inference capability information, the AI/ML inference services Consumer needs to provide its rAppId and the query criteria (including the model identifier and model information). The AI/ML workflow functions respond with inference capability information such as model identifier or inference latency.

11.13 AI/ML Workflow-related use case 13: Query of AI/ML Model training capability

11.13.1 Overview

This use case defines how an rApp queries AI/ML model training capability.

11.13.2 Background and goal of the use case

The query AI/ML model training capability is an optional procedure defined as part of the AI/ML model training capability query service in R1GAP [1].

11.13.3 Entities/resources involved in the use case

1) AI/ML workflow functions in the role of AI/ML model management and exposure service producer:
   a) Supports functionality allowing rApps to query the registered AI/ML model training capability;
2) rApp in the role of AI/ML model management and exposure service consumer:
   a) Initiates the procedure to query the registered AI/ML model training capability.

11.13.4 Solutions

11.13.4.1 Query AI/ML model training capability

Table 11.13.4.1-1: Query AI/ML model training capability.

Use Case Stage	Evolution / Specification	<<Uses>> Related use
Goal	The rApp queries registered AI/ML model training capability.	
Actors and Roles	- rApp in the role of AI/ML model management and exposure service Consumer. - AI/ML workflow functions in the role of AI/ML model management and exposure service Producer.	
Assumptions	n/a	
Preconditions	The rApp is authorized to access the AI/ML workflow services. At least one AI/ML model training capability is registered with the AI/ML workflow functions.	
Begins when	The rApp determines the need to query the registered AI/ML model training capability.	
Step 1 (M)	The rApp requests the AI/ML workflow functions for the information on the AI/ML model training capability by providing rAppId and selection criteria.	
Step 2 (M)	The AI/ML workflow functions check if the rApp is authorized to query the registered AI/ML model training capability.	
Step 3 (M)	The AI/ML workflow functions look up the information of queried AI/ML model training capability.	
Step 4 (M)	The AI/ML workflow functions provide the information about the registered AI/ML model training capability.	
Ends when	The AI/ML model training capability has been received.	
Exceptions	n/a	
Post Conditions	n/a	
Traceability	REQ-R1-AI/ML-Registertraincap-FUN1.	

```
@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
   endbox
  box " Non-anchored functions in SMO/Non-RT RIC Framework" #cadetBlue
    participant " AI/ML workflow functions " as aif
     endbox
rApp ->aif:<<R1>> Query AI/ML model training capability request\n(rAppId, Selection criteria)
activate aif
aif --> aif:AuthZ
note right
Check authorization in collaboration
with SME functions
end note
aif --> aif: Look up AI/ML model training capability registration
aif -> rApp :<<R1>> Query AI/ML model training capability response\n(AI/ML model Trainer rAppIds,
AI/ML model training capability information)
deactivate aif
@enduml
```

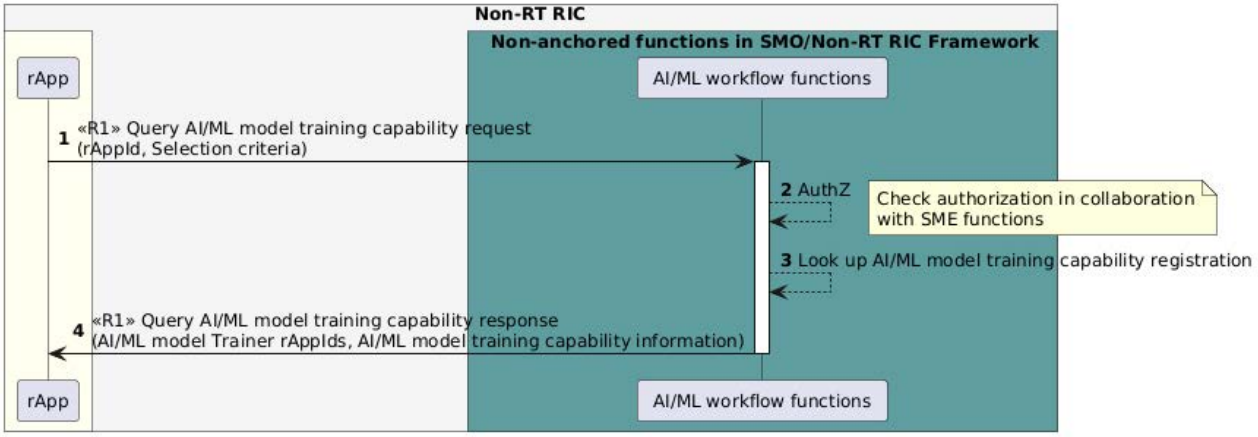

Figure 11.13.4.1-1: Query AI/ML model training capability use case flow diagram

11.13.5 Required data

For the Query AI/ML model training capability request, the rApp provides the rAppId and selection criteria.

The AI/ML workflow functions respond with information which includes the AI/ML model Trainer rAppIds and AI/ML model training capability information that match the filtering criteria. The AI/ML model training capability information contains training platform information and training resource information.

Annex A (informative): Change history

Date	Version	Information about changes
13.03.2025	V10.00	Added use cases for AI/ML model training capability registration and data job status.
21.11.2024	V09.00	Added use cases for AIML inference capability information Query.
18.07.2024	V08.00	Added use cases for AI/ML model training capability registration, Query for data subscription, AI/ML model inference use case for create and delete and updated the SME subscription use case.
20.04.2024	V07.00	Added the use cases for AI/ML performance monitoring use cases and updated the document with Producer and Consumer roles.
20.11.2023	V06.00	Added the use cases and requirements to Data management and exposure services, A1 related services, AI/ML model workflow services, updated the data type to DME type.
29.07.2023	V05.00	Published as Final version 05.00.
24.03.2023	V04.00	Published as Final Version 04.00.
10.11.2022	V03.00	Published as Final Version 03.00.
29.07.2022	V02.00	Published as Final version 02.00.
01.04.2022	V01.00	Published as Final version 01.00.

History

Version	Date	Status
V10.0.0	February 2026	Publication