

SPEC-CRE

Last changed:	20.11.2025 10:03:45
Version:	3.0.0-rc.6
Creator:	VDV ETS

Table of Contents

1	ION Actors	3
1.1	Initiator	3
1.2	Processor	3
1.3	Central routing engine	3
2	System Components and Interfaces	5
2.1	ION Public Key Infrastructure	5
2.1.1	Certificate Retrieval Service	5
2.2	Central routing engine	6
2.2.1	Central routing engine conceptual interface	6
2.3	Initiator system	6
2.3.1	Initiator interface	6
2.4	Processor system	6
2.4.1	Processor interface	6
3	Service availability notification	6
3.1	Overview	6
3.2	Use Cases	6
3.2.1	Handle request to set service as available for a participant	7
3.2.2	Handle request to set service as unavailable for a participant	9
4	Conceptual Routing	11
4.1	Overview	11
4.2	Use Cases	11
4.2.1	Process asynchronous message	11
4.2.2	Process synchronous message	13
5	Store & Forward	15
5.1	Overview	15
5.2	Use Cases	15
5.2.1	Deliver queued messages in scheduler	15
5.2.2	Deliver queued messages to service	17
5.2.3	Store asynchronous message	18
5.2.4	Discard queued messages	20
6	Monitoring and notification	22
6.1	Overview	22
6.2	Use Cases	22
6.2.1	Notify about discarded messages	22



1 ION Actors

This package contains the conceptual actors ([Initiator](#) and [Processor](#)) and the real actor [Central routing engine](#) which are used in the ION specification.

1.1 Initiator

Conceptual actor as a placeholder for all actors which initiate a use case that involves message exchanges via the ION.

The initiator is the one that starts a use case. Without sending an initial request, the whole use case would not be performed.

Initially, the initiator sends an [ION request message](#) to the [Processor](#).

In a second step, it receives a message synchronously ([synchronous ION response message](#)) or asynchronously ([asynchronous ION response message](#)).

Note: The initiator always remains the logical initiator, even if - due to the nature of distributed systems and asynchronous processes - the client/server role might change during the runtime of a use case.

1.2 Processor

Conceptual actor as a placeholder for all actors which react to an initial request sent by an [Initiator](#) in a use case that involves message exchanges via the ION.

Depending on the type of the request, it either sends only a [BusinessAcknowledgement](#) response or enhances it with further [Response business data](#).

Note: The processor always remains the logical processor, even if - due to the nature of distributed systems and asynchronous processes - the client/server role might change during the runtime of a use case.

1.3 Central routing engine

Actor which stands for the central routing engine (CRE).

The central routing engine routes messages from an [Initiator](#) to a [Processor](#) and back.

The initiator must pass the routing information, which is defined in the [IonRoutingHeader](#). The CRE uses these parameters stored in this header information to find the configured endpoint of the processor's services:

- Receiver organisation ID
- Receiver role
- Receiver service
- Interface version

The name of the operation is used in the CRE to determine the synchronous or asynchronous context and the timeout behaviour.

In the asynchronous context, the way back from the processor to the initiator works in the same way as described above. If the recipient's system or service is not available, the CRE is able to store the message temporarily and forward it later, when the recipient is available again.

In the synchronous context, the response (or business exception) does not have to be routed, since the open connection is directly used for the response. Thus, no header is needed.

The endpoint URLs of the participating systems can be configured in the CRE via the [Registrar](#) unit of the [Scheme Manager](#) system.



2 System Components and Interfaces

This chapter shows the system components, their interfaces and the realisation and usage of these interfaces.

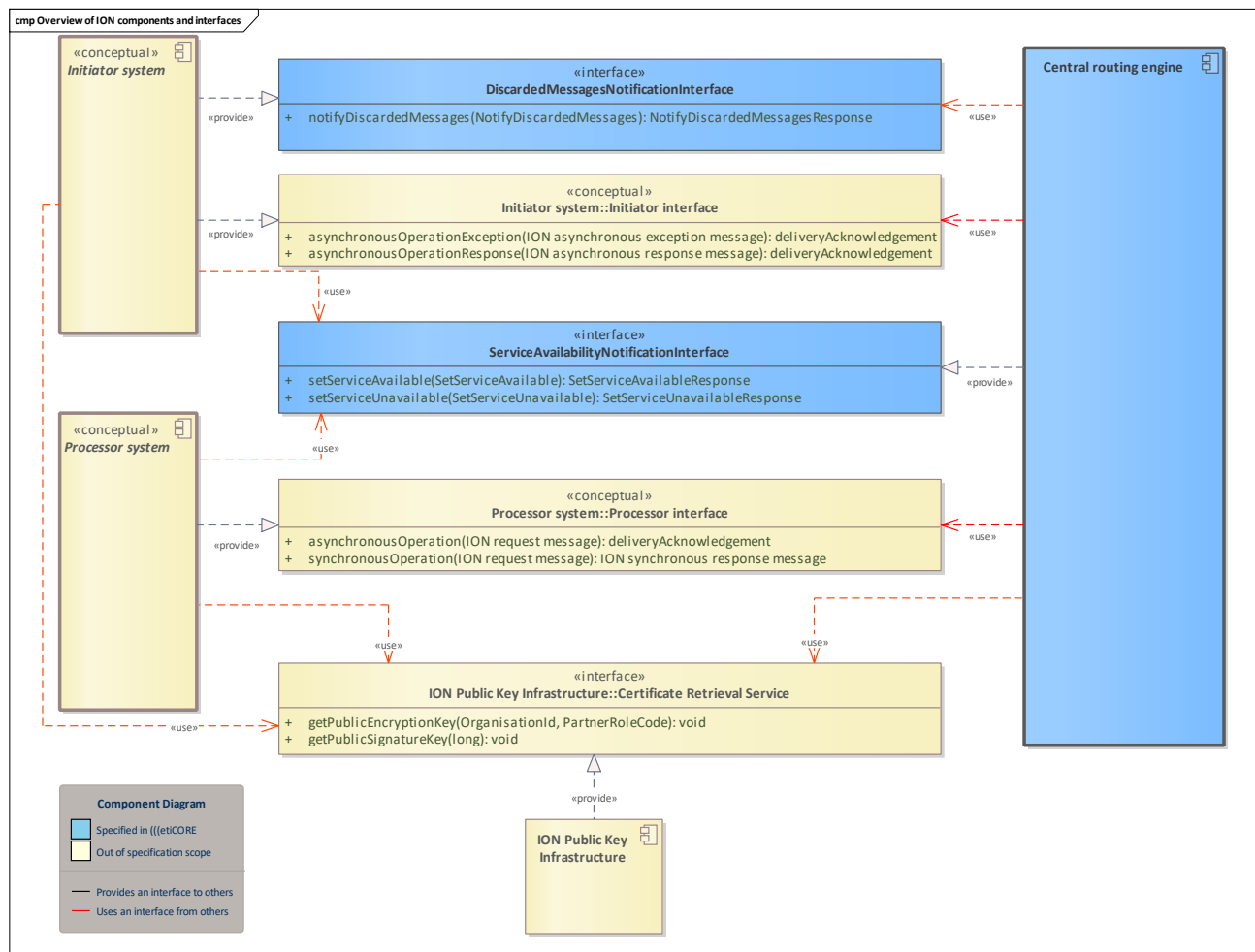


Figure 1: Overview of ION components and interfaces

This diagram shows the communication between the [Central routing engine](#), [Initiator system](#), [Processor system](#) and the [ION Public Key Infrastructure](#) via interfaces.

2.1 ION Public Key Infrastructure

PKI which stores the certificates and public keys for ION message exchange purposes:

- public encryption keys
- public signature keys
- TLS signature keys

2.1.1 Certificate Retrieval Service

Explanatory Interface that allows access to the public key of the [OrganisationalUnit](#) as [Initiator](#) for encryption and the signing key as [Processor](#) to verify the signature of messages and the transport layer.



2.2 Central routing engine

The component which implements the [Central routing engine](#). Several interfaces are implemented and functionality provided:

- Functionality for store & forward
- Functionality for routing
- Interface for service availability notifications: [ServiceAvailabilityNotificationInterface](#)
- Conceptual interface as proxy for synchronous and asynchronous operations of [Processor](#) and [Initiator](#) systems

2.2.1 Central routing engine conceptual interface

Conceptual interface of the CRE only needed for explanatory purposes in sequence diagrams.

2.3 Initiator system

A component that implements an imaginary system of an [Initiator](#).

2.3.1 Initiator interface

The imaginary interface of an [Initiator system](#).

The provided operations can be considered as a placeholder for all real operations in synchronous and asynchronous contexts.

2.4 Processor system

A component that implements an imaginary system of a [Processor](#).

2.4.1 Processor interface

The imaginary interface of a [Processor system](#).

The provided operations can be considered as a placeholder for all real operations in synchronous and asynchronous contexts.

3 Service availability notification

Functionality bundle that contains the use cases and activities to handle service (un)availability notifications from [Initiators](#).

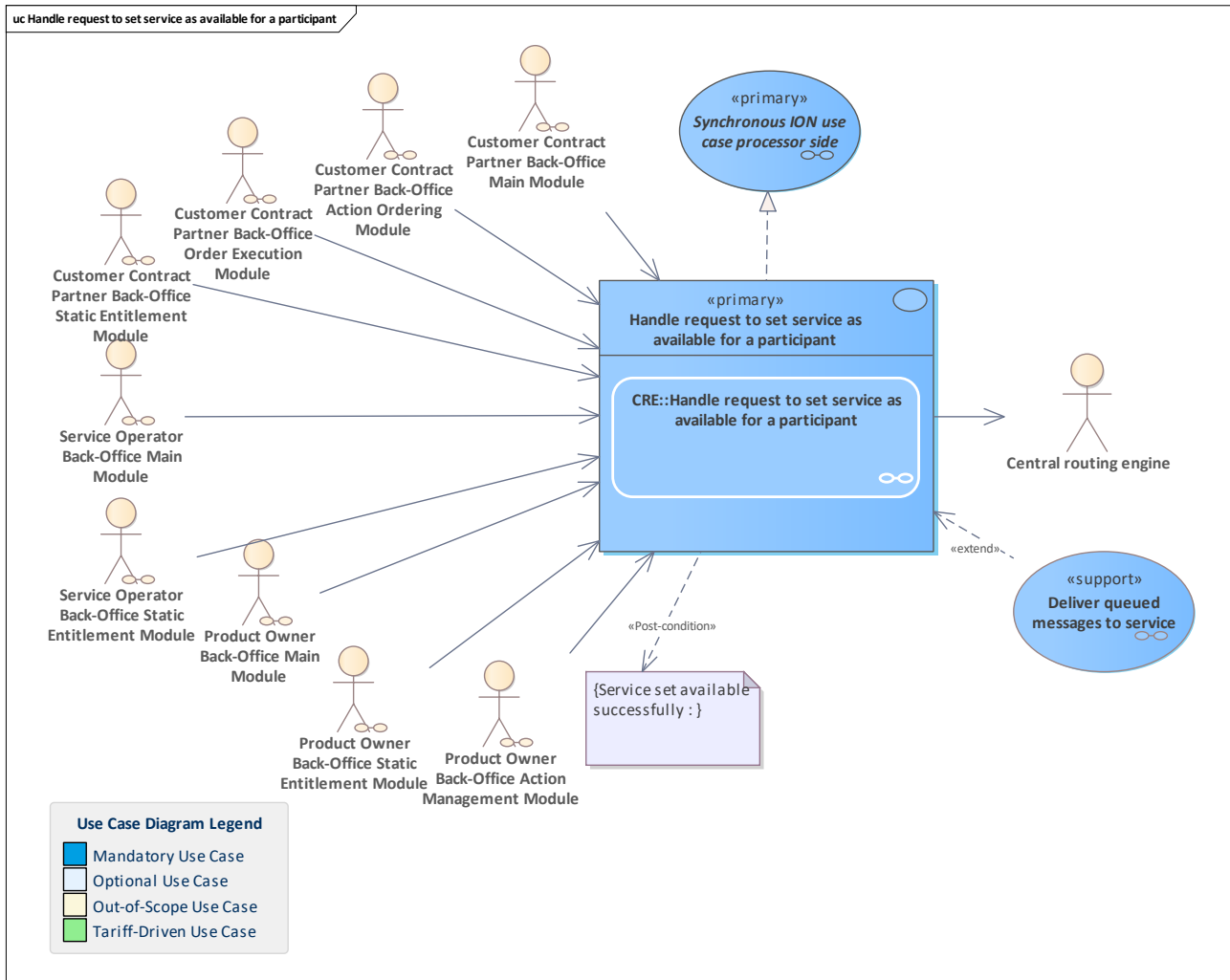
3.1 Overview

[Handle request to set service as available for a participant](#)

[Handle request to set service as unavailable for a participant](#)

3.2 Use Cases

3.2.1 Handle request to set service as available for a participant

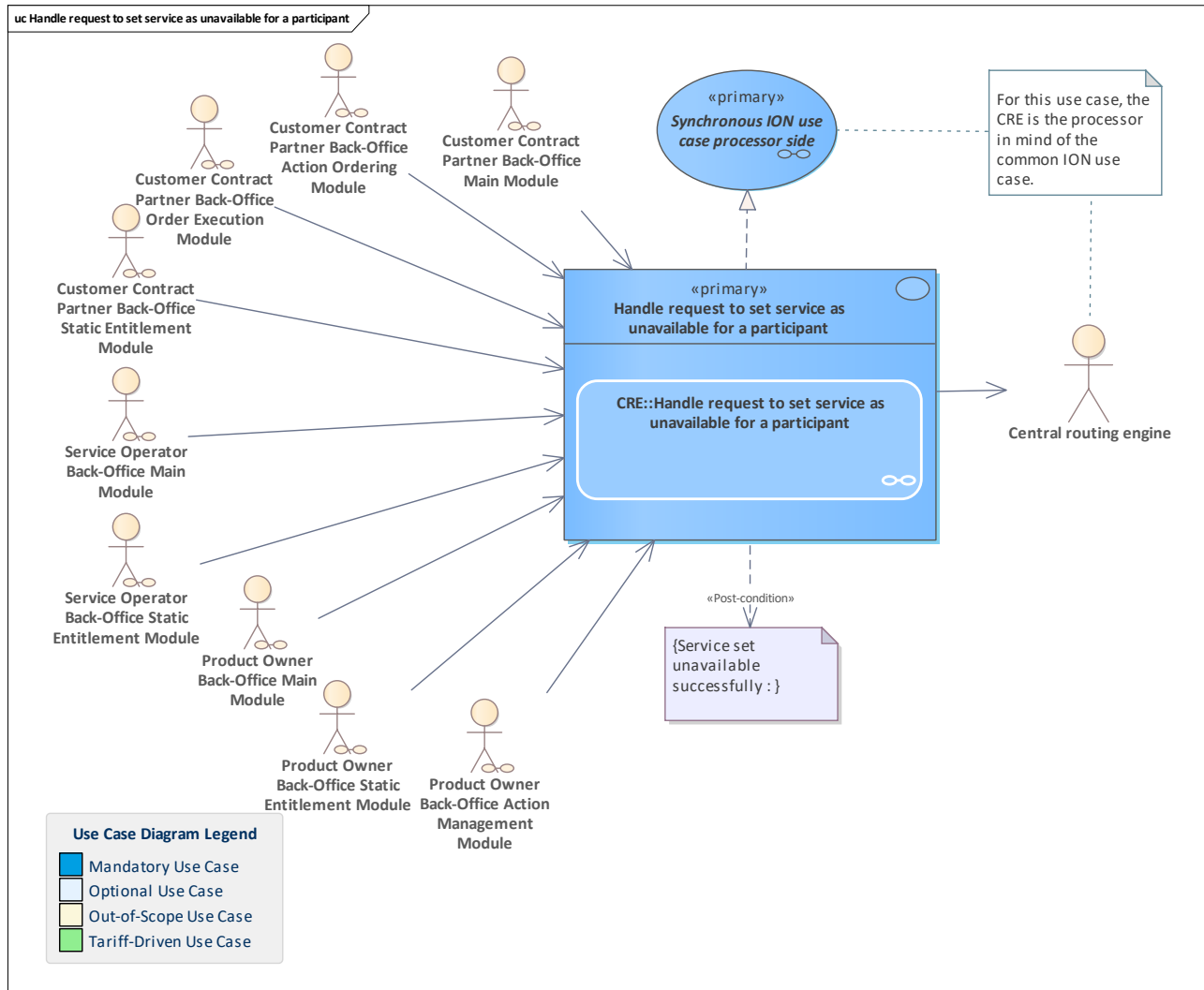


Use Case	Handle request to set service as available for a participant
Description	<p>Use case the on CRE side for a participant to set its service to available for receiving asynchronous messages. If the participant provides more than one service, it has to call the use case operation for each of its provided service.</p> <p>With this use case, the participant indicates to the CRE that this service is available at its configured URL and can be used by all other participants.</p> <p>If any asynchronous messages have been stored temporarily during the offline period for this participant and service, this use case will trigger the use case Deliver queued messages to service. Announcing own services being available is necessary in the asynchronous context to indicate that the system is ready to receive messages on its configured URL (for configuration, see [4] Documentation Registrar System for Service Configuration).</p> <p>Note: Synchronous messages cannot be queued temporarily. Therefore, in this context, notifying the availability of a service that</p>



	works exclusively with synchronous messages does not make sense. Note: this use case works with WSS like all further use cases.
Initiating Actor	Initiator Processor Customer Contract Partner Back-Office Main Module Customer Contract Partner Back-Office Action Ordering Module Customer Contract Partner Back-Office Order Execution Module Product Owner Back-Office Main Module Product Owner Back-Office Action Management Module Service Operator Back-Office Main Module Product Owner Back-Office Static Entitlement Module Service Operator Back-Office Static Entitlement Module Customer Contract Partner Back-Office Static Entitlement Module
Reacting Actor	Central routing engine
Preconditions	
Postconditions	Service set available successfully
Linked Use Cases (Extended By)	Deliver queued messages to service
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Synchronous ION use case processor side
Base Activity	
Inputs	service information : setServiceAvailable
Outputs	set service available response : setServiceAvailableResponse
Error Cases	E_CRE_SERVICE_NOT_CONFIGURED set service available exception : setServiceAvailableException
Activity Diagram	CRE::Handle request to set service as available for a participant

3.2.2 Handle request to set service as unavailable for a participant



Use Case	Handle request to set service as unavailable for a participant
Description	<p>Use case on CRE side for a participant to announce its service as unavailable for receiving asynchronous messages. If the participant provides more than one service, it has to call the use case operation for each of its provided services.</p> <p>After the state transition of this service, the participant indicates that this service is no longer available at its configured URL to all other participants . It is very important to indicate the unavailability of the system or service. It prevents the CRE from sending messages frequently and produces error messages, normally E IONC RECEIVER NOT REACHABLE.</p> <p>If asynchronous messages are sent to this participant's service, they will be stored temporarily by the CRE until the service is announced to be available again by Handle request to set service available for participant.</p> <p>Note: Synchronous messages cannot be queued temporarily. Therefore, in this context, notifying the unavailability of a service</p>



	that works exclusively with synchronous messages does not make sense and will not stop the routing of messages. Note: this use case works with WSS like all further use cases.
Initiating Actor	Processor Initiator Customer Contract Partner Back-Office Main Module Customer Contract Partner Back-Office Action Ordering Module Customer Contract Partner Back-Office Order Execution Module Customer Contract Partner Back-Office Static Entitlement Module Service Operator Back-Office Main Module Service Operator Back-Office Static Entitlement Module Product Owner Back-Office Main Module Product Owner Back-Office Static Entitlement Module Product Owner Back-Office Action Management Module
Reacting Actor	Central routing engine
Preconditions	
Postconditions	Service set unavailable successfully
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Synchronous ION use case processor side
Base Activity	
Inputs	service information : setServiceUnavailable
Outputs	set service unavailable response : setServiceUnavailableResponse
Error Cases	E_CRE_SERVICE_NOT_CONFIGURED set service unavailable exception : setServiceUnavailableException
Activity Diagram	CRE::Handle request to set service as unavailable for a participant

4 Conceptual Routing

Functionality bundle that contains use cases which describe the routing of messages for synchronous and asynchronous context.

For explanation purposes, the use cases and diagrams show the exchange of fictional messages (as a placeholder for all real messages).

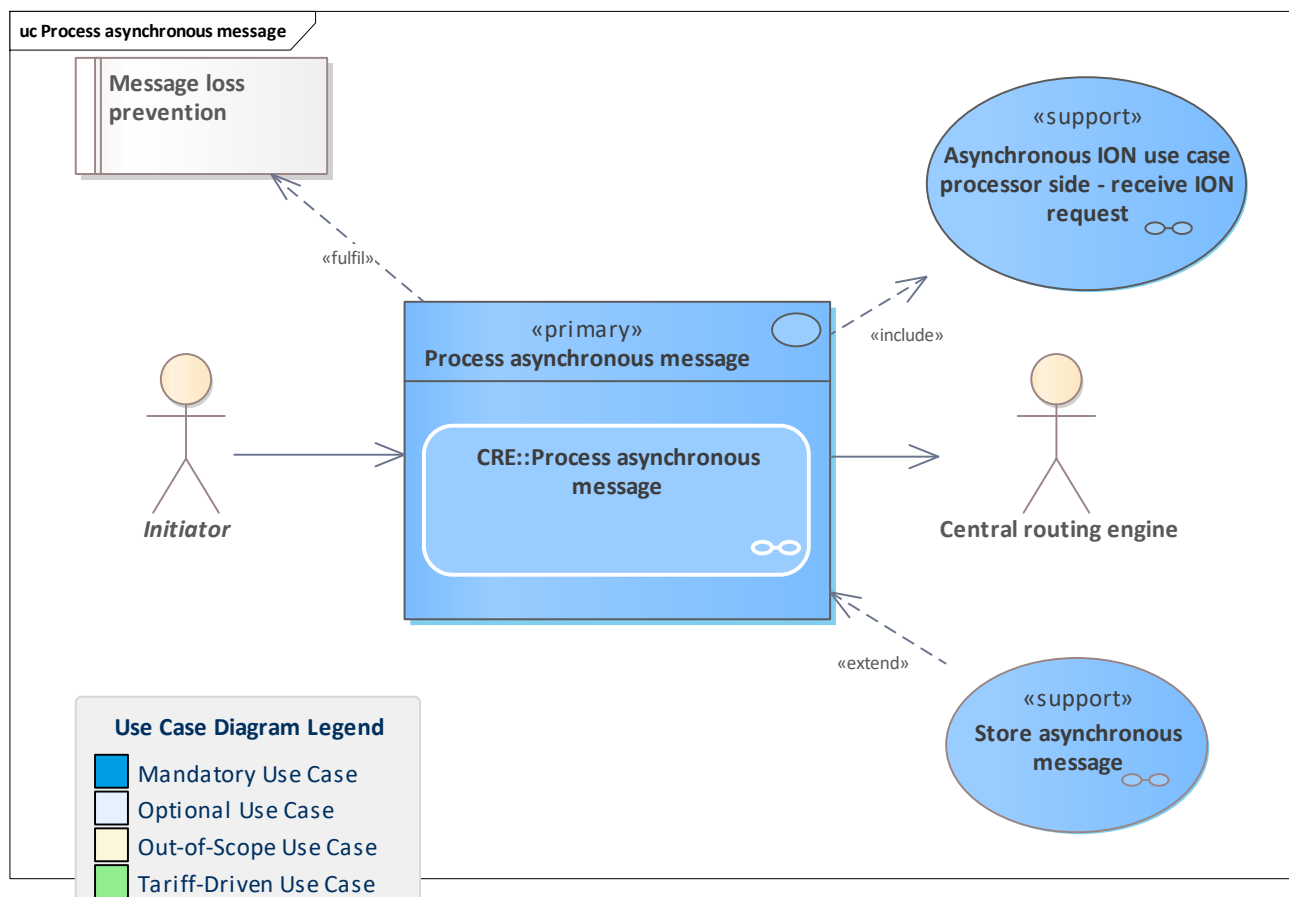
4.1 Overview

[Process asynchronous message](#)

[Process synchronous message](#)

4.2 Use Cases

4.2.1 Process asynchronous message

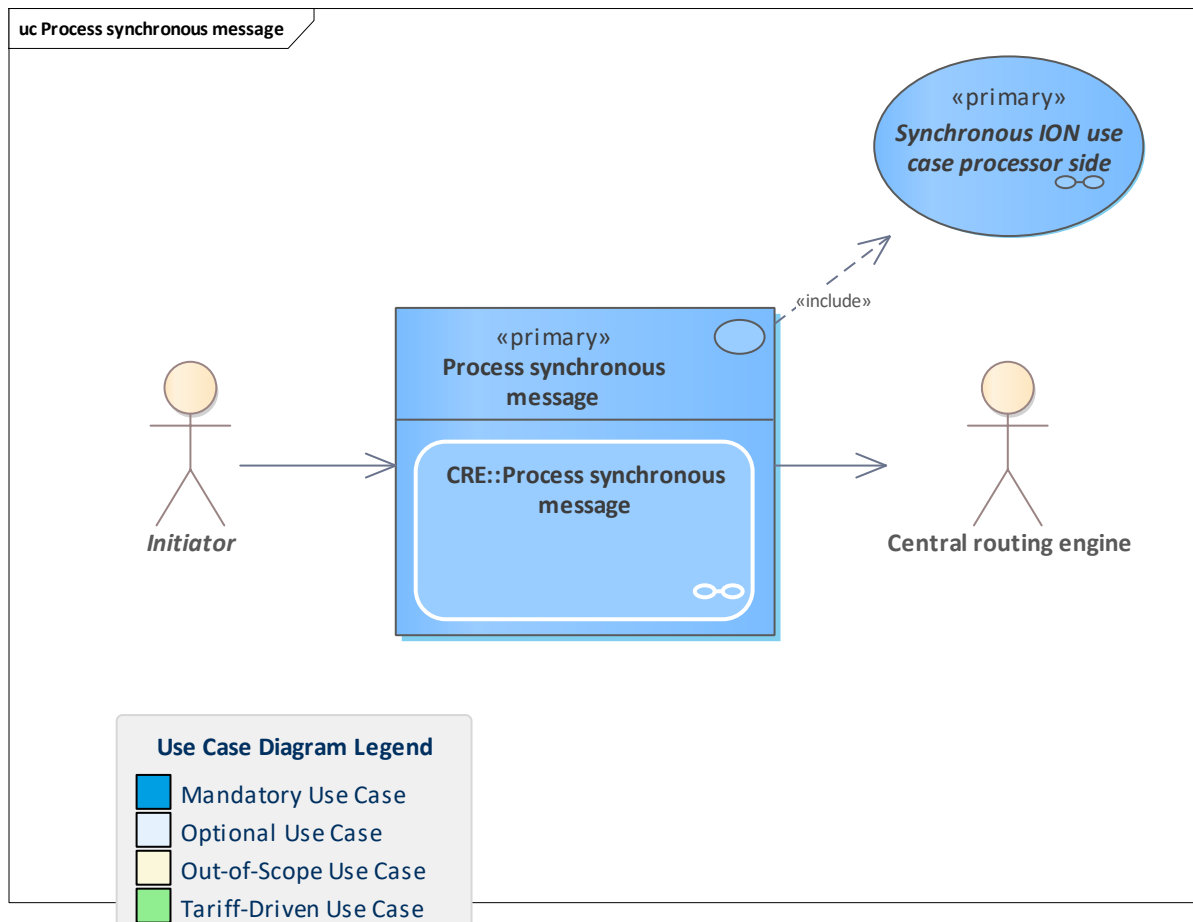


Use Case	Process asynchronous message
Description	Process message in an asynchronous context. The internal configuration in the CRE of the message parameters indicates that a message is sent in an asynchronous context. For the CRE, there is no difference if the message is sent from the Initiator to the Processor or the asynchronous response is sent from the processor



	<p>back to the initiator. From a technical viewpoint, there is no difference for the routing. The CRE is stateless and does not correlate the first message with the second message.</p> <p>The only important reason to know if the context for the message is asynchronous is to perform message caching if the target system is not available.</p> <p>In this case, Store asynchronous ION message is performed.</p>
Initiating Actor	Initiator
Reacting Actor	Central routing engine
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Store asynchronous message
Linked Use Cases (Includes)	Asynchronous ION use case processor side - receive ION request
Linked Use Cases (Realises)	
Base Activity	
Inputs	delivery acknowledgement : deliveryAcknowledgement incoming ION message : ION message with WSS
Outputs	
Error Cases	delivery rejection : deliveryRejection
Activity Diagram	CRE::Process asynchronous message

4.2.2 Process synchronous message



Use Case	Process synchronous message
Description	<p>Process a message in a synchronous context.</p> <p>The CRE forwards the message directly from the Initiator to the Processor.</p> <p>Message caching is not possible. If the processor is not available, an exception is sent to the initiator.</p> <p>The CRE keeps the communication channel open until the message exchange is completed or the configured timeout occurred.</p> <p>The response of the processor is returned directly via the open communication channel.</p>
Initiating Actor	Initiator
Reacting Actor	Central routing engine
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Synchronous ION use case processor side
Linked Use Cases (Realises)	
Base Activity	
Inputs	<p>Incoming ION message : ION request message</p> <p>receiver's response : ION synchronous response message</p>



Outputs	
Error Cases	<u>E ION RECEIVER ROLE MISMATCH</u> <u>E ION RECEIVER SERVICE MISMATCH</u> <u>E ION RECEIVER ORGANISATION MISMATCH</u> <u>E ION UNKNOWN SENDER</u> <u>E ION WRONG SENDER ROLE</u> <u>E ION WRONG SENDER SERVICE</u> <u>E ION SAME MESSAGE IN PROGRESS</u> <u>E ION DUPLICATE ION MESSAGE ID</u> <u>E ION DUPLICATE ION MESSAGE ID</u> <u>E ION DUPLICATE ION MESSAGE ID</u> <u>receiver's exception : ION synchronous exception message</u> <u>soap fault : SOAP fault</u>
Activity Diagram	<u>CRE::Process synchronous message</u>

5 Store & Forward

Functionality bundle that contains use cases which are relevant for queuing messages and implementing a store & forward functionality in the CRE.

5.1 Overview

[Deliver queued messages in scheduler](#)

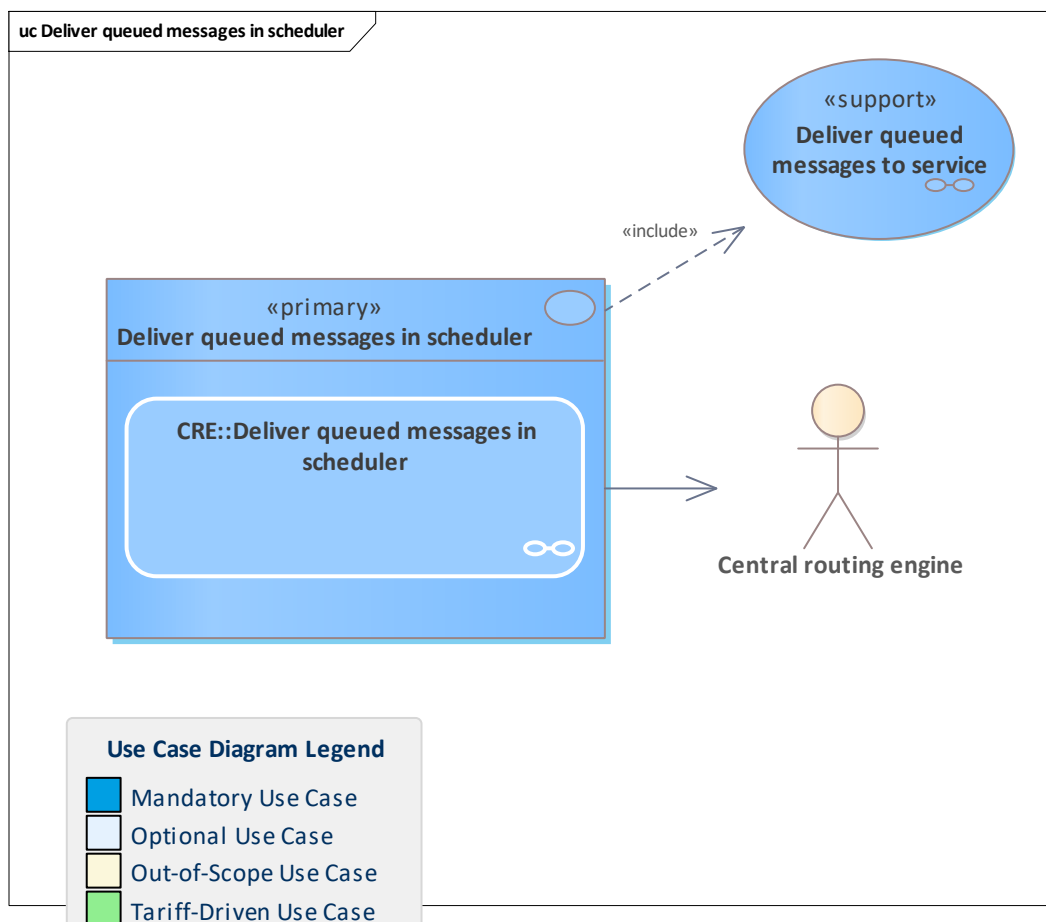
[Deliver queued messages to service](#)

[Store asynchronous message](#)

[Discard queued messages](#)

5.2 Use Cases

5.2.1 Deliver queued messages in scheduler

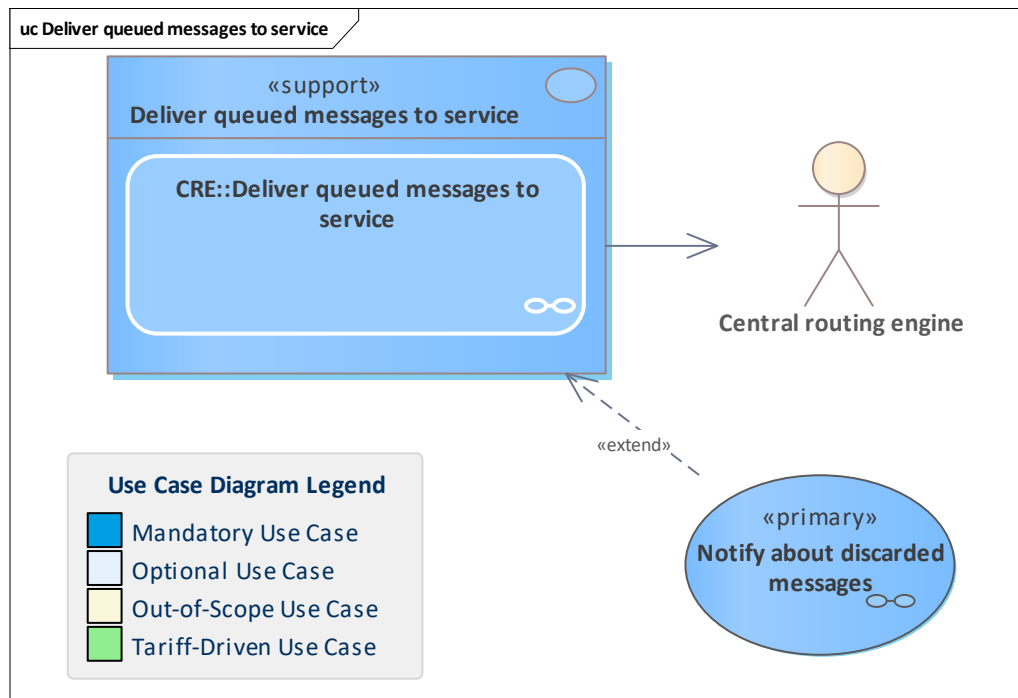


Use Case	Deliver queued messages in scheduler
Description	Use case frequently started in a scheduler (see [3] CRE Specification: Message Store and Forward) which delivers one or more messages which were previously stored in the queue. For any reason, a participant and its services can be announced as



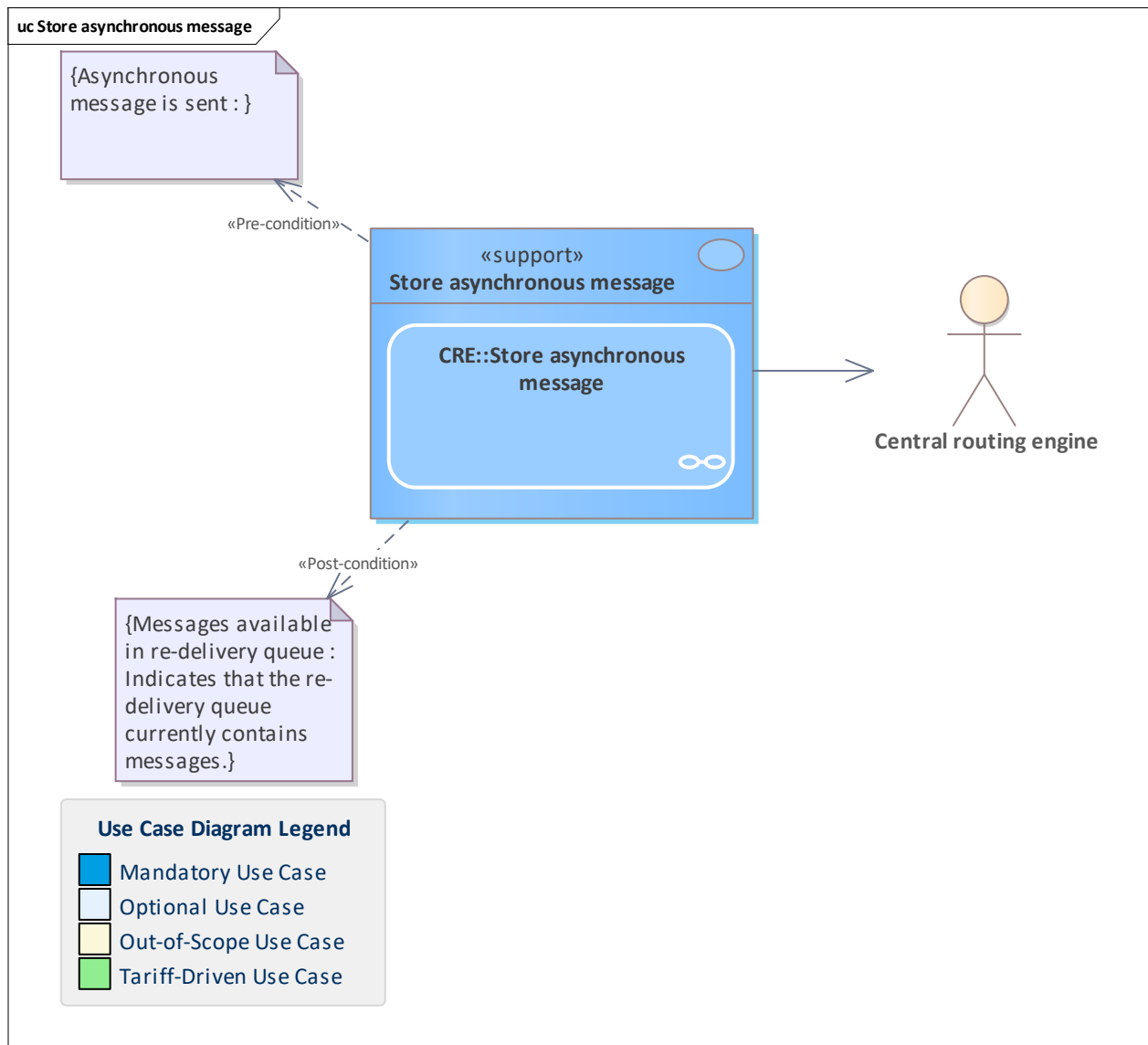
	<p>available even in cases where communication problems might have occurred, resulting in the temporary storage of messages. Thus, a scheduled process is needed to determine, if any messages have to be delivered for each service announced as available by each active participant.</p> <p>Note: by considering only active participants, messages of deactivated senders and receivers remain in the queue until there have been activated again or the messages expire and are moved to the dead letter queue.</p>
Initiating Actor	
Reacting Actor	Central routing engine
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Deliver queued messages to service
Linked Use Cases (Realises)	
Base Activity	
Inputs	Interface version
Outputs	
Error Cases	
Activity Diagram	CRE::Deliver queued messages in scheduler

5.2.2 Deliver queued messages to service



Use Case	Deliver queued messages to service
Description	Supporting use case that delivers queued ION messages to the endpoint URL of a defined service configuration. Collects also potential discarded messages information about expired messages or messages that ultimately cannot be delivered to the intended receiver.
Initiating Actor	
Reacting Actor	Central routing engine
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Notify about discarded messages
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Service configuration : Service configuration
Outputs	
Error Cases	
Activity Diagram	CRE::Deliver queued messages to service

5.2.3 Store asynchronous message

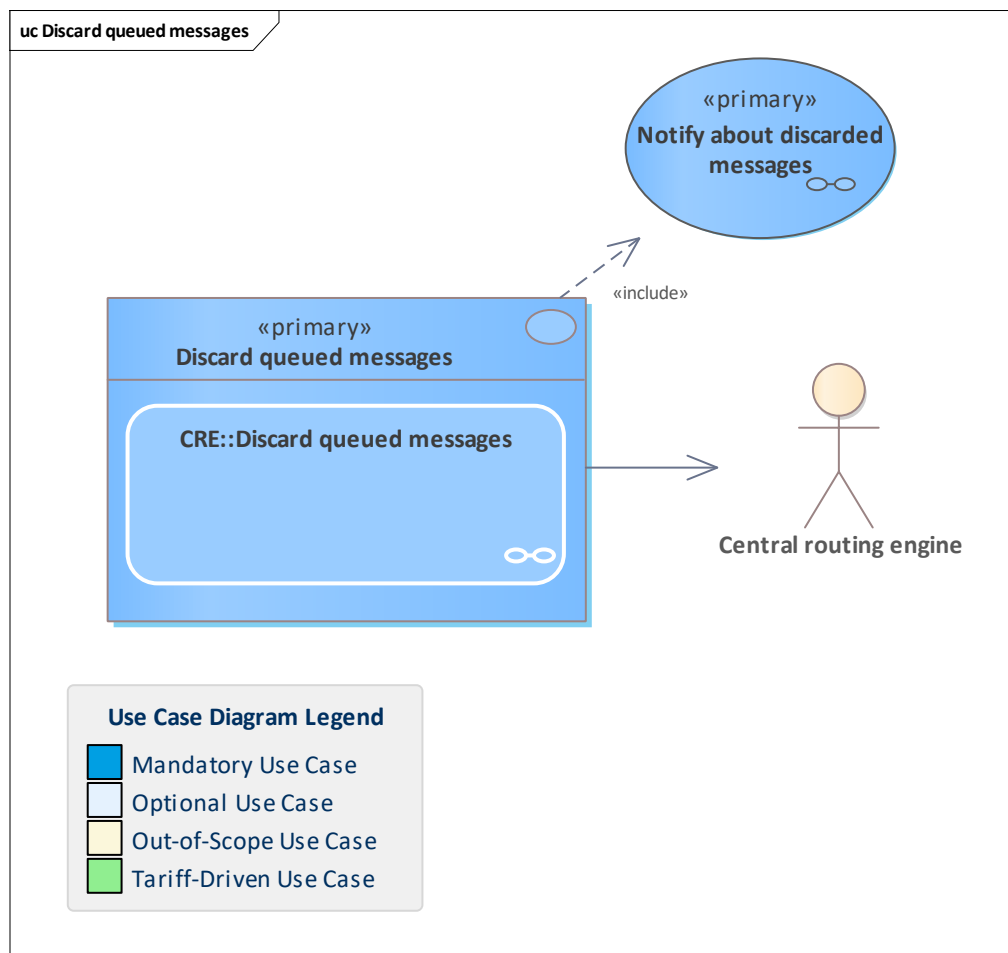


Use Case	<u>Store asynchronous message</u>
Description	<p>Supporting use case to store a message temporarily in an asynchronous context. Queuing is only possible if the message is configured as asynchronous in the CRE. In this case, the CRE may store the message in certain scenarios:</p> <ul style="list-style-type: none"> The receiving system's service is not set to available The receiving system's service is announced as available but the response HTTP code is 503 (temporary not available) <p>In these cases, the message is put into a queue for later delivery.</p>
Initiating Actor	
Reacting Actor	<u>Central routing engine</u>
Preconditions	<u>Asynchronous message is sent</u>
Postconditions	<u>Messages available in re-delivery queue</u>



Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>ION message in asynchronous context : Temporarily storable ION message</u>
Outputs	
Error Cases	<u>E IONC CRE COULD NOT STORE MESSAGE</u> <u>E IONC CRE COULD NOT STORE MESSAGE</u> <u>E IONC CRE COULD NOT STORE MESSAGE:</u> <u>E IONC CRE COULD NOT STORE MESSAGE</u>
Activity Diagram	<u>CRE::Store asynchronous message</u>

5.2.4 Discard queued messages



Use Case	Discard queued messages
Description	Scheduled process which frequently looks up for expired messages in the re-delivery queue. The final attempt of (configured) delivery iterations failed. Messages become expired due to the WSS expiration timestamp defined in [5] SLAs for Message Transfer . The messages are moved to the dead letter queue and the meta-information of the messages (ionRoutingHeader elements) is used to build a notifyDiscardedMessages element which is sent to the original sender.
Initiating Actor	
Reacting Actor	Central routing engine
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Notify about discarded messages
Linked Use Cases (Realises)	
Base Activity	



Inputs	scheduled datetime : ZonedDateTime
Outputs	
Error Cases	
Activity Diagram	CRE::Discard queued messages

6 Monitoring and notification

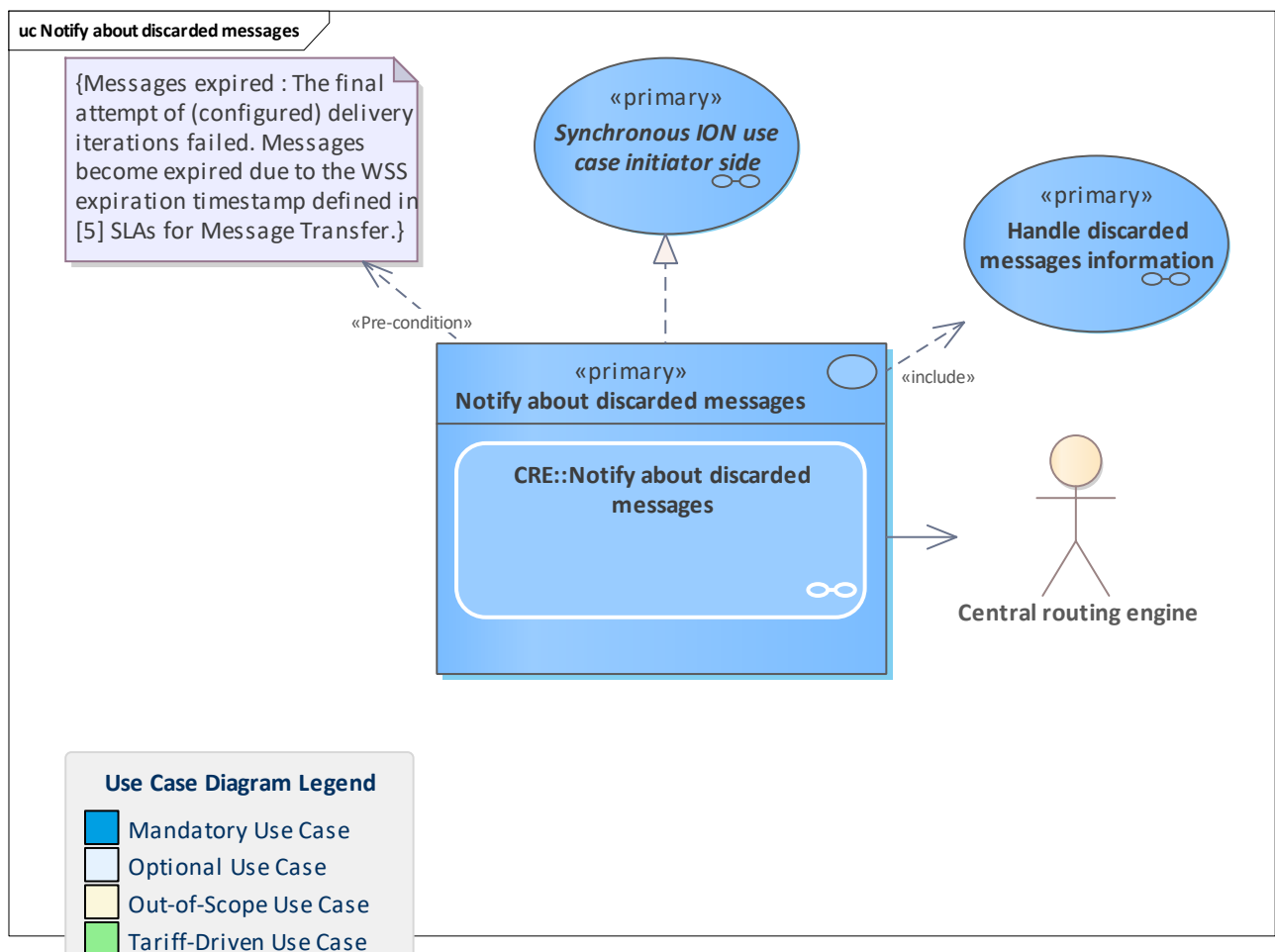
Functionality bundle that contains use cases for monitoring in the CRE especially in the store & forward context. Describes the [Initiator](#) side (CRE part).

6.1 Overview

[Notify about discarded messages](#)

6.2 Use Cases

6.2.1 Notify about discarded messages



Use Case	Notify about discarded messages
Description	Messages in the Redelivery queue can be discarded for a specific sender service (organisation ID, role ID and service name) when the use cases Deliver queued messages to service (triggered by scheduler or when setting a service to available status) or Discard queued messages are executed. Accordingly, this use case is called in these use cases.



	Group these messages for the sender service by recipient, recipient role, service and message type (operation name) as well as the number of messages in this group. Then send this information in the form of DiscardedMessages back to the sender which had previously sent these messages to the intended, but not reached, recipient. Distinguish between expired messages and undeliverable messages.
Initiating Actor	
Reacting Actor	Central routing engine
Preconditions	Messages expired
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle discarded messages information
Linked Use Cases (Realises)	Synchronous ION use case initiator side
Base Activity	
Inputs	routing headers of undeliverable messages : ionRoutingHeader routing headers of expired messages : ionRoutingHeader
Outputs	
Error Cases	
Activity Diagram	CRE::Notify about discarded messages

