

# SPEC-SO-RS

Last changed:	20.11.2025 11:03:59
Version:	3.0.0-rc.6
Creator:	VDV ETS

## Table of Contents

1	Introduction	6
2	Service Operator System	6
2.1	Service Operator System	7
2.2	Service Operator Back-Office Main Module	8
2.3	Service Operator Back-Office Static Entitlement Module	8
3	Notification Process Patterns	8
3.1	Supporting activities	11
3.1.1	Perform transaction to XY	11
3.1.2	Prepare XY parameters	14
3.2	Supporting classes	15
3.2.1	Action parameters	15
3.2.2	XY parameters	15
3.2.3	SignedXYAttestation	15
3.2.4	Notification parameters	15
3.2.5	XYNotification	16
3.2.6	ForwardedXYNotification	16
3.2.7	tNotifyXY	16
3.2.8	tNotifyXYAborted	16
3.2.9	notifyXY	16
3.2.10	notifyXYResponse	16
3.2.11	notifyXYException	16
3.2.12	forwardXYNotification	16
3.2.13	forwardXYNotificationResponse	16
3.2.14	forwardXYNotificationException	16
3.3	Perform entitlement XY and notify	16
3.3.1	Perform entitlement XY and notify	17
3.3.2	T-Module::Perform entitlement XY and notify	18
3.3.3	Notify entitlement XY	19
3.3.4	Notify entitlement XY aborted	20
3.3.5	Notify entitlement XY aborted based on attestation	20
3.3.6	Notify XY (entitlement owned)	21
3.3.7	Notify XY (entitlement non-owned)	22
3.4	Perform application XY and notify	22
3.4.1	Perform application XY and notify	23
3.4.2	T-Module::Perform application XY and notify	23
3.4.3	Notify application XY	25
3.4.4	Notify application XY aborted	25
3.4.5	Notify XY (application owned)	26

3.4.6	Notify XY (application non-owned)	27
3.5	Handle entitlement XY notification from operational perspective	27
3.5.1	Handle entitlement XY notification from operational perspective	28
3.5.2	Role-BO-Module::Handle entitlement XY notification from operational perspective	28
3.5.3	Check and process entitlement XY notification from operational perspective	29
3.5.4	Role-BO-Module::Handle entitlement XY aborted notification from operational perspective	30
3.6	Handle application XY notification from operational perspective	31
3.6.1	Handle application XY notification from operational perspective	32
3.6.2	Role-BO-Module::Handle application XY notification from operational perspective	32
3.6.3	Check and process application XY notification from operational perspective	33
3.6.4	Role-BO-Module::Handle application XY aborted notification from operation perspective	35
3.7	Handle entitlement XY notification from product perspective	35
3.7.1	Handle entitlement XY notification from product perspective	36
3.7.2	PO-BO-Module::Handle entitlement XY notification from product perspective	36
3.7.3	Check and process entitlement XY notification from product perspective	38
3.7.4	Conditionally forward notification to pCCP	38
3.8	Handle entitlement XY notification from contractual perspective	39
3.8.1	Handle entitlement XY notification from contractual perspective	40
3.8.2	pCCP-BO-Module::Handle entitlement XY notification from contractual perspective	40
3.8.3	Check and process entitlement XY notification from contractual perspective	41
3.8.4	pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification	42
3.9	Handle application XY notification from contractual perspective	43
3.9.1	Handle application XY notification from contractual perspective	44
3.9.2	pCCP-BO-Module::Handle application XY notification from contractual perspective	44
3.9.3	Check and process application XY notification from contractual perspective	45
3.9.4	pCCP-BO-Module::Perform specific contractual tasks on application XY notification	46
4	Verifying Lists Updated via Increments	47
4.1	Checksum calculation for hotlist and action list verification	49
4.2	Example calculation for an action list inventory	49
4.3	Example calculation for an application hotlist inventory	50
4.4	Example calculation for an entitlement hotlist inventory	51
5	Basic Bundle Back-Office System	52
5.1	Overview	52
5.2	Use Cases	53
5.2.1	Handle discarded messages information	53
5.2.2	Set service as available for a participant	55
5.2.3	Set service as unavailable for a participant	57
5.2.4	Retrieve CV certificate over signing key	59

5.2.5	Retrieve and distribute the CA certificate repository	61
5.2.6	Retrieve and distribute the CV certificate revocation list	63
5.2.7	Notify events	65
5.2.8	Handle events notification	67
5.2.9	Demand application hotlisting	69
5.2.10	Determine user medium owner	71
5.2.11	Demand entitlement hotlisting	73
5.2.12	Demand SAM hotlisting	75
5.2.13	Determine SAM owner	77
5.2.14	Revoke application hotlisting demand	79
5.2.15	Revoke entitlement hotlisting demand	81
5.2.16	Retrieve entitlement hotlist	83
5.2.17	Optional: Retrieve entitlement hotlist with product information	85
5.2.18	Optional: Retrieve incremental entitlement hotlist	87
5.2.19	Optional: Verify entitlement hotlist updated via increments	89
5.2.20	Update organisation hotlist inventory	91
5.2.21	Update SAM hotlist inventory	93
5.2.22	Retrieve and distribute organisation list	95
6	Basic Bundle Terminal Operator System - Foundation	97
6.1	Overview	97
6.2	Use Cases	97
6.2.1	Handle SAM hotlisting demand	97
6.2.2	Check and add SAM to hotlist	100
6.2.3	Retrieve application hotlist	102
6.2.4	Optional: Retrieve incremental application hotlist	104
6.2.5	Optional: Verify application hotlist updated via increments	106
6.2.6	Update authentication key hotlist inventory	108
6.2.7	Update hotlist inventory from operational perspective	110
6.2.8	Distribute SAM configuration	112
6.2.9	Optional: Distribute the SAM reset script	114
6.2.10	Distribute tariff module	116
6.2.11	Monitor SAMs from operational perspective	118
7	Basic Bundle Terminal Operator System - Extended Logging	120
7.1	Overview	120
7.2	Use Cases	120
7.2.1	Optional: Process extended logging for an entitlement	120
7.2.2	Optional: Process extended logging for an application	122
8	Basic Bundle Terminal Operator System - UM with Application	124
8.1	Overview	124
8.2	Use Cases	124
8.2.1	Handle defective user medium with application	124
8.2.2	Optional: Determine UM app instance ID for Medium ID	126

8.2.3	Handle application blocked notification from operational perspective	128
8.2.4	Handle entitlement blocked notification from operational perspective	130
9	Electronic Ticket Bundle SO-System	132
9.1	Overview	132
9.2	Use Cases	132
9.2.1	Handle entitlement inspected notification from operational perspective	132
10	Account-Based Payment Bundle SO-System	134
11	Stored-Value Payment Bundle SO-System	134
12	Sale Electronic Ticket via Account-Based Payment Bundle SO-System	134
13	Sale Electronic Ticket via Stored-Value Payment Bundle SO-System	134
14	IN-OUT Bundle SO-System	134
14.1	Overview	134
14.2	Use Cases	134
14.2.1	Handle check-in notification from operational perspective	134
14.2.2	Handle check-out notification from operational perspective	137
14.2.3	Handle stored-value payment method recharged notification from operational perspective	139
15	Ordered Action Management Bundle SO-System	141
16	Static Entitlements Bundle SO-System	141
16.1	Overview	141
16.2	Use Cases	141
16.2.1	Handle static entitlement inspected notification from operational perspective	141
17	Miscellaneous Bundle SO-System	144
17.1	Overview	144
17.2	Use Cases	144
17.2.1	Optional: Handle entitlement validated notification from operational perspective	144
17.2.2	Optional: Retrieve valid entitlements for given app instance ID	146

# 1 Introduction

From the EFM perspective, the service provider system processes incoming notifications of entitlements and applications. It performs initial monitoring and forwards messages to the system of the relevant product owner.

This reference specification contains all functionality bundles and use cases for a service provider back-office system.

Depending on the deployment variant, different functionality bundles can be combined.

# 2 Service Operator System

This chapter contains all components and interfaces that are involved with the [Service Operator System](#). Depending on the deployment variant, not all components and interfaces are required.

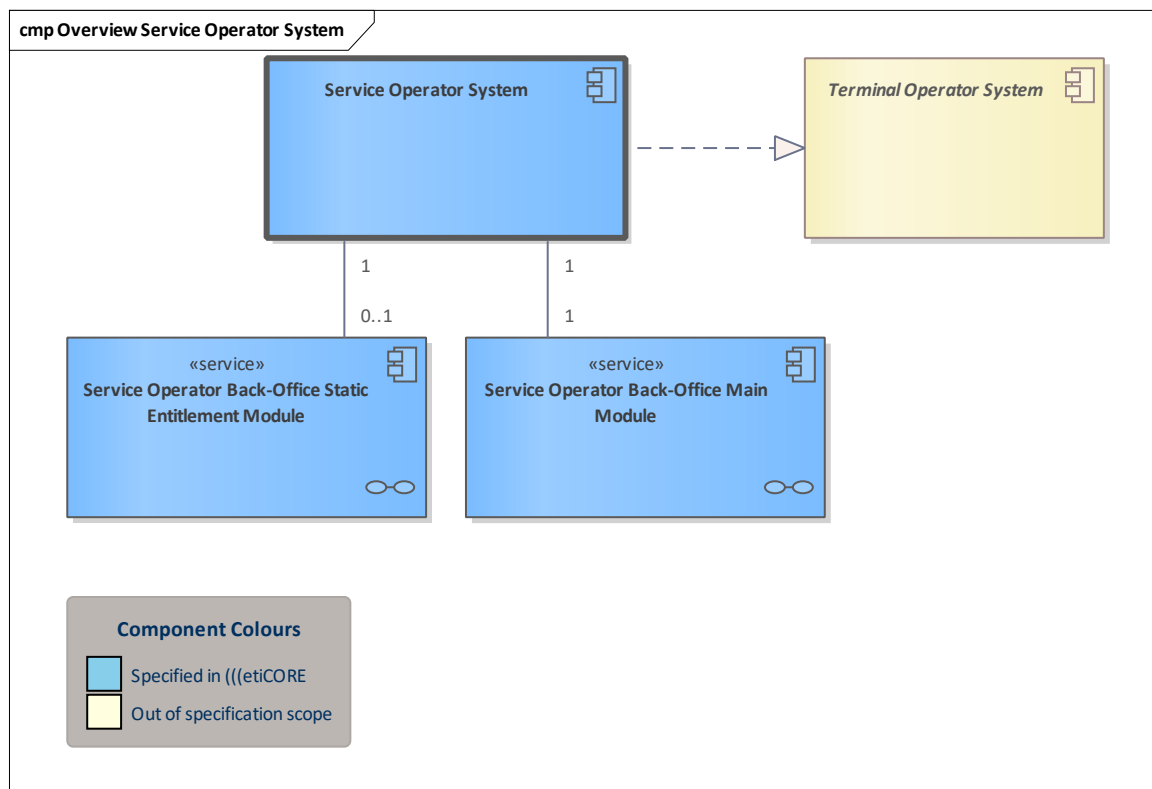


Figure 1: Overview Service Operator System

Shows the composition of a [Service Operator System](#).

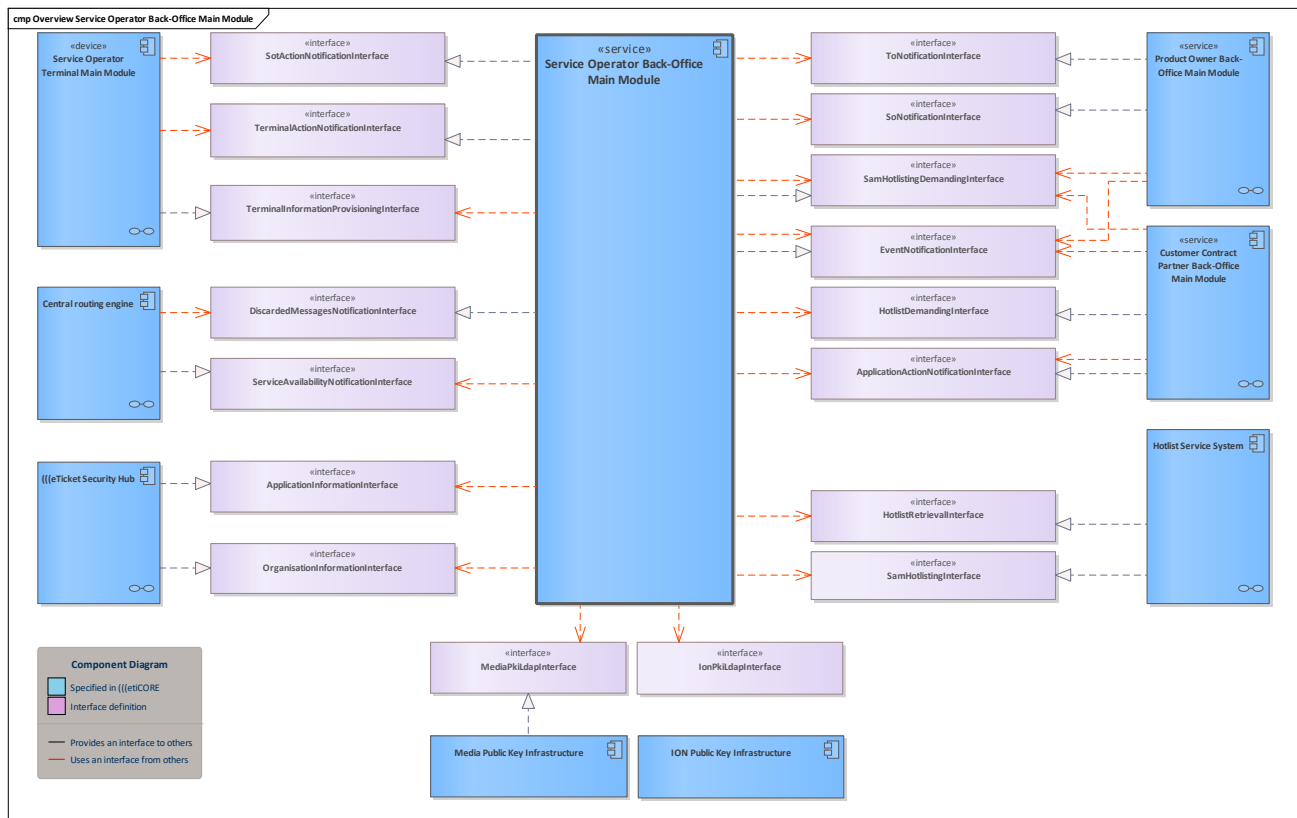


Figure 2: Overview Service Operator Back-Office Main Module

Shows the interaction of a [Service Operator Back-Office Main Module](#) via interfaces with other components.

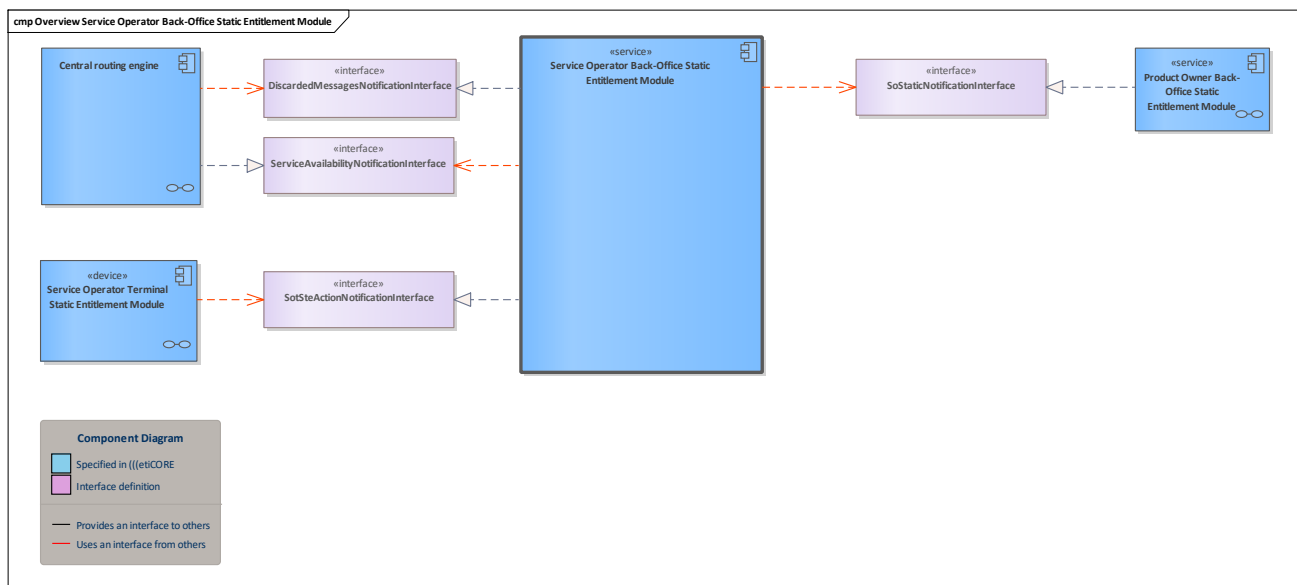


Figure 3: Overview Service Operator Back-Office Static Entitlement Module

Shows the interaction of a [Service Operator Back-Office Static Entitlement Module](#) via interfaces with other components.

## 2.1 Service Operator System

Component for a service operator system. Depending on the implemented functionality bundles, the service operator system consists of



- [Service Operator Back-Office Main Module](#) (always)
- [Service Operator Back-Office Static Entitlement Module](#) (if the SO also works with static entitlements)

## 2.2 Service Operator Back-Office Main Module

Component which implements the basic functionality for a [Service Operator](#) in a [Service Operator System](#).

## 2.3 Service Operator Back-Office Static Entitlement Module

System (component) which implements the necessary functionality for the [Service Operator](#) and [Service Operator System](#) that works with static entitlements.

# 3 Notification Process Patterns

This chapter deals with the patterns which are widely used in the model, especially for notification processes.

If you understand these patterns, you will understand the structure of most of the use cases.

The different actions that can be executed by the user medium fall into different categories with respect to who is authorised to execute them. The important distinction here is whether the executing organisation is also the owner of the target object (UM application for actions targeting the application, entitlement for actions targeting the entitlement).

For some actions, the executing organisation is never the object owner (i.e. for all actions only relevant to Service Operators, which are never owners of UMs or entitlements). Others may only be performed by the owner, i.e. unblocking (outside of ordered action execution). The rest may be performed by organisations that may or may not be the object owner, e.g. debiting a payment method or blocking entitlement or application.

The actions (and their notifications) fall into four different categories :

- Entitlement owned
- Entitlement non-owned
- Application owned
- Application non-owned

Depending on the category the action falls into, the handling of their notifications with respect to potential forwarding to the owner and with respect to when the contractual processing happens may be different.

Templates for the execution of actions and the handling of their notifications for the categories entitlement owned and non-owned, as well as application owned and non-owned are shown.

Within these categories, the notification handling is very similar.



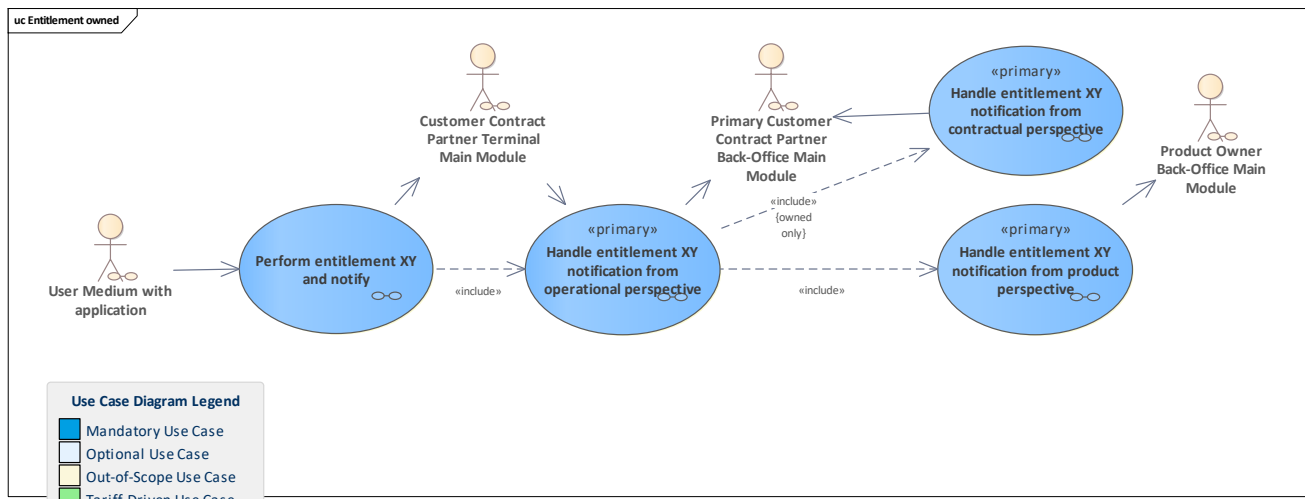


Figure 4: Entitlement owned

An entitlement-specific notification can only be created by the owner of the entitlement. The PO is informed about the action by the party performing the action, which always is the owner in this scenario.

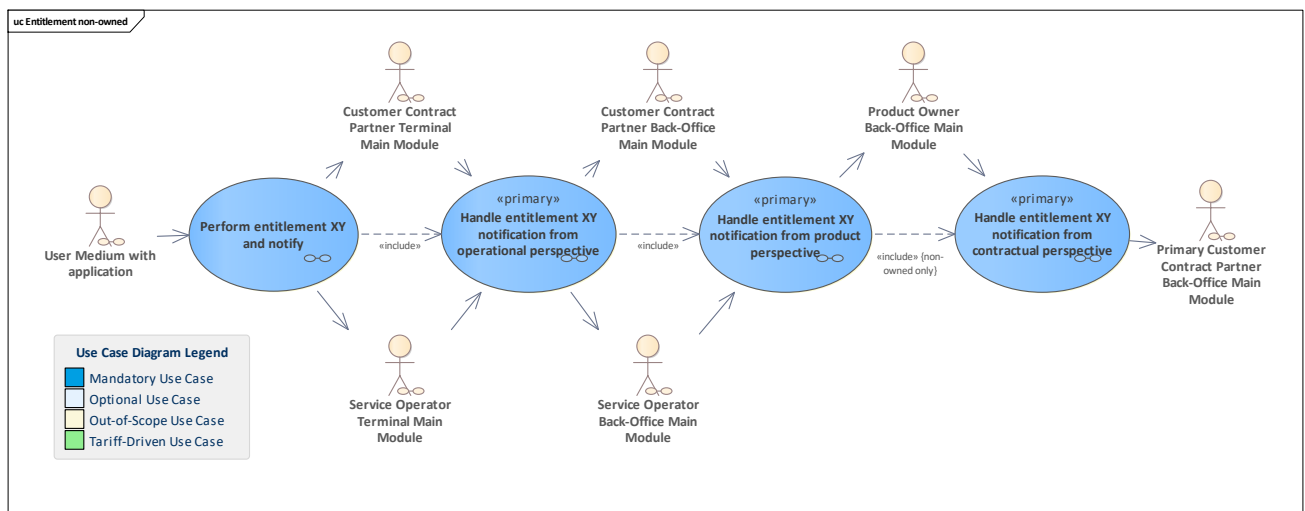


Figure 5: Entitlement non-owned

An entitlement-specific notification can only be created by parties other than the owner of the entitlement. The PO is informed about the action by the party performing the action. The owner is informed about the action by the PO.

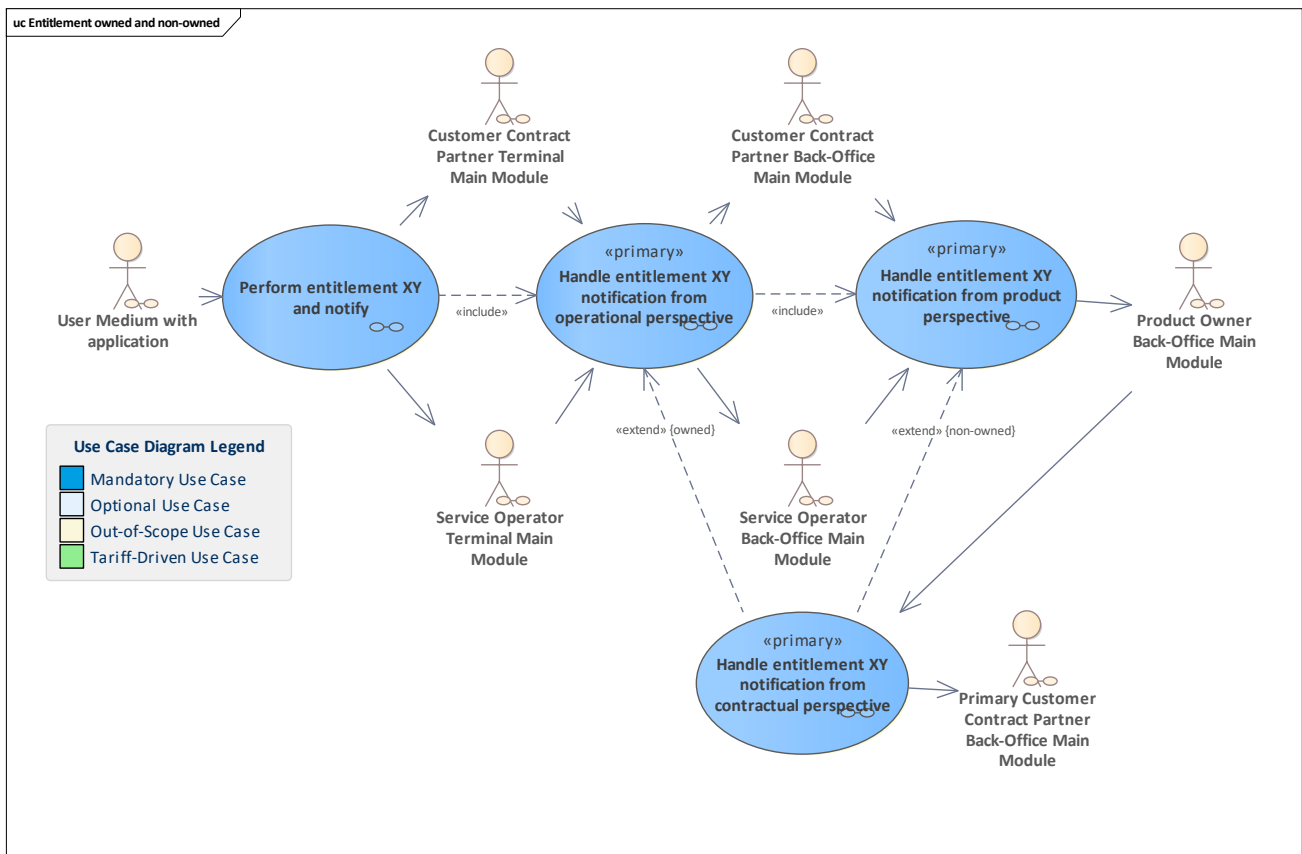


Figure 6: Entitlement owned and non-owned

An entitlement-specific notification can be created by the owner of the entitlement and other parties.

The PO is informed about the action by the party performing the action.

The owner is informed about the action by the PO if he did not perform the action himself.

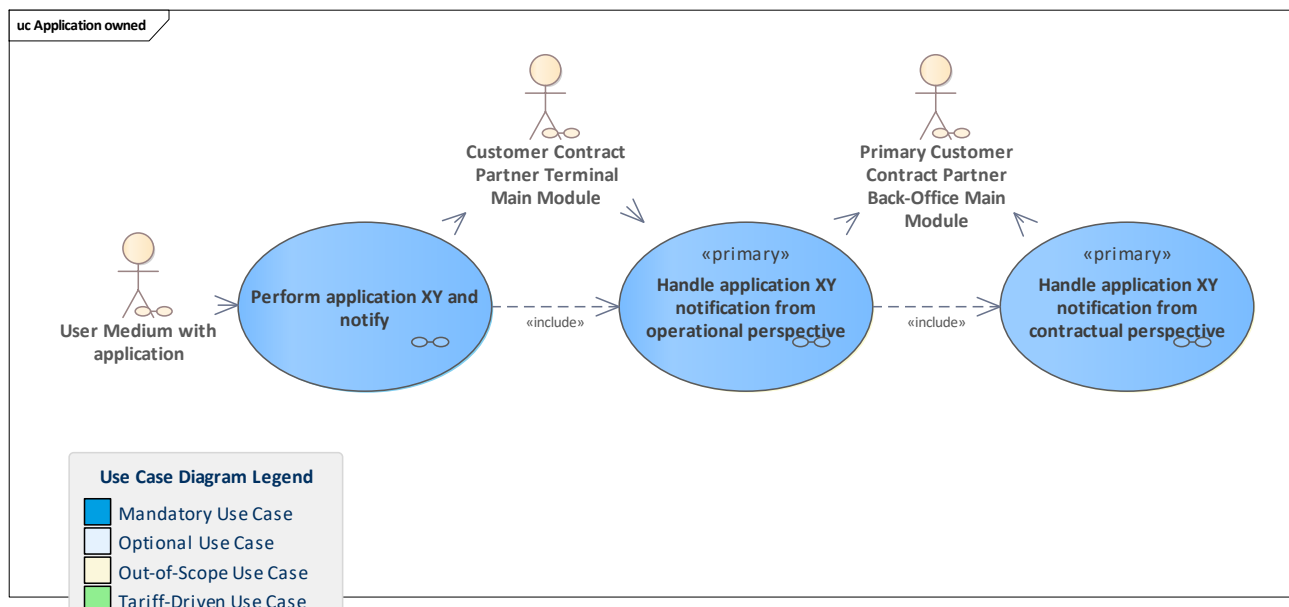


Figure 7: Application owned

An application-specific notification can only be created by the owner of the application.

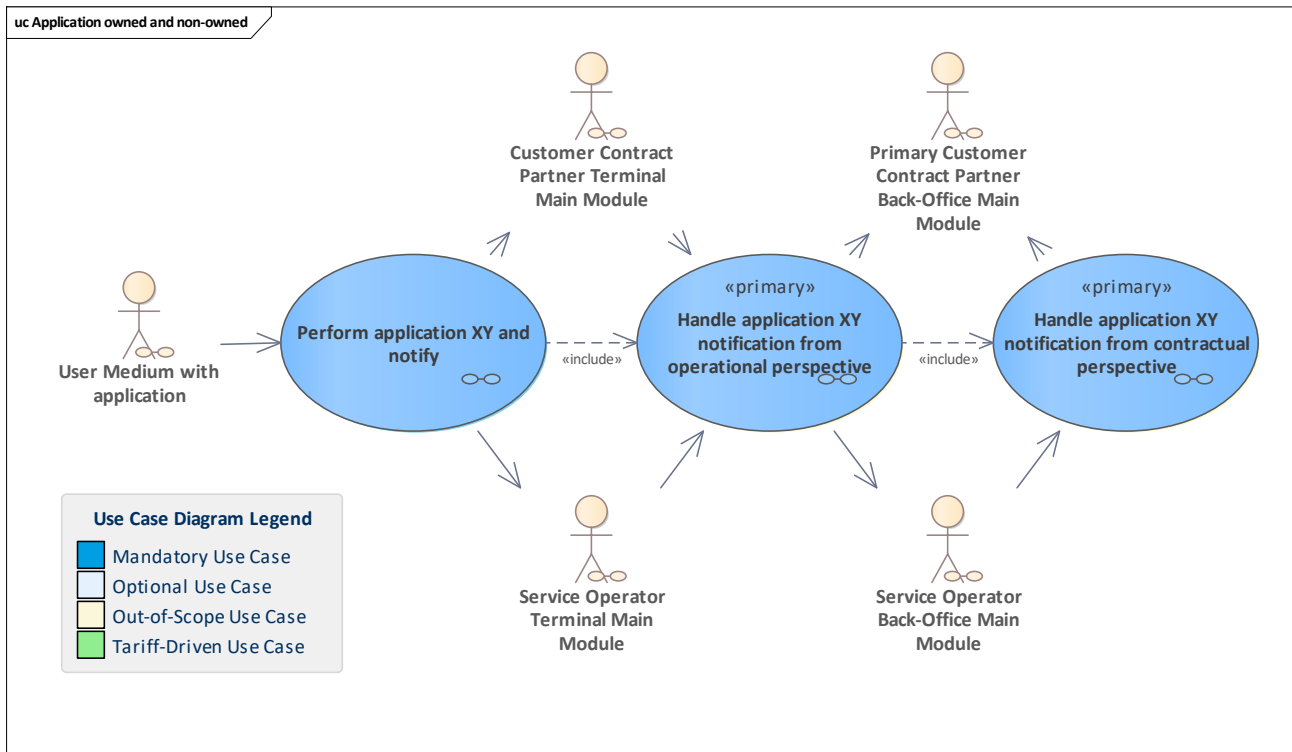


Figure 8: Application owned and non-owned

An application-specific notification can be created by the owner of the application and other parties.

The owner is informed about the action by the party performing the action if he did not perform the action himself.

## 3.1 Supporting activities

This chapter contains activities that are used in more than one use case.

### 3.1.1 Perform transaction to XY

Shows the actual transaction including XY. May start by ensuring, that a session is established. This is left out if there is a business requirement to securely retrieve the entitlement in question before performing the operation since that already had to be done in a session. The next step is to prepare the XY parameters for the action to be performed. Finally, it contains a transaction following the schema described in [Transaction](#).

In some special cases, a transaction may involve more than one action execution. In that case, there will be a variation of this diagram type leaving out the commit transaction step at the end (and the corresponding timeout error situation). This variation is then used in conjunction with a classic version of this diagram type within a single [Role-T-Module::Perform entitlement XY and notify](#) diagram.

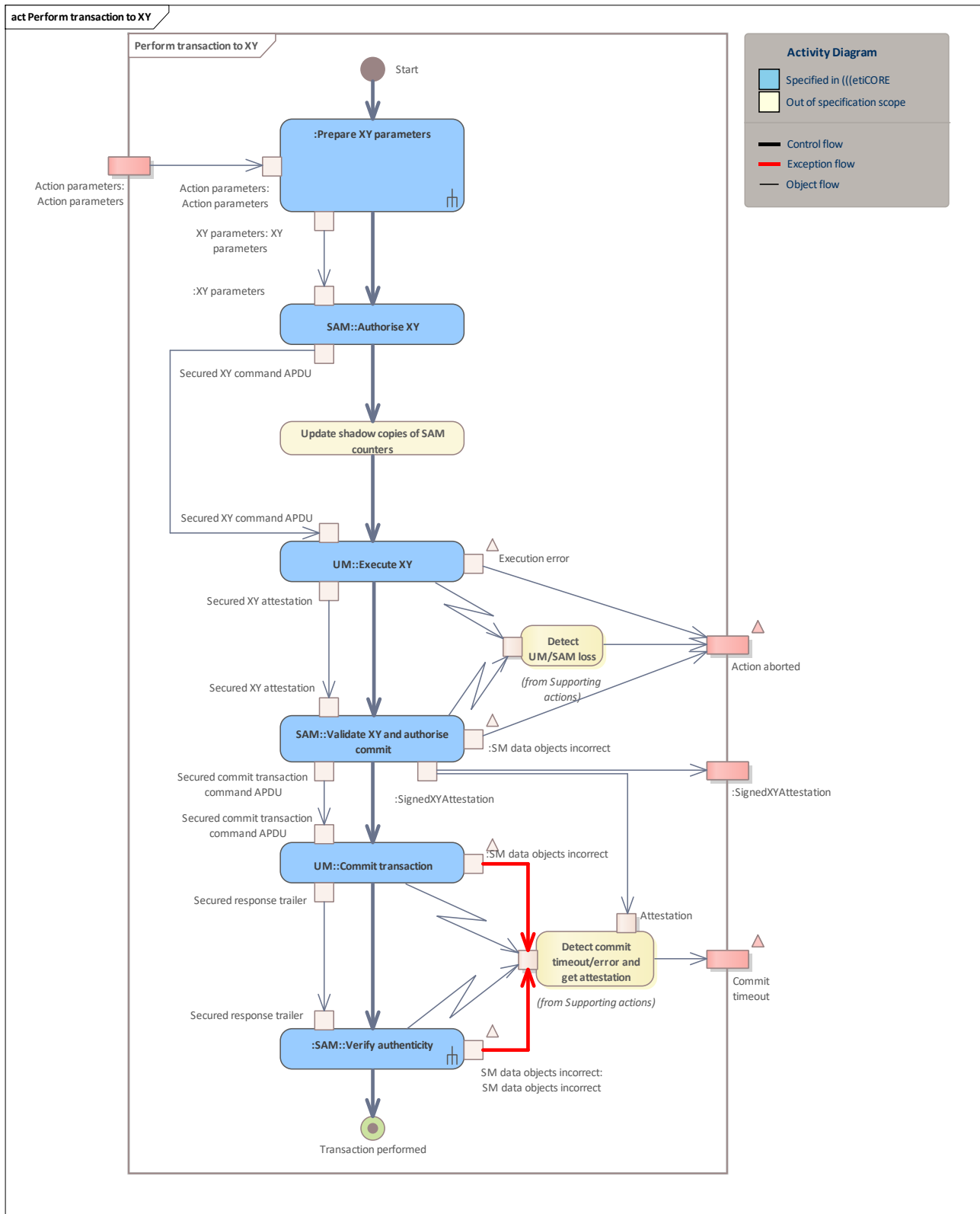


Figure 9: Perform transaction to XY

### 3.1.1.1

See [Prepare XY parameters](#)

### 3.1.1.2 SAM::Authorise XY

Call the SAM operation AUTHORISE\_ENTITLEMENT/AUTHORISE\_ACTION using the prepared parameters.

Since the SAM performs a roll-back in case of an error, no special error handling is shown here. If the SAM configuration does not allow the transaction to be completed, the error scenario "Action aborted" should be followed. This is not shown here, since the terminal should be able to anticipate this situation, e.g. by retrieving the SAM product issuance rights before trying to AUTHORISE\_ENTITLEMENT to make sure the SAM is configured to issue the product in question.

### 3.1.1.3 SAM::Validate XY and authorise commit

Call the SAM operation VALIDATE\_ACTION using the secured attestation prepared by the UM. The only relevant error code in this situation is the error code *SM data objects incorrect*. All other error codes should either not happen at all or be easy to correct by the terminal. If that is not true for any reason, proceed as if the action had to be aborted. If the connection to the SAM or UM is lost during the execution of this command, the transaction is also aborted.

### 3.1.1.4 UM::Commit transaction

Call the UM operation COMMIT\_TRANSACTION using the secured attestation prepared by the UM.

The only relevant error code in this situation is the error code *SM data objects incorrect*. All other error codes should either not happen at all or be easy to correct by the terminal. If that is not true for any reason, proceed as if the action had to be aborted. If the connection to the UM is lost during the execution of this command such that it cannot be ruled out that the command successfully ran on the UM side (and only the response was not transmitted), the *commit timeout* error scenario has to be followed.

### 3.1.1.5 UM::Execute XY

Call the UM operation ISSUE\_ENTITLEMENT / EXECUTE\_ACTION using the Command APDU readily prepared by the SAM.

In case of an error, the terminal cannot correct the message to eliminate the problem since it is, and has to be, secured (encrypted and MAC-signed) using the session keys only SAM and UM know. Thus, the operation has to be re-authorised requiring the use of further SAM counters, effectively aborting the transaction since it may - on a business level - also affect other actions within the same transaction or may already have (in case of the error code *SM data objects incorrect*) reset the session on the UM side. For the sake of simplicity, this is not distinguished here and the whole transaction is aborted.

If the connection to the SAM or UM is lost during the execution of this command, the transaction is also aborted.

### 3.1.1.6 Update shadow copies of SAM counters

Update the shadow copies of the SAM counters.

For AUTHORISE\_ACTION this means incrementing the SAM action counter by one.

For AUTHORISE\_ENTITLEMENT this means incrementing the SAM entitlement issuance counter and the product issuance counter corresponding to the issued product by one.

### 3.1.2 Prepare XY parameters

Composes the parameters of the UM operation (ISSUE\_ENTITLEMENT/EXECUTE\_ACTION) that are set by the terminal for action parameters. These parameters are then given to the authorising operation of the SAM (AUTHORISE\_ENTITLEMENT/AUTHORISE\_ACTION) and further adjusted by the SAM yielding a fully prepared Command APDU to be used for the UM.

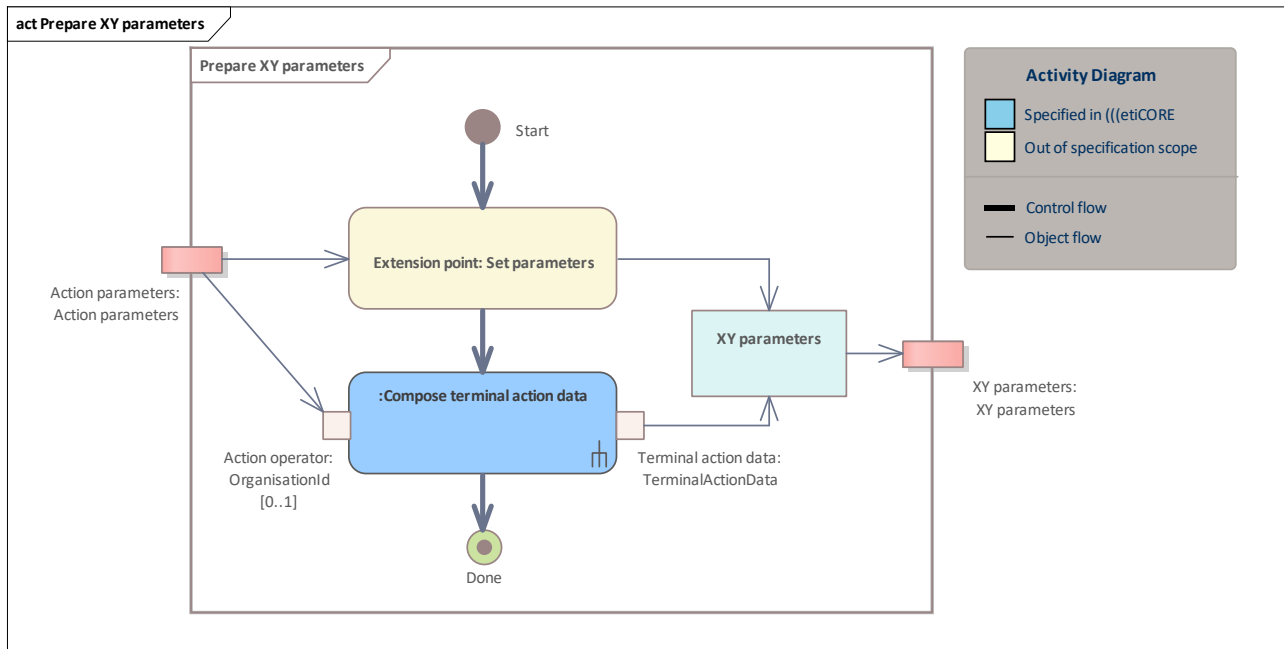


Figure 10: Prepare XY parameters

#### 3.1.2.1 Extension point: Set parameters

This extension point is a placeholder for the use case-specific preparatory steps to create an action parameters.

For entitlement actions, the product ID given via the parameters is always part of the action parameters.

For application actions, a pseudo product ID is constructed using the UM Owner ID as the PO Org ID.

Gives an overview of the supporting classes used for the pattern diagrams.

Figure 11: Supporting classes

May contain additional, use case-specific parameters, e.g. action tariff parameters.

Parameters for the action used to call the SAM operation authorising the action on the UM side.

Also identifies the UM so that back-office systems can retrieve the corresponding certificate and verify the signature.

Additional parameters to insert into the notification bearing the signed attestation.



### 3.2.5 XYNotification

Every XY notification consists of the following three building blocks:

- an XY attestation signed by the UM
- for some use cases: additional information (see [Notification parameters](#))
- optional: a warning list

### 3.2.6 ForwardedXYNotification

A forwarded XY notification is a wrapper around the XY notification that may carry additional events in a separate event list.

### 3.2.7 tNotifyXY

Element used to transmit an XY notification from the terminal to its back-office system.

### 3.2.8 tNotifyXYAborted

Element used to transmit the information about an aborted XY action from the terminal to its back-office system.

### 3.2.9 notifyXY

Element used to inform the product owner about an entitlement action execution and to inform the owner of an application about an application action execution.

### 3.2.10 notifyXYResponse

Element used in response to [notifyXY](#) for normal process termination.

### 3.2.11 notifyXYException

Element used in reply to [notifyXY](#) for abnormal process termination.

### 3.2.12 forwardXYNotification

Element used to inform the entitlement owner about an entitlement action execution by a third party.

### 3.2.13 forwardXYNotificationResponse

Element used in response to [forwardXYNotification](#) for normal process termination.

### 3.2.14 forwardXYNotificationException

Element used in reply to [forwardXYNotification](#) for abnormal process termination.

## 3.3 Perform entitlement XY and notify



See [Perform entitlement XY and notify](#)

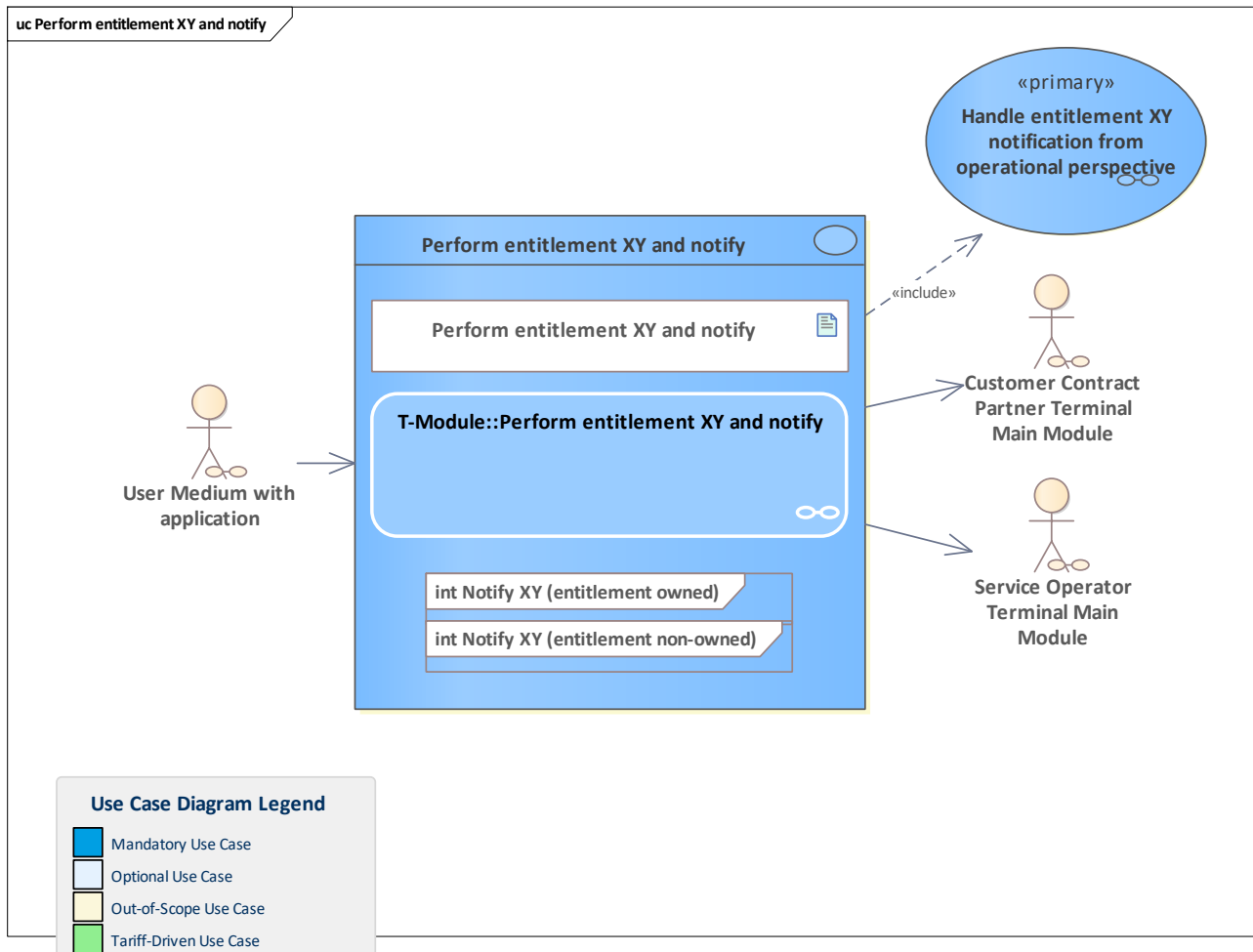


Figure 12: Perform entitlement XY and notify

### 3.3.1 Perform entitlement XY and notify

The user medium-based entitlement action XY is performed by the terminal in conjunction with a SAM and the back-office system corresponding to the terminal is notified about the action execution.

Combines the transaction including the action XY and the notification processes triggered by this. Shows the interaction between the UM transaction and the notification processes. The timeout warning is generated in the corresponding diagram if needed.

The business preconditions to perform the transaction (e.g. determining the actual need to block an entitlement or making sure that a stored-value payment method is sufficiently charged) are already satisfied before the process shown in this diagram begins.

**Note:**

Executing orders in the context of ordered action management involves the same UM operations as outside of that scope, but leads to different notification processes (partly belonging to different notification categories). In this case, only one diagram "Perform transaction to XY" may be needed, which is used in two diagrams "T-XYZ::Perform XY and notify" combining it with different notification activities (i.e. the ordered and the regular variant).

### 3.3.2 T-Module::Perform entitlement XY and notify

See [Perform entitlement XY and notify](#)

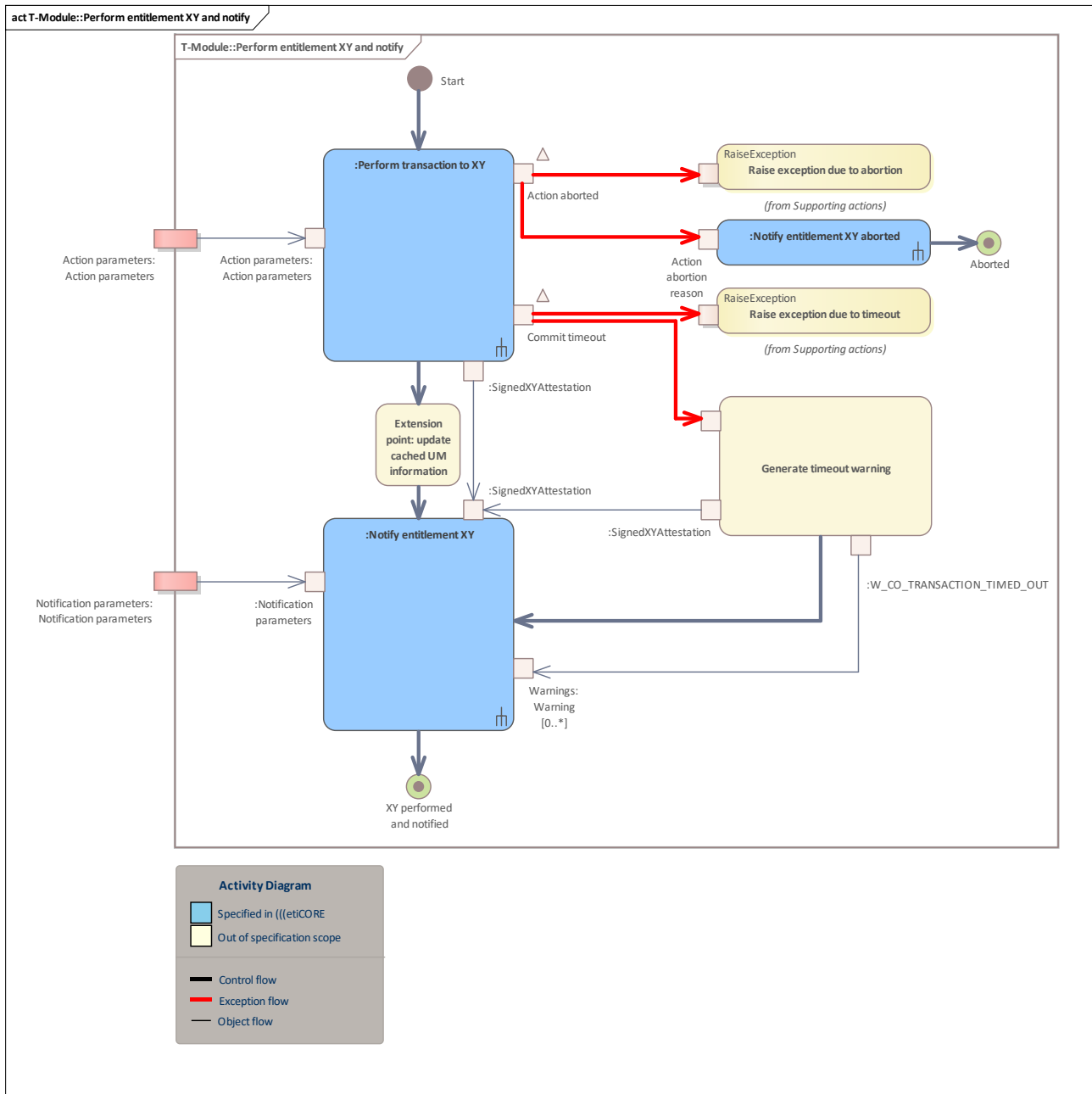


Figure 13: T-Module::Perform entitlement XY and notify

#### 3.3.2.1

See [Perform transaction to XY](#)

#### 3.3.2.2

See [Notify entitlement XY](#)

### 3.3.2.3

See [Notify entitlement XY aborted](#)

### 3.3.2.4 Extension point: update cached UM information

Update the cached information about the UM as appropriate for the executed action, e.g. entitlement action counter, application status, etc.

### 3.3.2.5 Generate timeout warning

Generate a timeout warning to be placed into the notification sent about the executed action.

## 3.3.3 Notify entitlement XY

The terminal notifies its back-office system about a successful action execution.

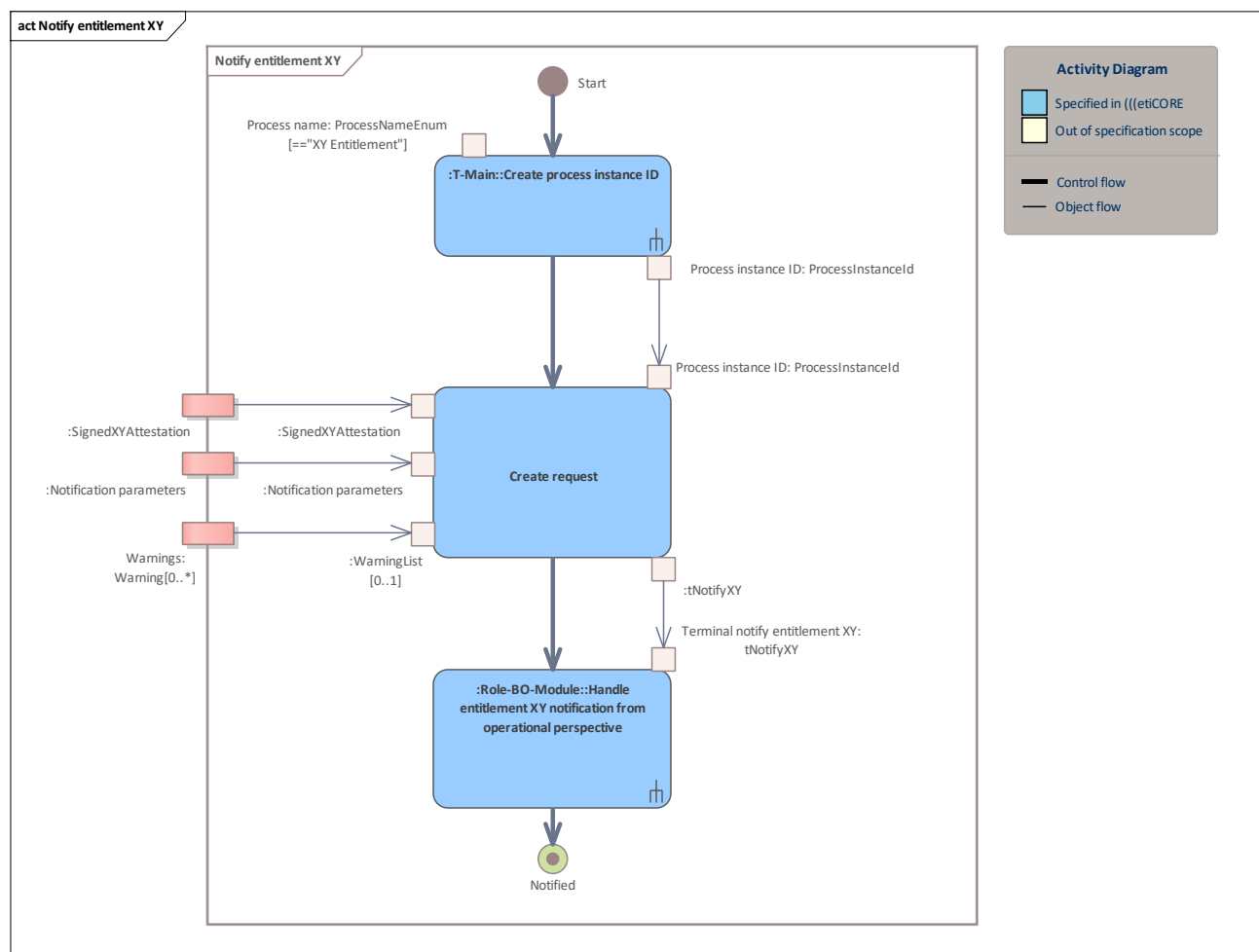


Figure 14: Notify entitlement XY

### 3.3.4 Notify entitlement XY aborted

The terminal notifies its back-office system about an aborted action.

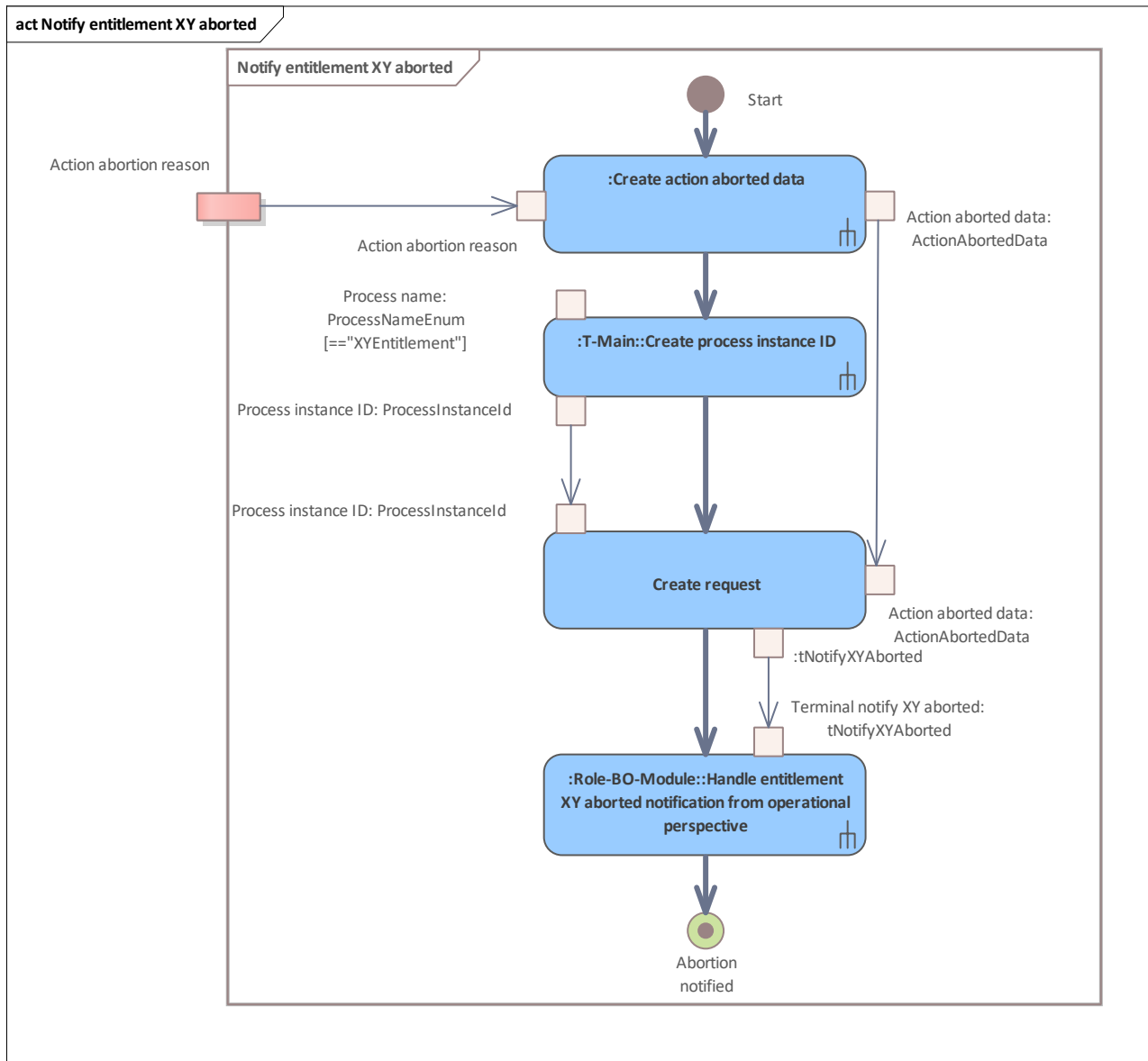


Figure 15: Notify entitlement XY aborted

### 3.3.5 Notify entitlement XY aborted based on attestation

If there are several actions within a transaction, this variant is to be used for the actions already completed (but not committed).

Since the attestation is already available, we can extract the required data from it instead of reproducing it.

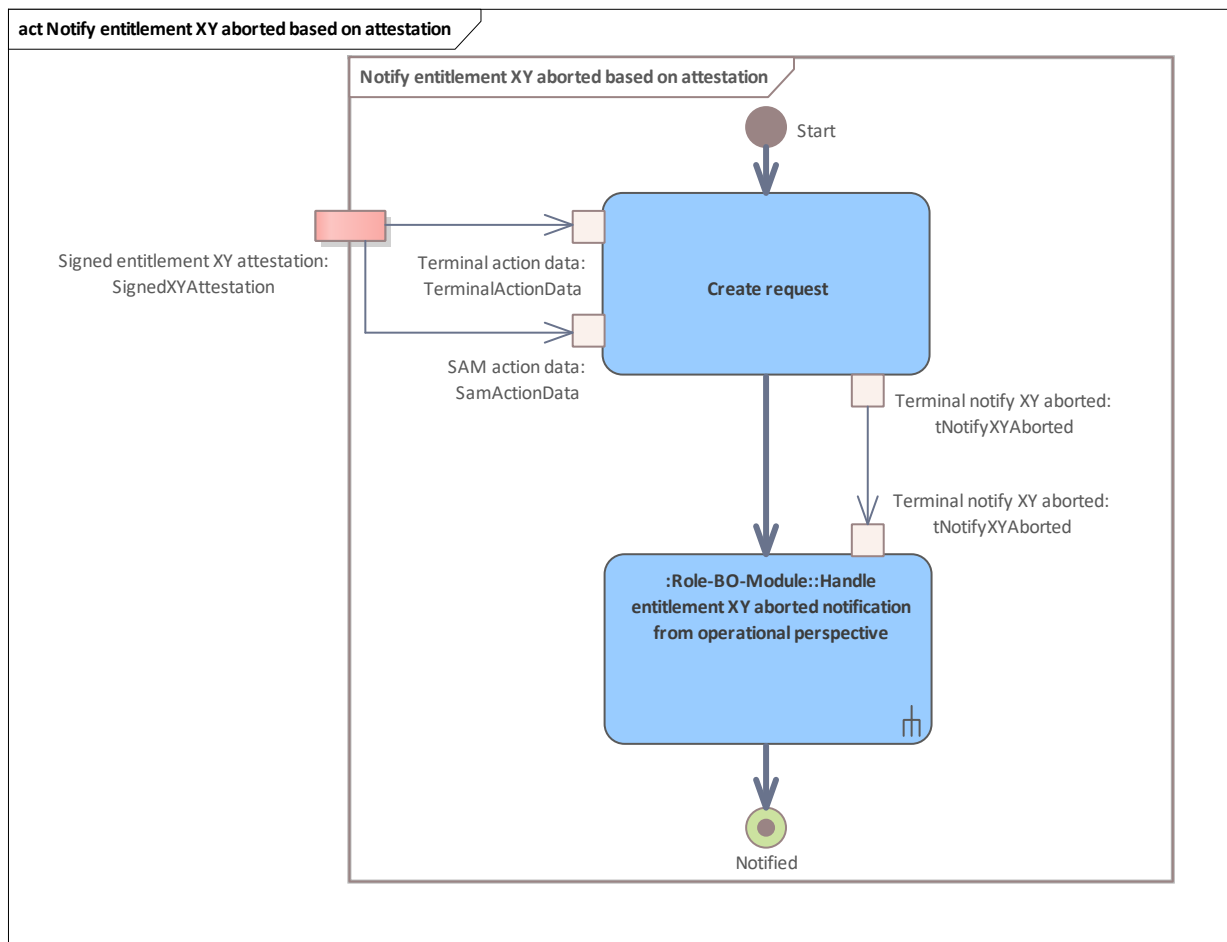


Figure 16: Notify entitlement XY aborted based on attestation

### 3.3.6 Notify XY (entitlement owned)

Shows the message chain starting in a terminal (normally customer contract partner terminal), sent to a terminal operator back-office system (primary customer contract partner system) and finally sent to the product owner system.

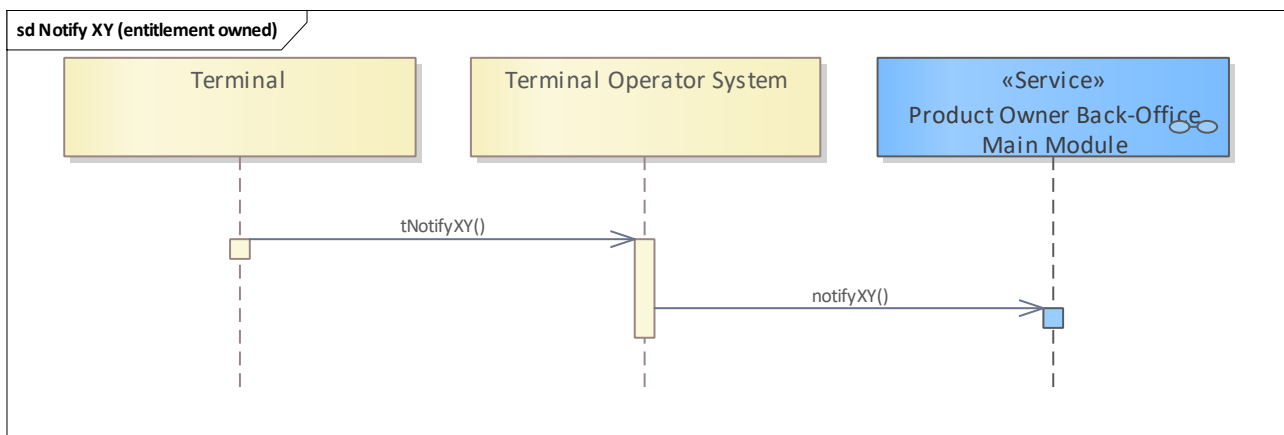


Figure 17: Notify XY (entitlement owned)

See [Notify XY \(entitlement owned\)](#)

### 3.3.7 Notify XY (entitlement non-owned)

Shows the message chain starting in a terminal (customer contract partner terminal or service operator terminal), sent to a terminal operator back-office system (customer contract partner system or service operator system) then sent to the product owner system and finally to the owning primary customer contract partner system.

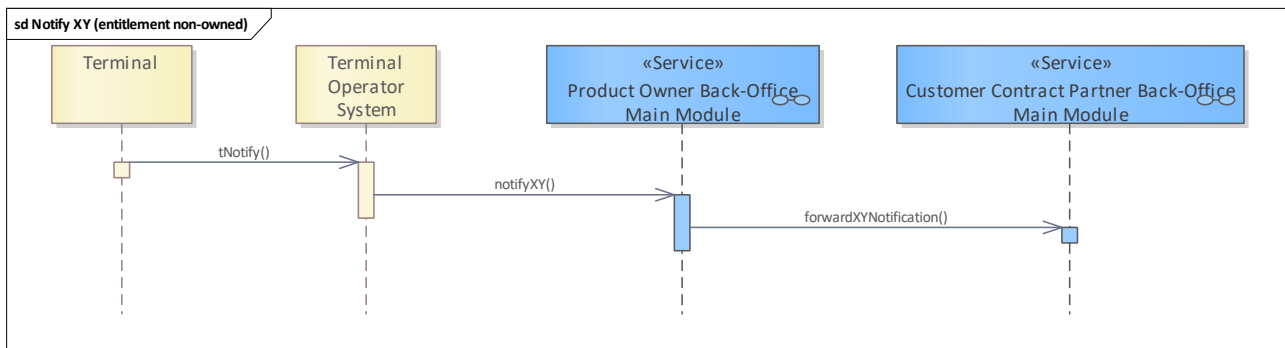


Figure 18: Notify XY (entitlement non-owned)

See [Notify XY \(entitlement non-owned\)](#).

## 3.4 Perform application XY and notify

See [Perform application XY and notify](#)

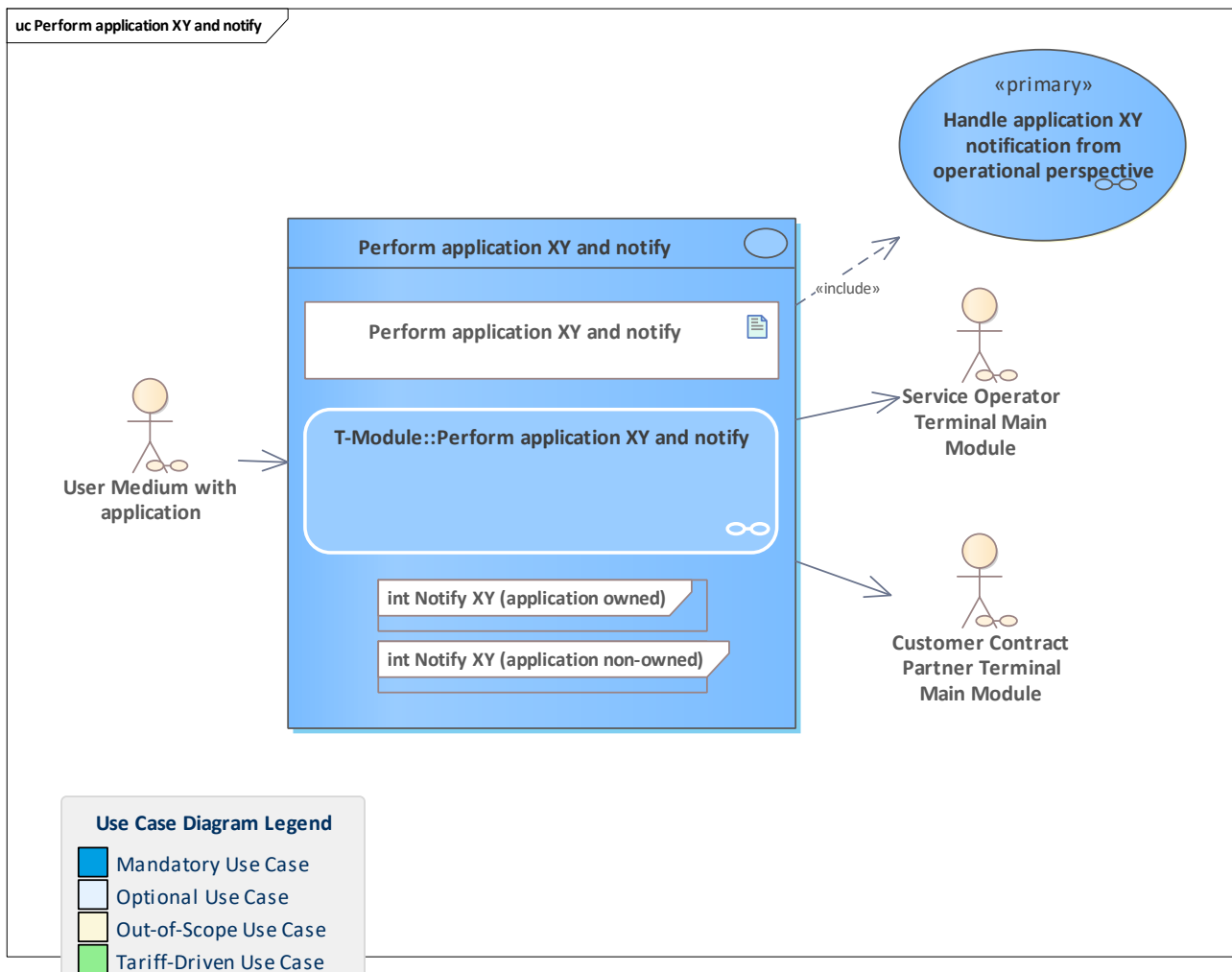


Figure 19: Perform application XY and notify

### 3.4.1 Perform application XY and notify

The UM application action XY is performed by the terminal in conjunction with a SAM and the back-office system corresponding to the terminal is notified about the action execution.

Combines the transaction including the action XY and the notification processes triggered by this. Shows the interaction between the UM transaction and the notification processes. The timeout warning is generated in this diagram if needed.

The business preconditions to perform the transaction (e.g. determining the actual need to block an application) are already satisfied before the process shown in this diagram begins.

### 3.4.2 T-Module::Perform application XY and notify

See [Perform application XY and notify](#)

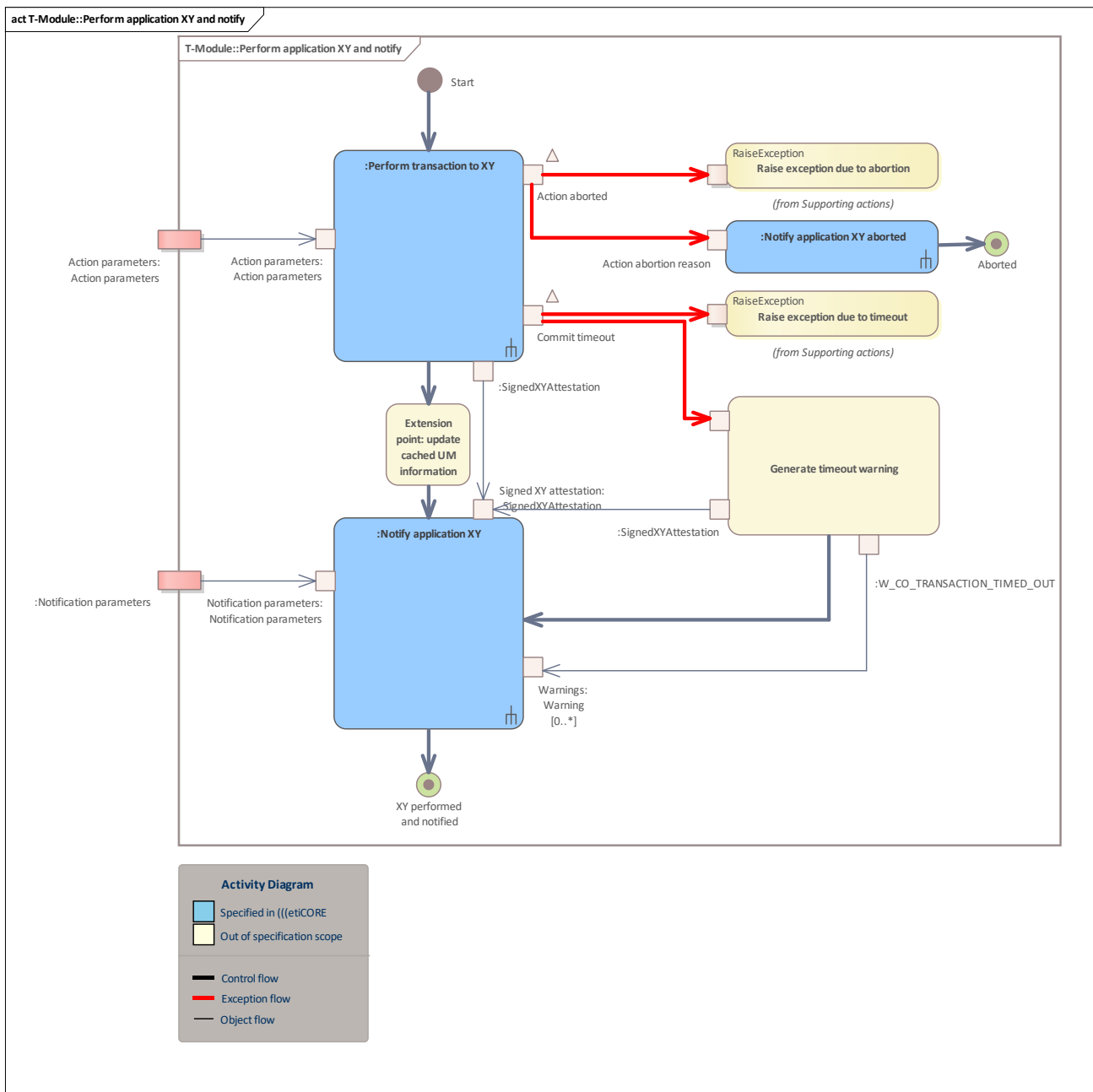


Figure 20: T-Module::Perform application XY and notify

### 3.4.2.1

See [Perform transaction to XY](#)

### 3.4.2.2

See [Notify entitlement XY aborted](#)

### 3.4.2.3

See [Notify entitlement XY](#)



### 3.4.2.4 Extension point: update cached UM information

Update the cached information about the UM as appropriate for the executed action, e.g. entitlement action counter, application status, etc.

### 3.4.2.5 Generate timeout warning

Generate a timeout warning to be placed into the notification sent about the executed action.

## 3.4.3 Notify application XY

The terminal notifies its back-office system about a successful action execution.

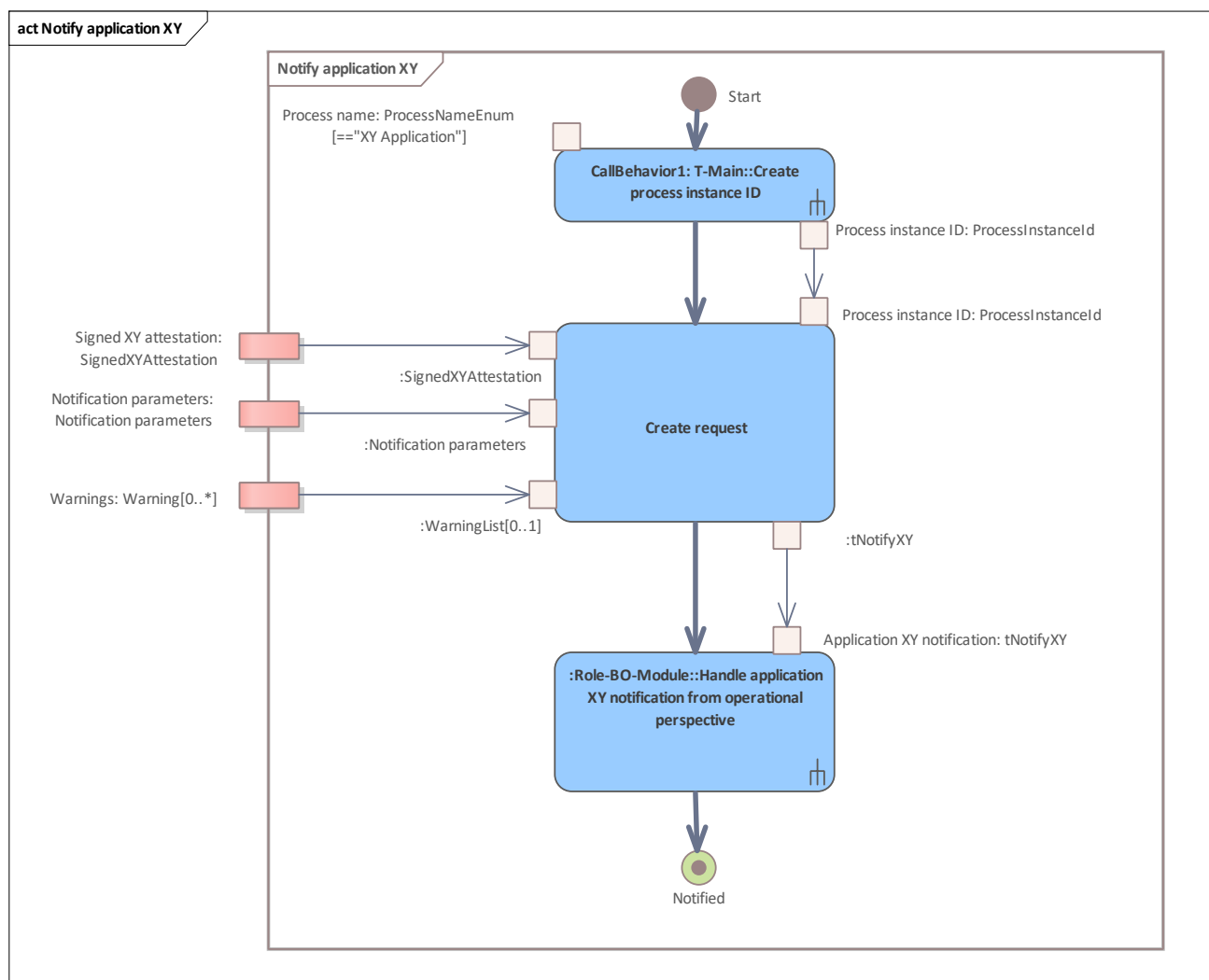


Figure 21: Notify application XY

### 3.4.4 Notify application XY aborted

The terminal notifies its back-office system about an aborted action.

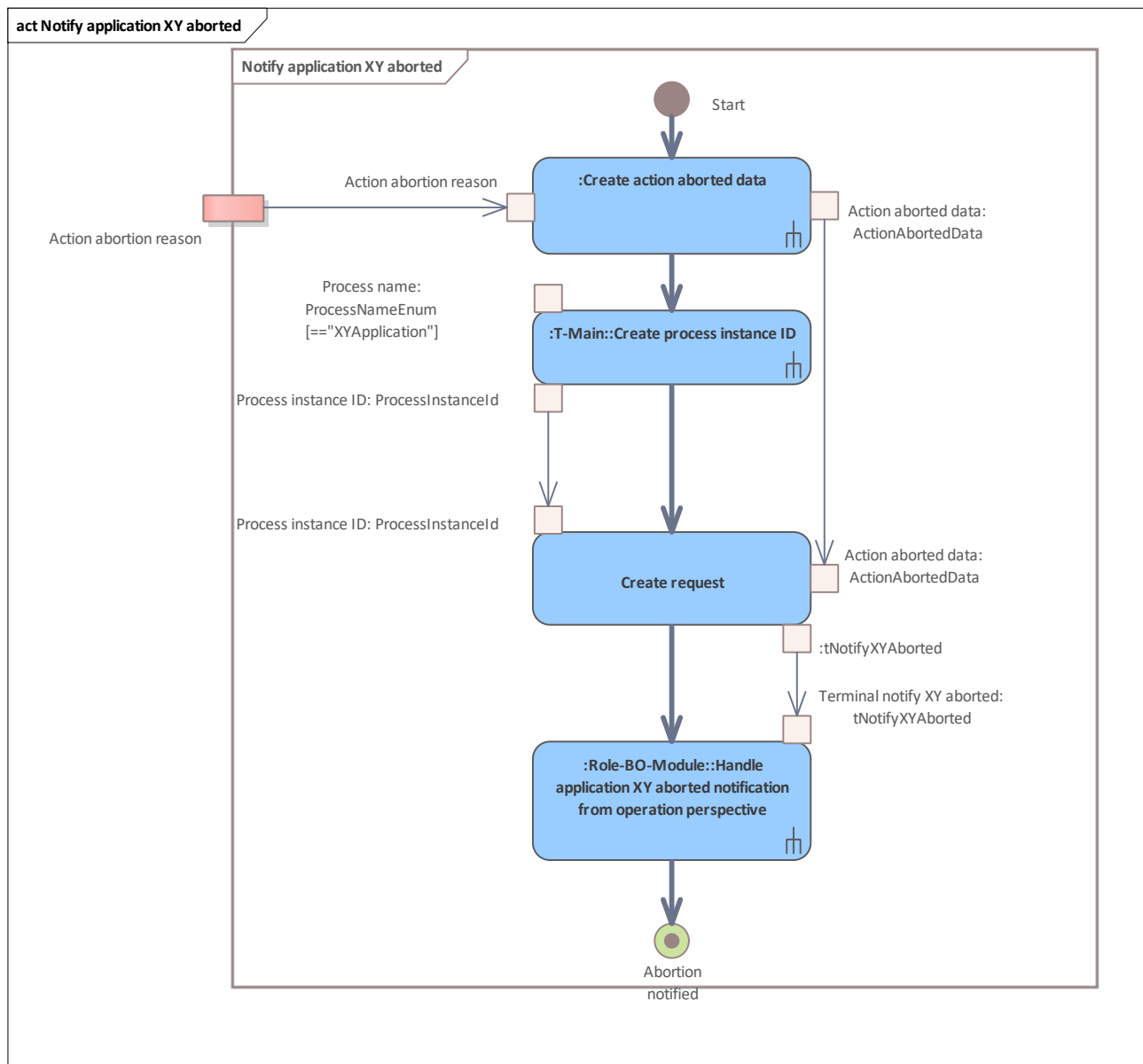


Figure 22: Notify application XY aborted

### 3.4.5 Notify XY (application owned)

Shows the message chain starting in a terminal (normally customer contract partner terminal), sent to a terminal operator back-office system (primary customer contract partner system).

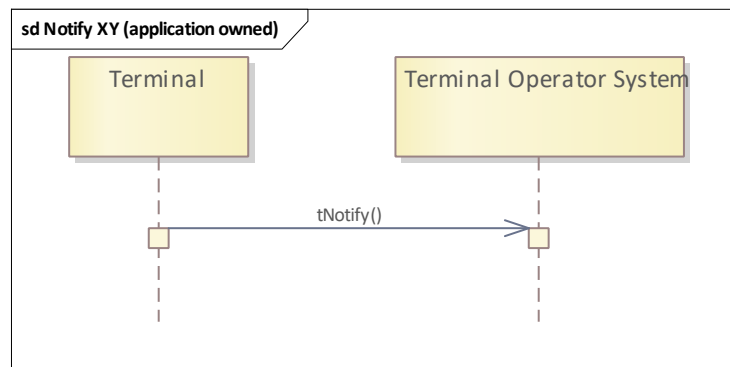


Figure 23: Notify XY (application owned)

See [Notify XY \(application owned\)](#).

### 3.4.6 Notify XY (application non-owned)

Shows the message chain starting in a terminal (customer contract partner terminal or service operator terminal), sent to a terminal operator back-office system (customer contract partner system or service operator system) and finally to the owning primary customer contract partner system.

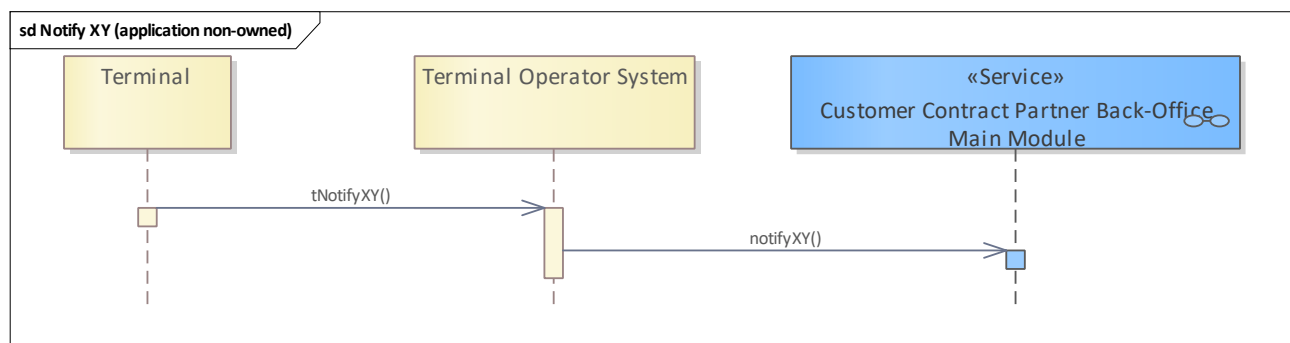


Figure 24: Notify XY (application non-owned)

See [Notify XY \(application non-owned\)](#).

## 3.5 Handle entitlement XY notification from operational perspective

See [Handle entitlement XY notification from operational perspective](#)

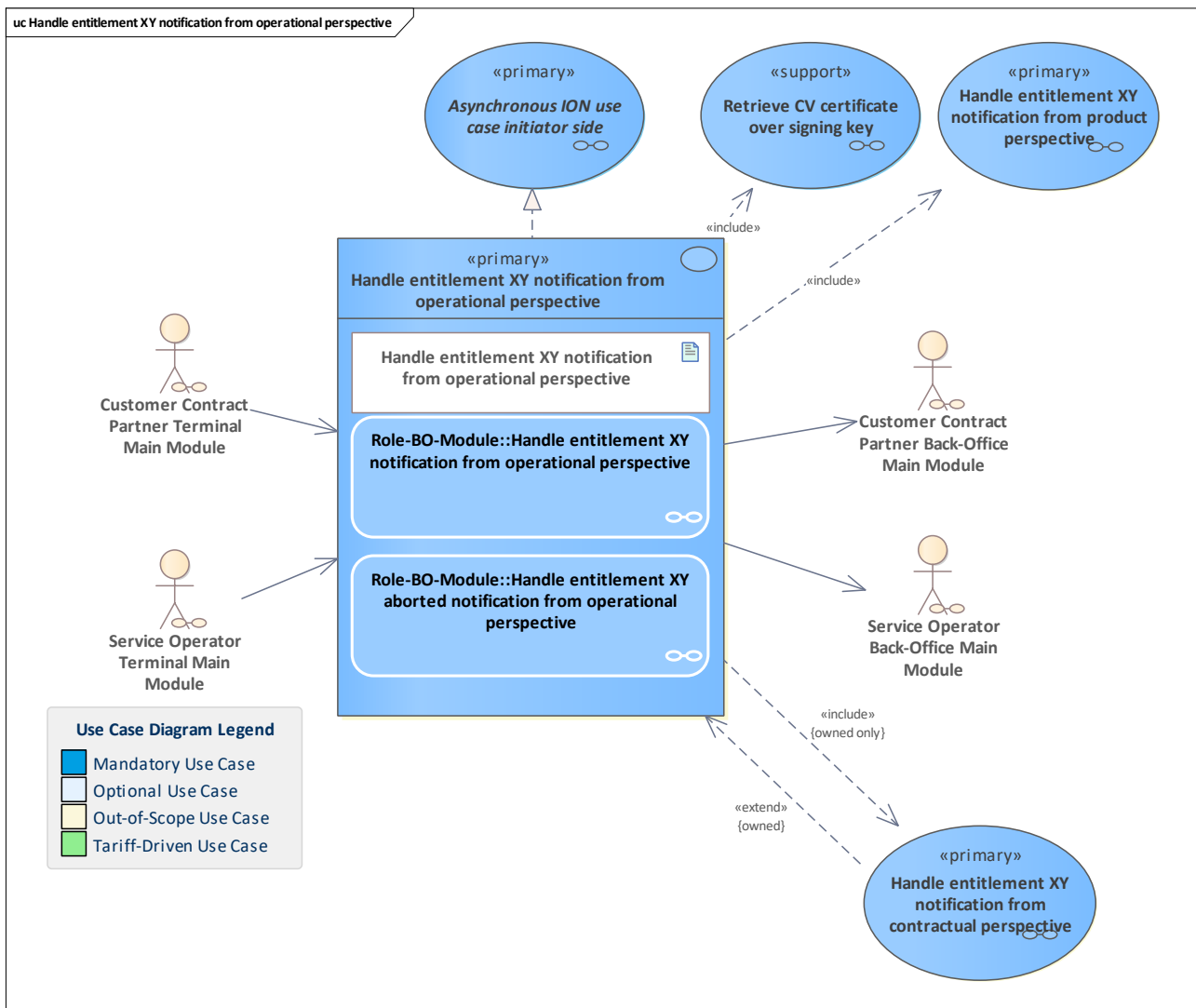


Figure 25: Handle entitlement XY notification from operational perspective

### 3.5.1 Handle entitlement XY notification from operational perspective

A back-office system belonging to a terminal that executed a UM action with an entitlement processes the notification about that action execution.

The processing is done from the operational perspective, focusing on the terminal-side of the action execution, i.e. logging the used SAM counter values.

Depending on the use case and the ownership of the entitlement the action was executed on, other back-office systems are informed about the action execution.

### 3.5.2 Role-BO-Module::Handle entitlement XY notification from operational perspective

See [Handle entitlement XY notification from operational perspective](#)

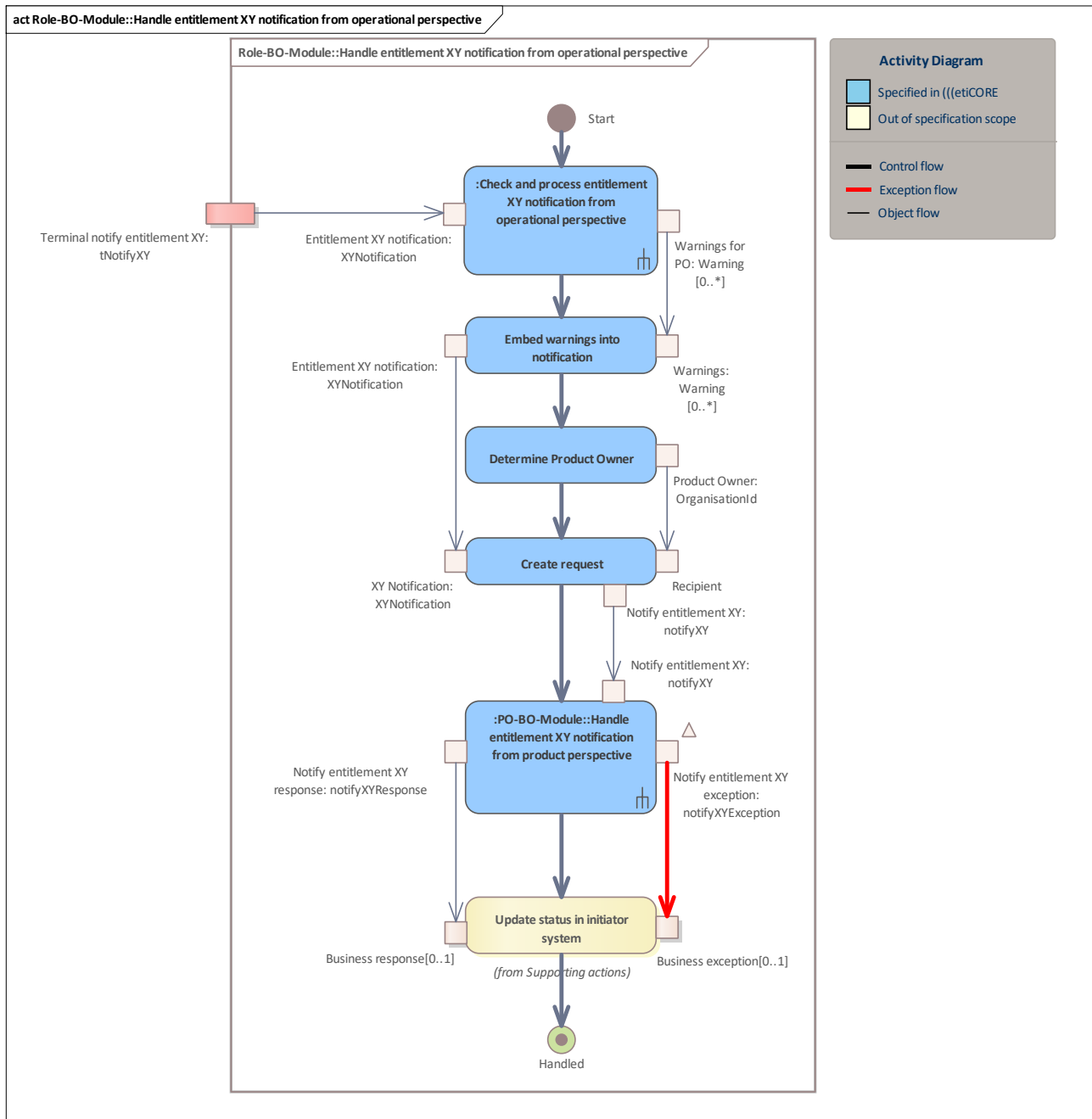


Figure 26: Role-BO-Module::Handle entitlement XY notification from operational perspective

### 3.5.3 Check and process entitlement XY notification from operational perspective

This activity performs the system internal processing of an entitlement-based notification coming from a terminal. All necessary monitoring checks are performed, divided into general checks and notification-specific checks.

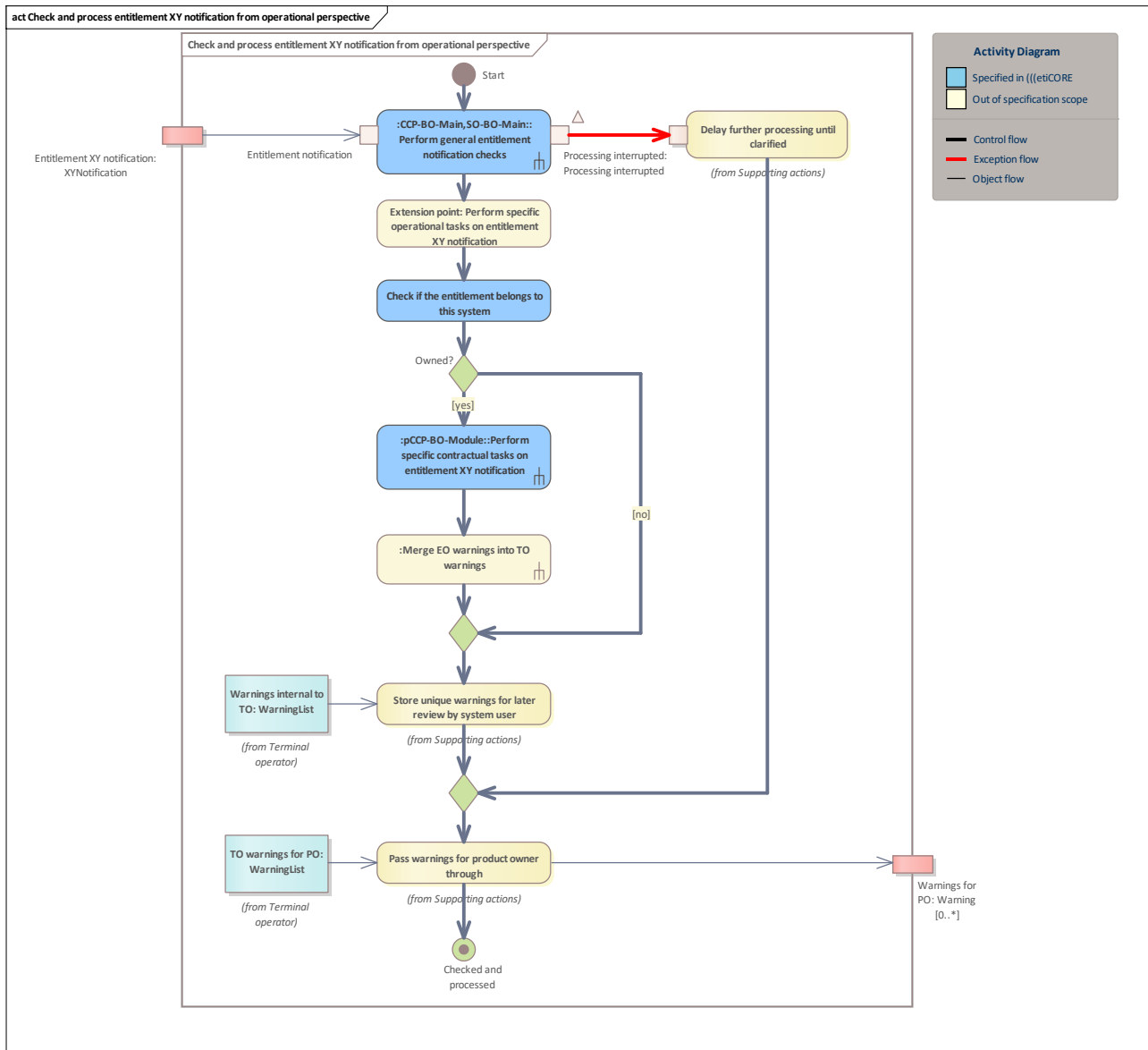


Figure 27: Check and process entitlement XY notification from operational perspective

### 3.5.3.1 Extension point: Perform specific operational tasks on entitlement XY notification

Log SAM action counter value / SAM entitlement issuance counter value / product issuance counter value usage, etc.

### 3.5.4 Role-BO-Module::Handle entitlement XY aborted notification from operational perspective

The back-office system belonging to the terminal handles a notification about an abortion during entitlement XY from an operational perspective.

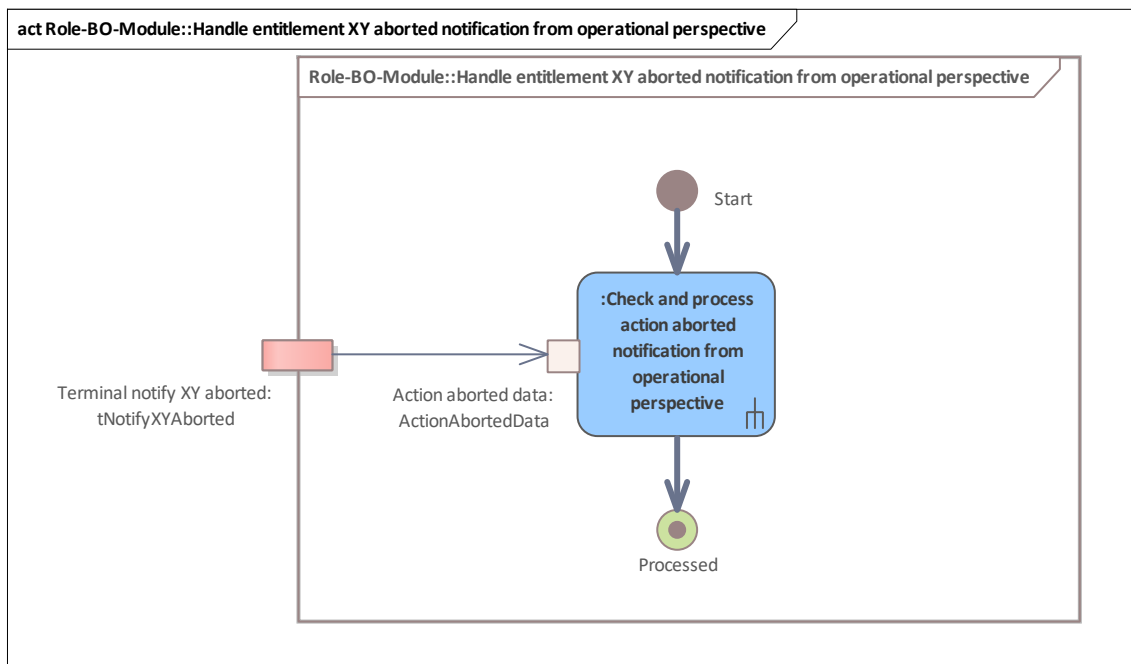


Figure 28: Role-BO-Module::Handle entitlement XY aborted notification from operational perspective

## 3.6 Handle application XY notification from operational perspective

See [Handle application XY notification from operational perspective](#)

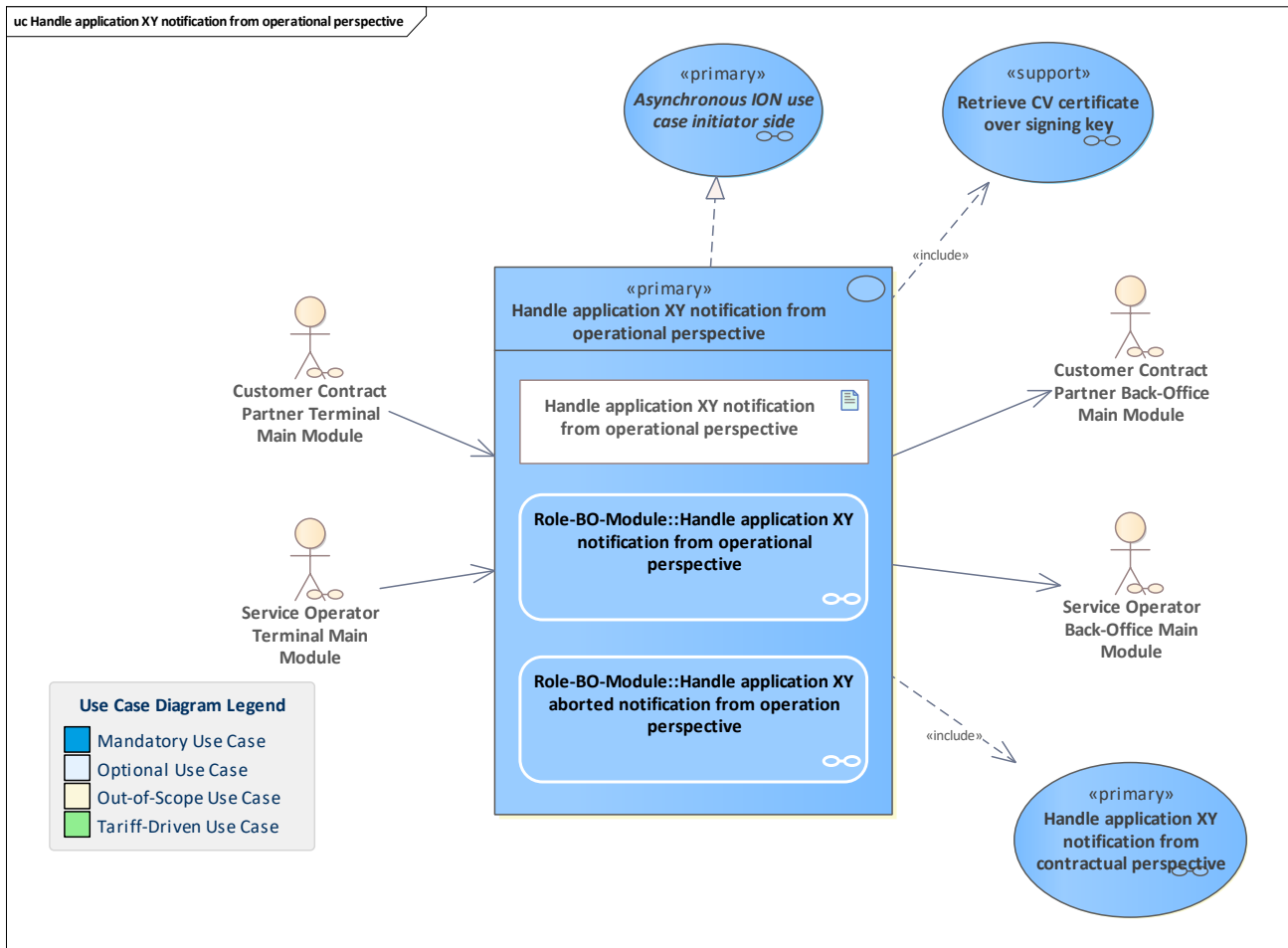


Figure 29: Handle application XY notification from operational perspective

### 3.6.1 Handle application XY notification from operational perspective

A back-office system belonging to a terminal that executed a UM action with an application processes the notification about that action execution.

The processing is done from the operational perspective, focusing on the terminal-side of the action execution, i.e. logging the used SAM counter values.

Depending on the use case and the ownership of the application the action was executed on, other back-office systems are informed about the action execution.

### 3.6.2 Role-BO-Module::Handle application XY notification from operational perspective

See [Handle application XY notification from operational perspective](#)



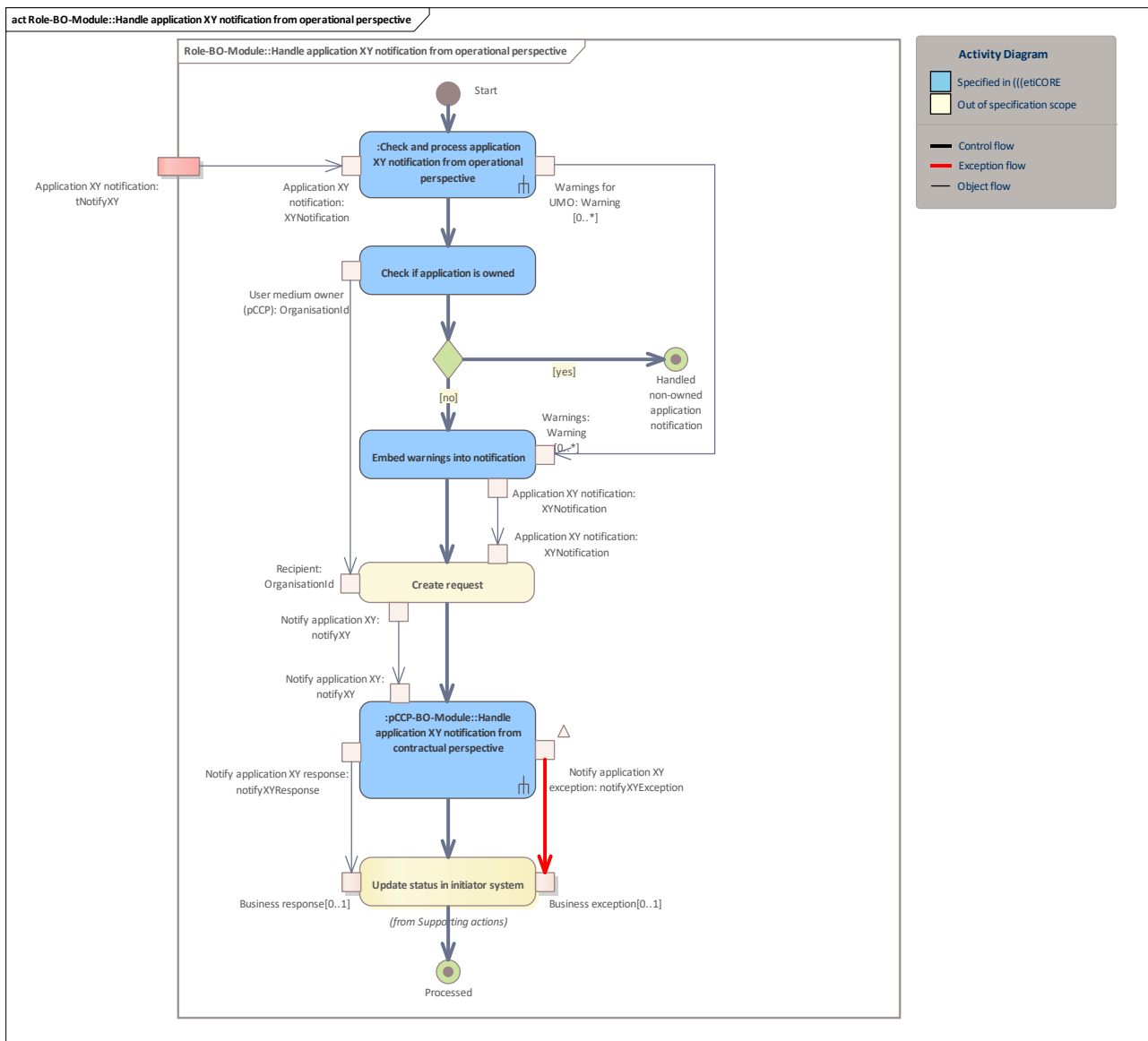


Figure 30: Role-BO-Module::Handle application XY notification from operational perspective

### 3.6.3 Check and process application XY notification from operational perspective

This activity performs the system internal processing of an application-based notification coming from a terminal. All necessary monitoring checks are performed, divided into general checks and notification specific checks.

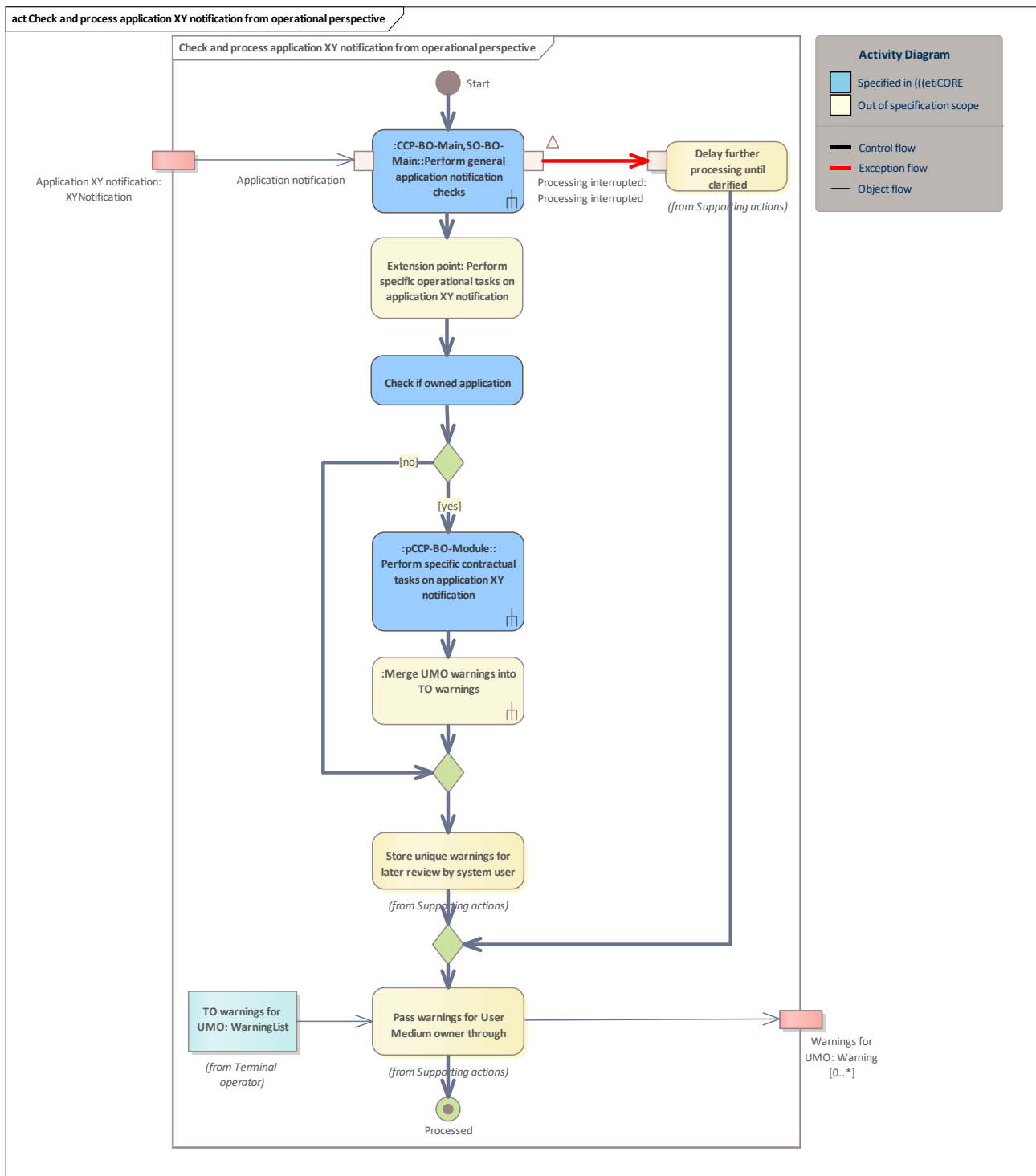


Figure 31: Check and process application XY notification from operational perspective

### 3.6.3.1 Extension point: Perform specific operational tasks on application XY notification

Log SAM action counter value, etc.

### 3.6.4 Role-BO-Module::Handle application XY aborted notification from operation perspective

The back-office system belonging to the terminal handles a notification about an abortion during application XY from an operational perspective.

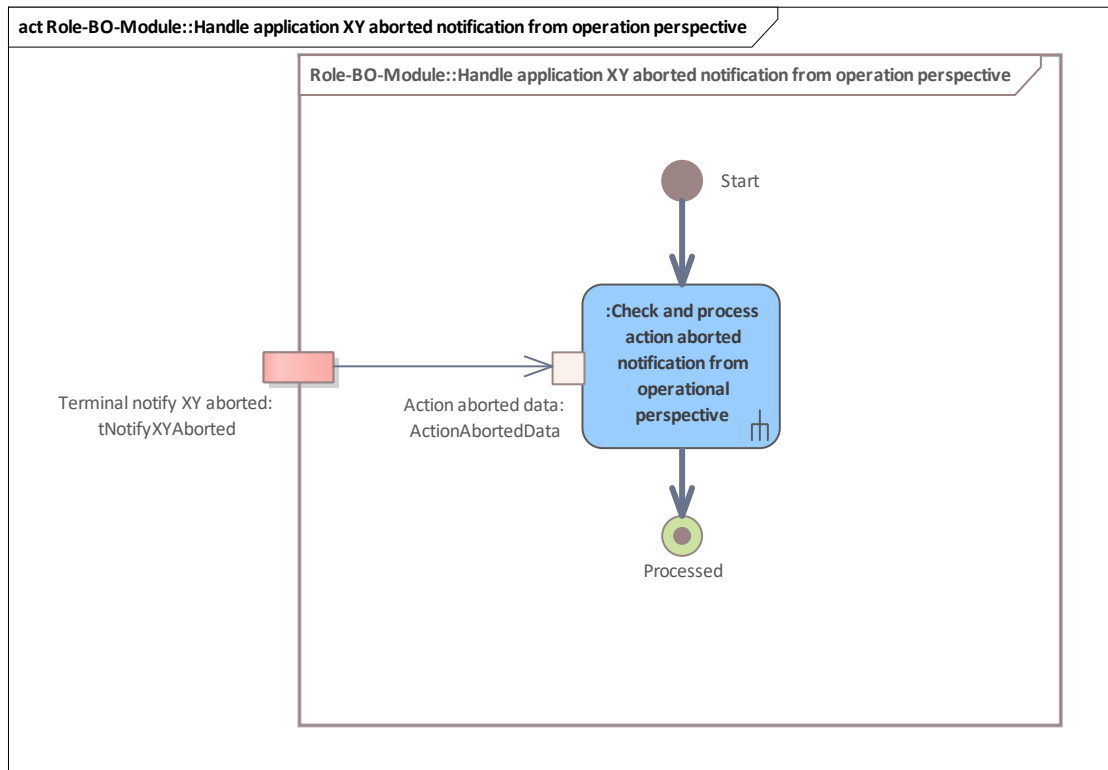
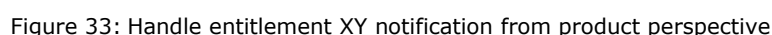


Figure 32: Role-BO-Module::Handle application XY aborted notification from operation perspective

## 3.7 Handle entitlement XY notification from product perspective

See [Handle entitlement XY notification from product perspective](#)



A PO system processes a notification about an entitlement action executed on an entitlement that is an instance of an owned product. The processing is done from the product perspective, focusing on the entitlement lifecycle and tariff aspects.

See [Handle entitlement XY notification from product perspective](#)

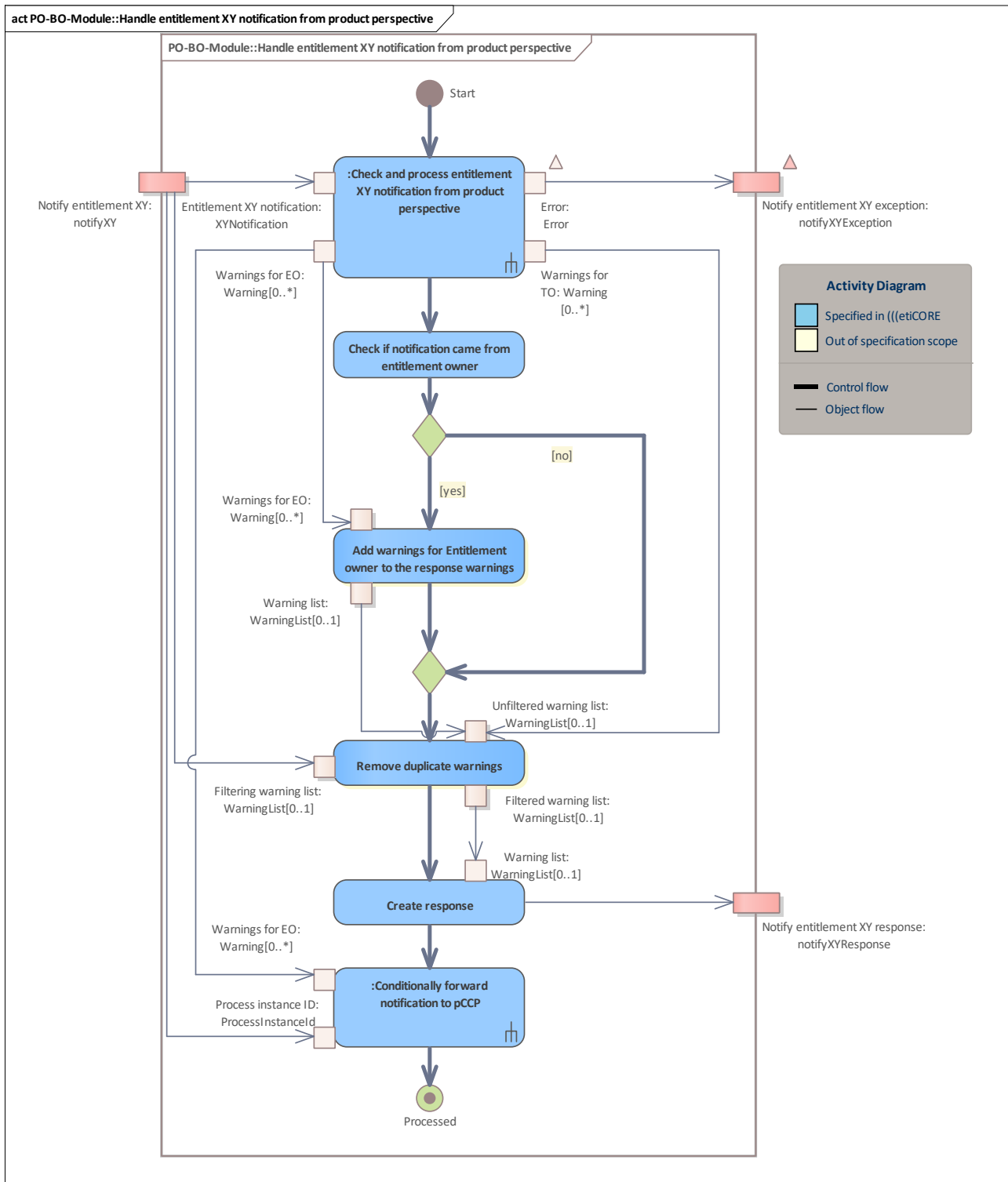


Figure 34: PO-BO-Module::Handle entitlement XY notification from product perspective

### 3.7.3 Check and process entitlement XY notification from product perspective

This activity performs the system internal processing of an entitlement-based notification coming from the terminal operator system (SO or CCP). All necessary monitoring checks are performed, divided into general checks and notification specific checks.

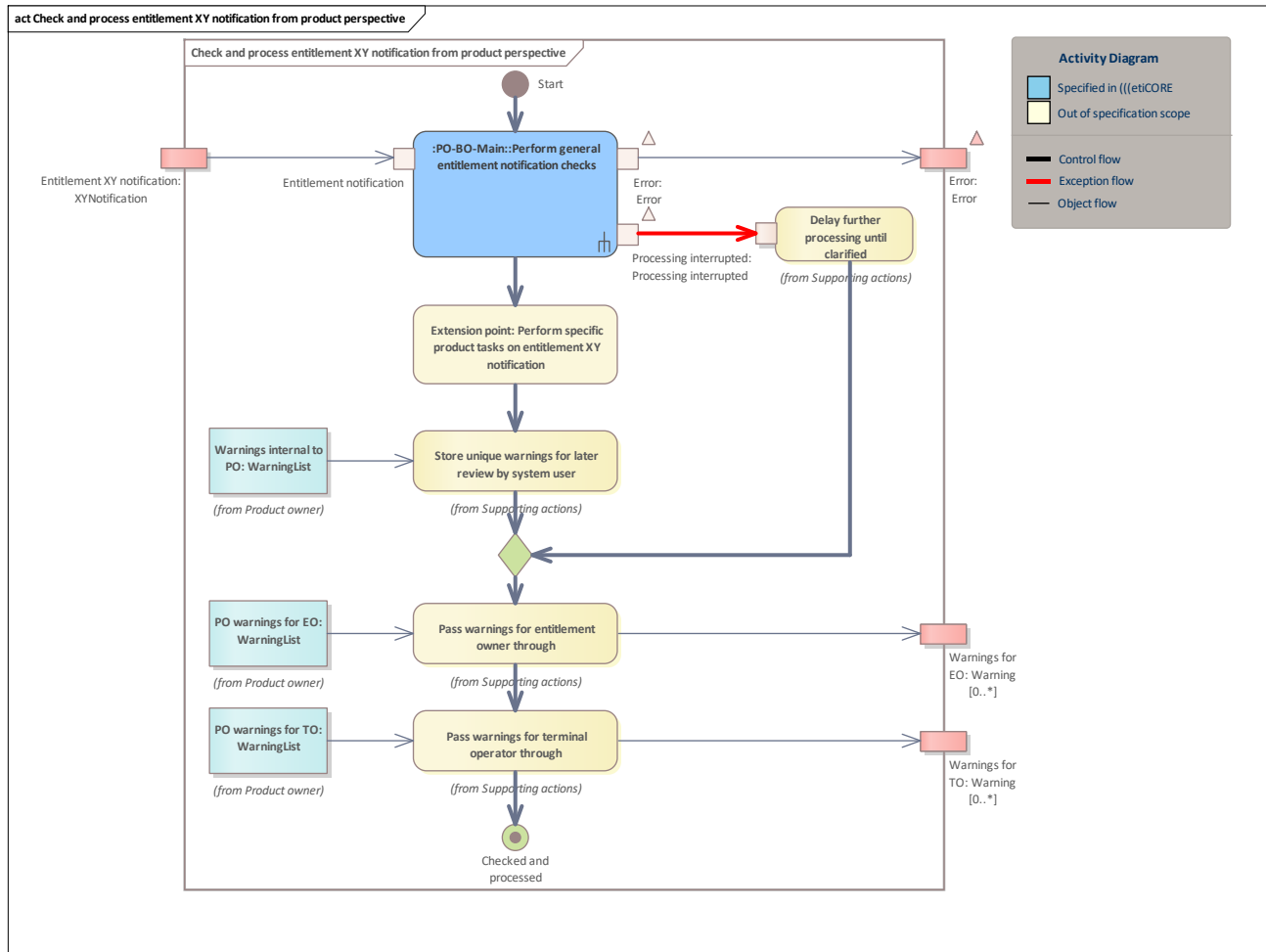


Figure 35: Check and process entitlement XY notification from product perspective

### 3.7.4 Conditionally forward notification to pCCP

Activity that is used if the sender of the notification was not owner of the entitlement. In this case, the notification is forwarded to the primary customer contract partner.

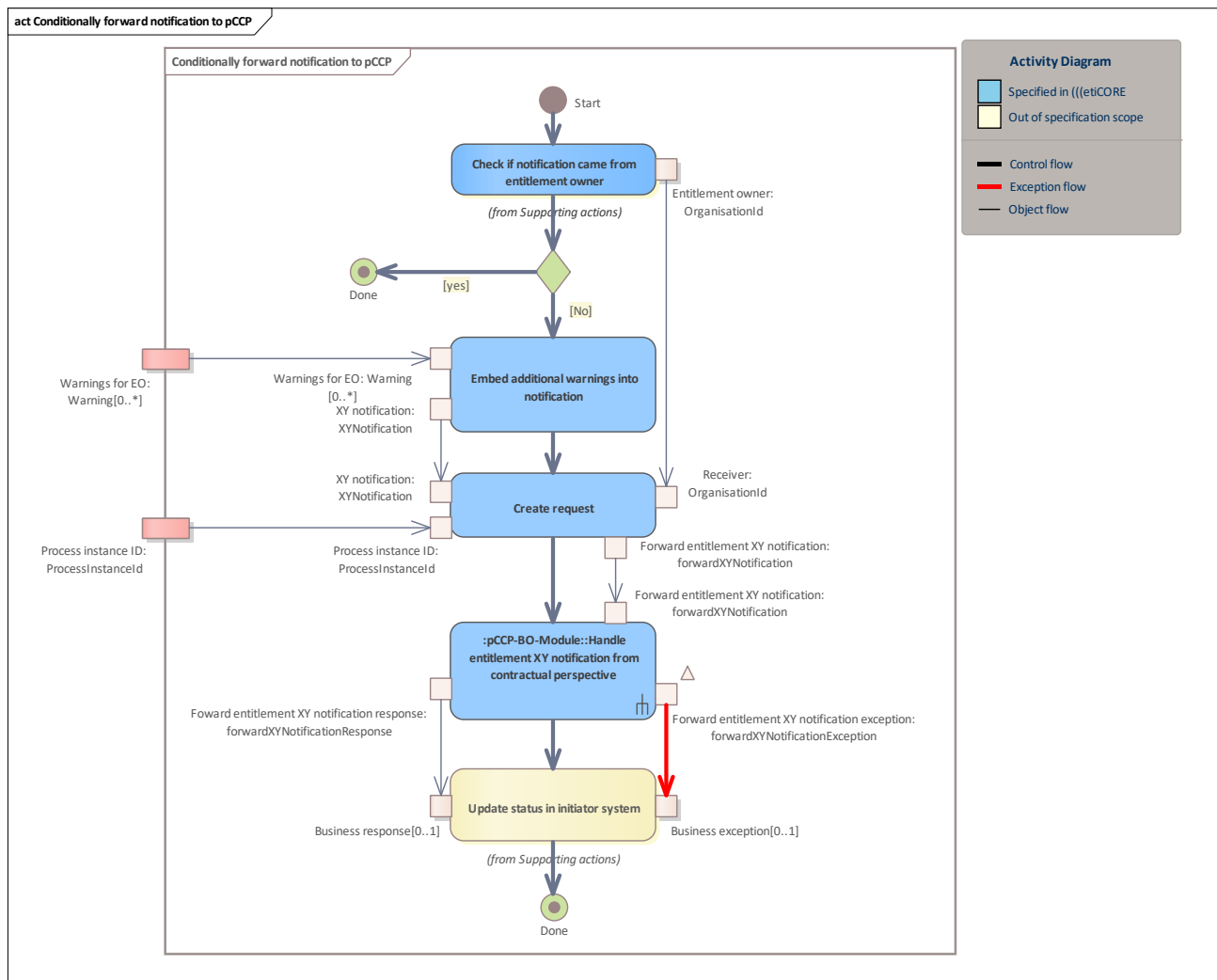


Figure 36: Conditionally forward notification to pCCP

### 3.7.4.1 Embed additional warnings into notification

Embed the given warnings into the notification currently being handled.

## 3.8 Handle entitlement XY notification from contractual perspective

See [Handle entitlement XY notification from contractual perspective](#)

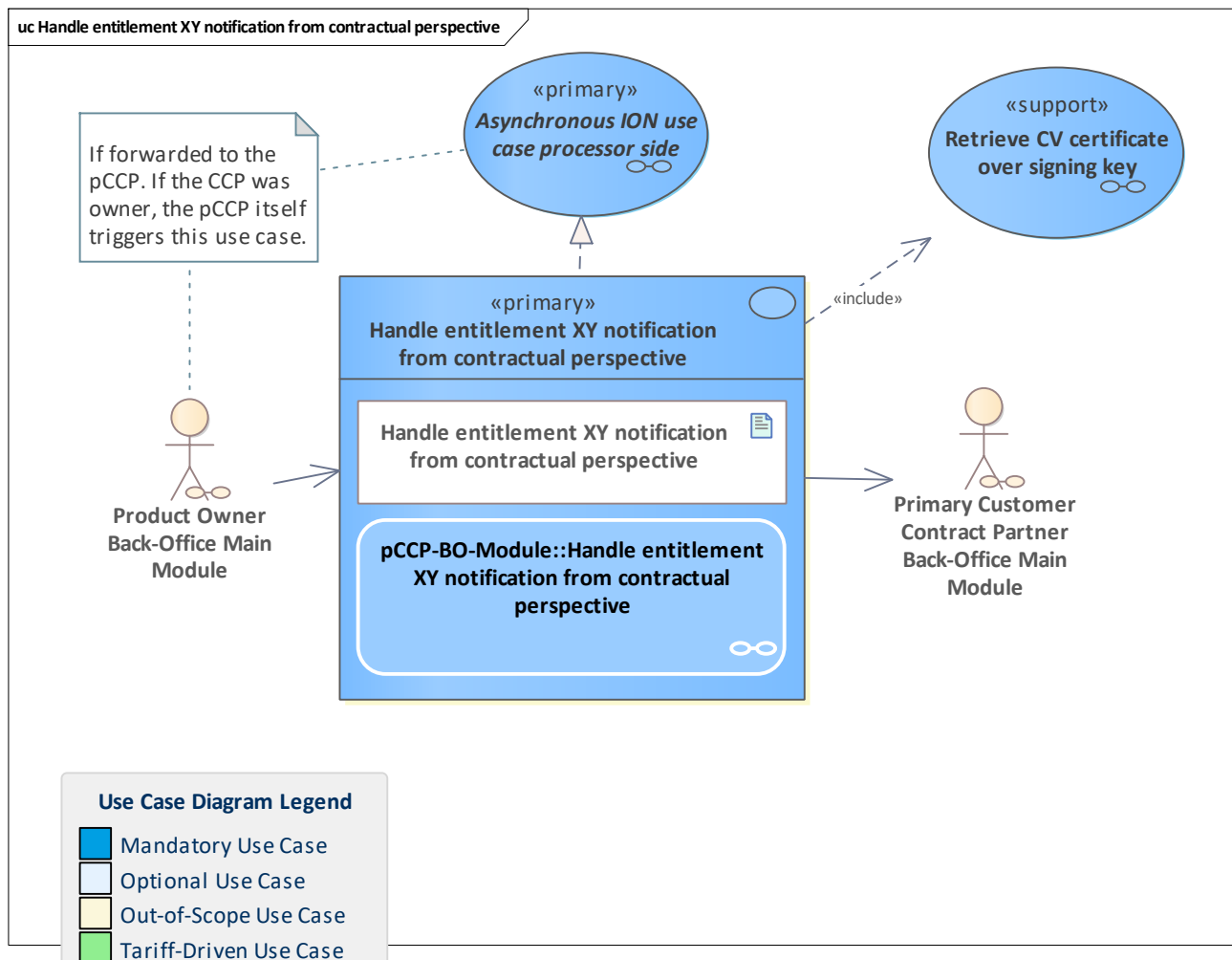


Figure 37: Handle entitlement XY notification from contractual perspective

### 3.8.1 Handle entitlement XY notification from contractual perspective

A CCP system processes a notification about an action executed on an owned entitlement. The processing is done from the contractual perspective focusing on the entitlement lifecycle and payment aspects.

This use case has two entry points:

- [pCCP-BO-Module::Handle entitlement XY notification from contractual perspective](#)  
This entry point is used in the non-owned scenarios
- [pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification](#)  
This entry point is used in the owned scenarios

The first includes the latter, in which most of the processing is done.

### 3.8.2 pCCP-BO-Module::Handle entitlement XY notification from contractual perspective

See [Handle entitlement XY notification from contractual perspective](#)



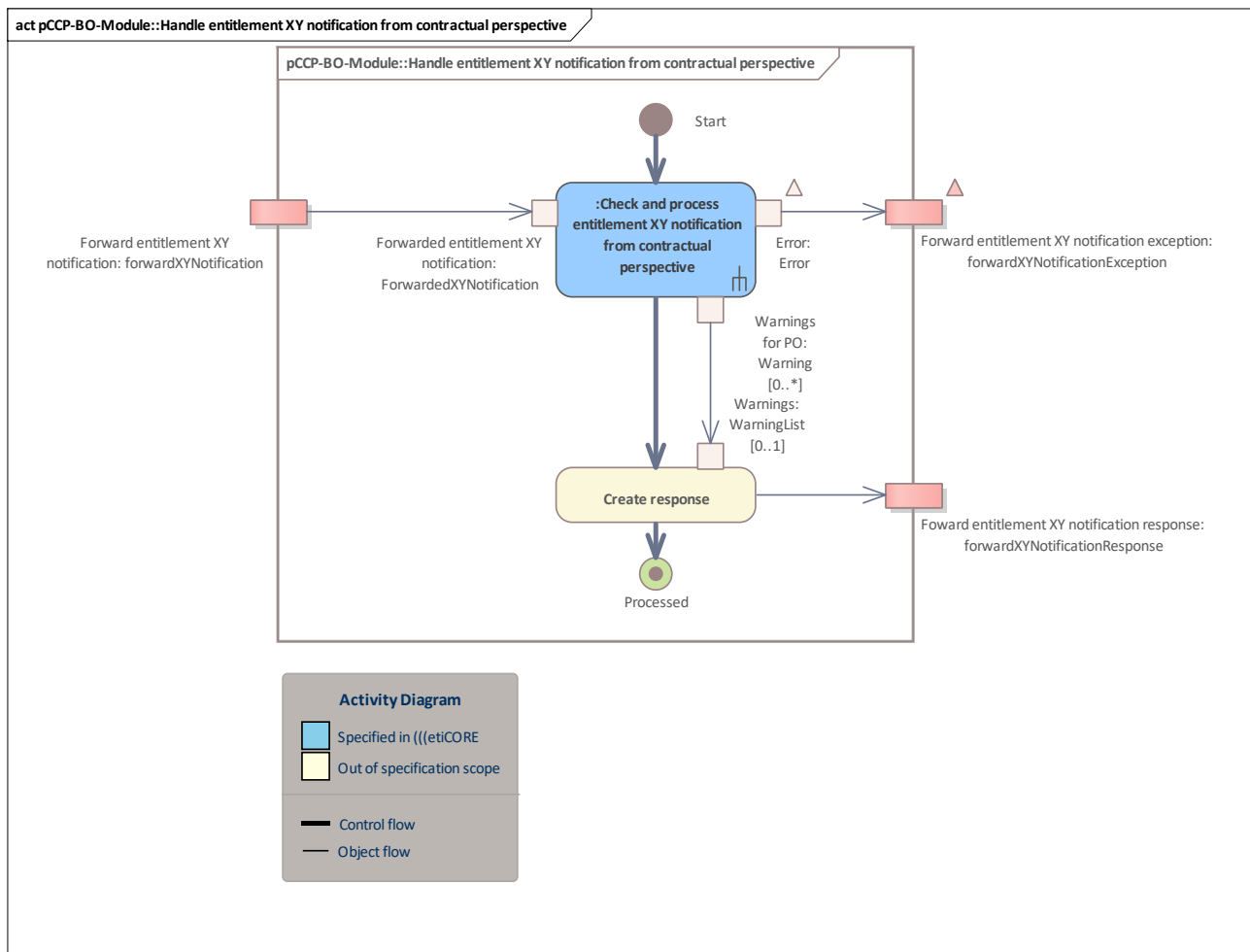


Figure 38: pCCP-BO-Module::Handle entitlement XY notification from contractual perspective

### 3.8.3 Check and process entitlement XY notification from contractual perspective

This activity performs the system internal processing of an entitlement-based notification either forwarded by the PO system or triggered directly after [Handle entitlement XY notification from operational perspective](#), if the current actor is the owner of the entitlement . All necessary monitoring checks are performed, divided into general checks and notification specific checks.

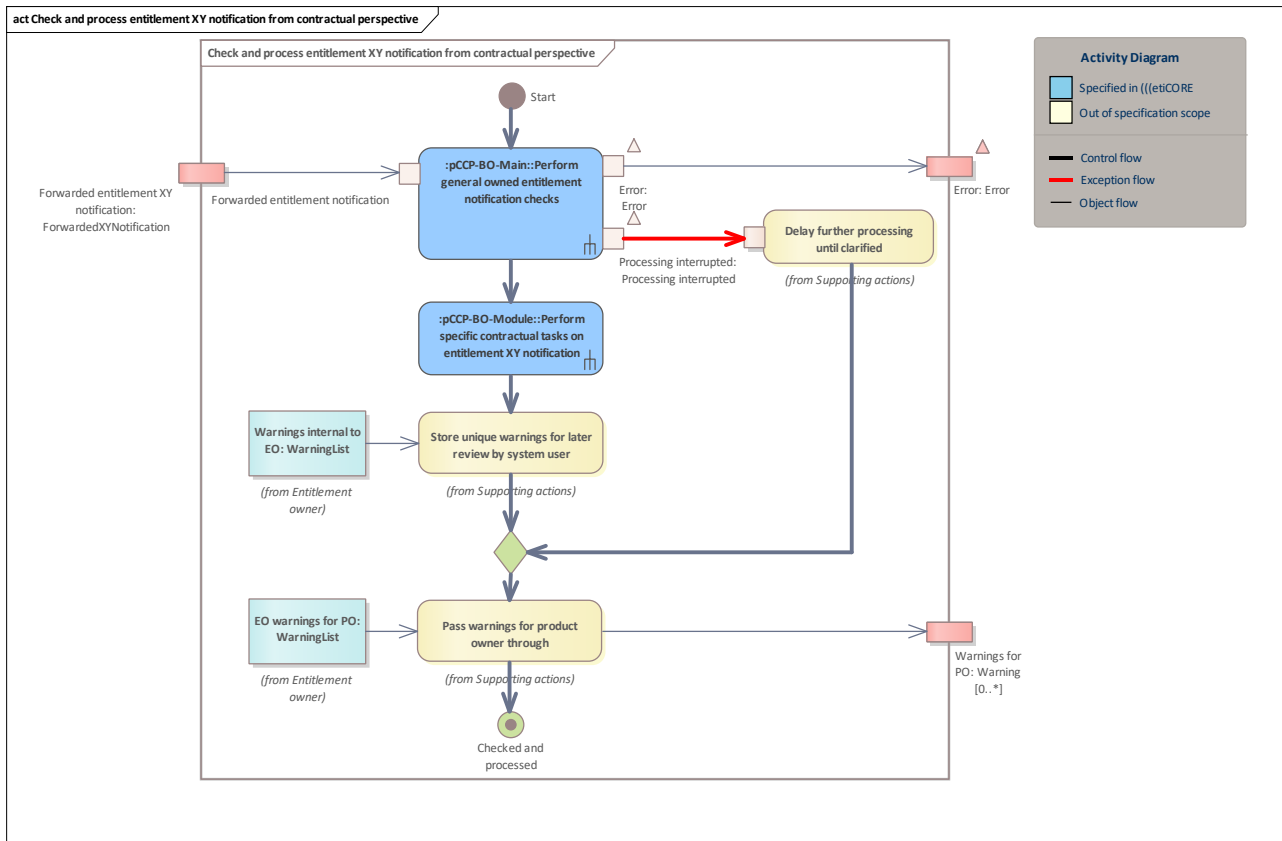


Figure 39: Check and process entitlement XY notification from contractual perspective

### 3.8.4 pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification

See [Handle entitlement XY notification from contractual perspective](#)

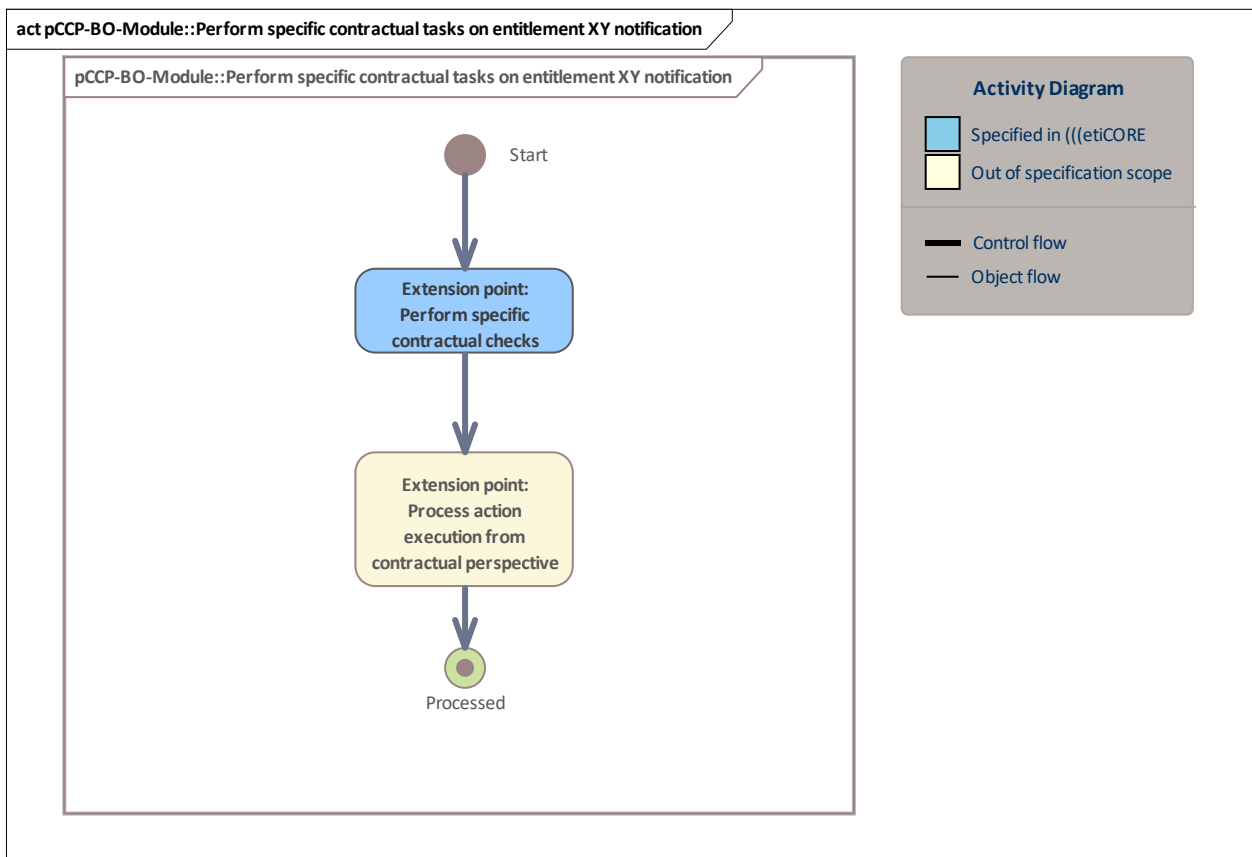


Figure 40: pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification

### 3.8.4.1 Extension point: Process action execution from contractual perspective

Update sales register, update entitlement state, etc.

## 3.9 Handle application XY notification from contractual perspective

See [Handle application XY notification from contractual perspective](#)

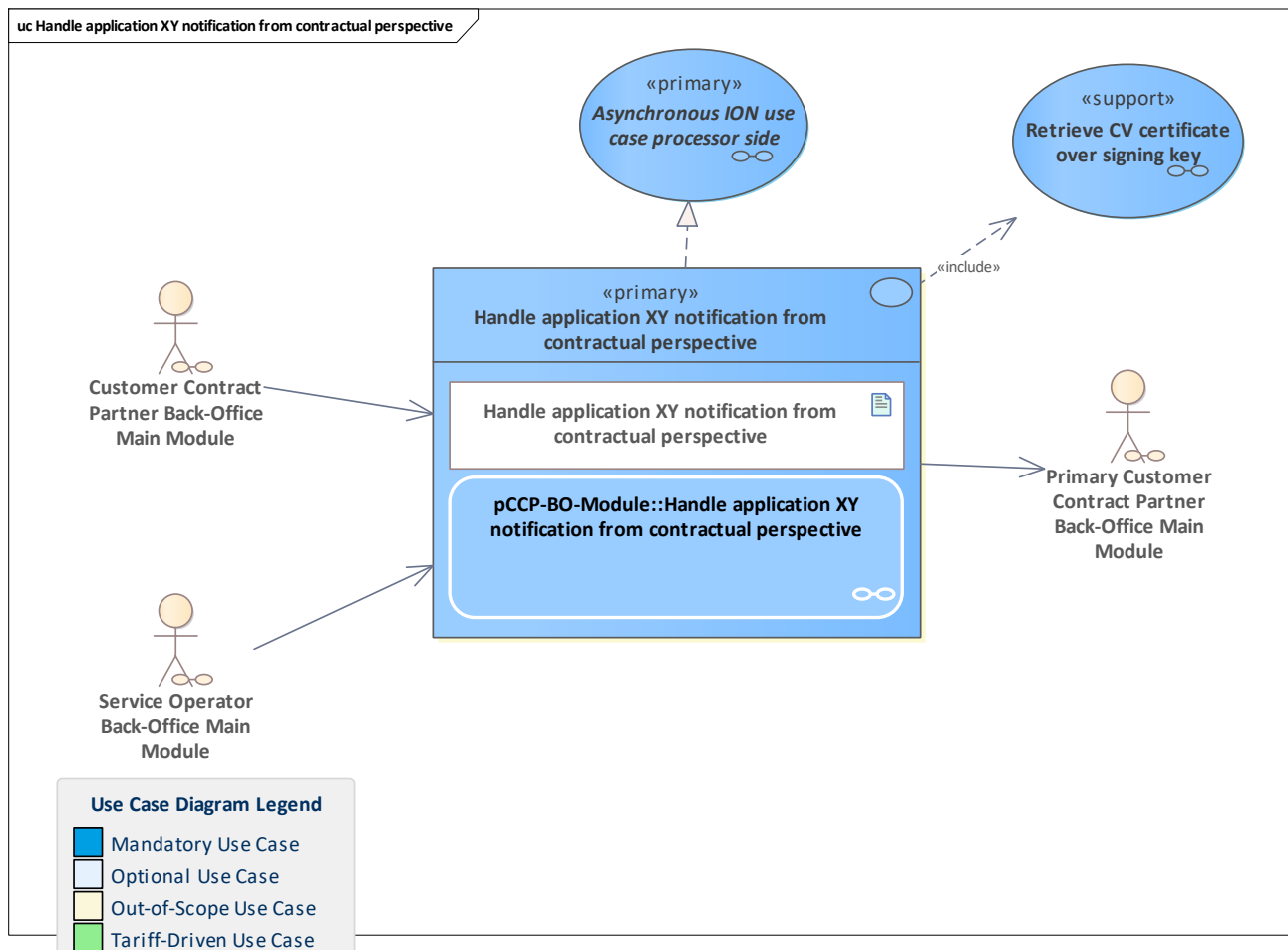


Figure 41: Handle application XY notification from contractual perspective

### 3.9.1 Handle application XY notification from contractual perspective

A CCP system processes a notification about an action executed on an owned application. The processing is done from the contractual perspective focusing on the application lifecycle aspects.

This use case has two entry points:

- [pCCP-BO-Module::Handle application XY notification from contractual perspective](#)  
This entry point is used in the non-owned scenarios
- [pCCP-BO-Module::Perform specific contractual tasks on application XY notification](#)  
This entry point is used in the owned scenarios

The first includes the latter, in which most of the processing is done.

### 3.9.2 pCCP-BO-Module::Handle application XY notification from contractual perspective

See [Handle application XY notification from contractual perspective](#)

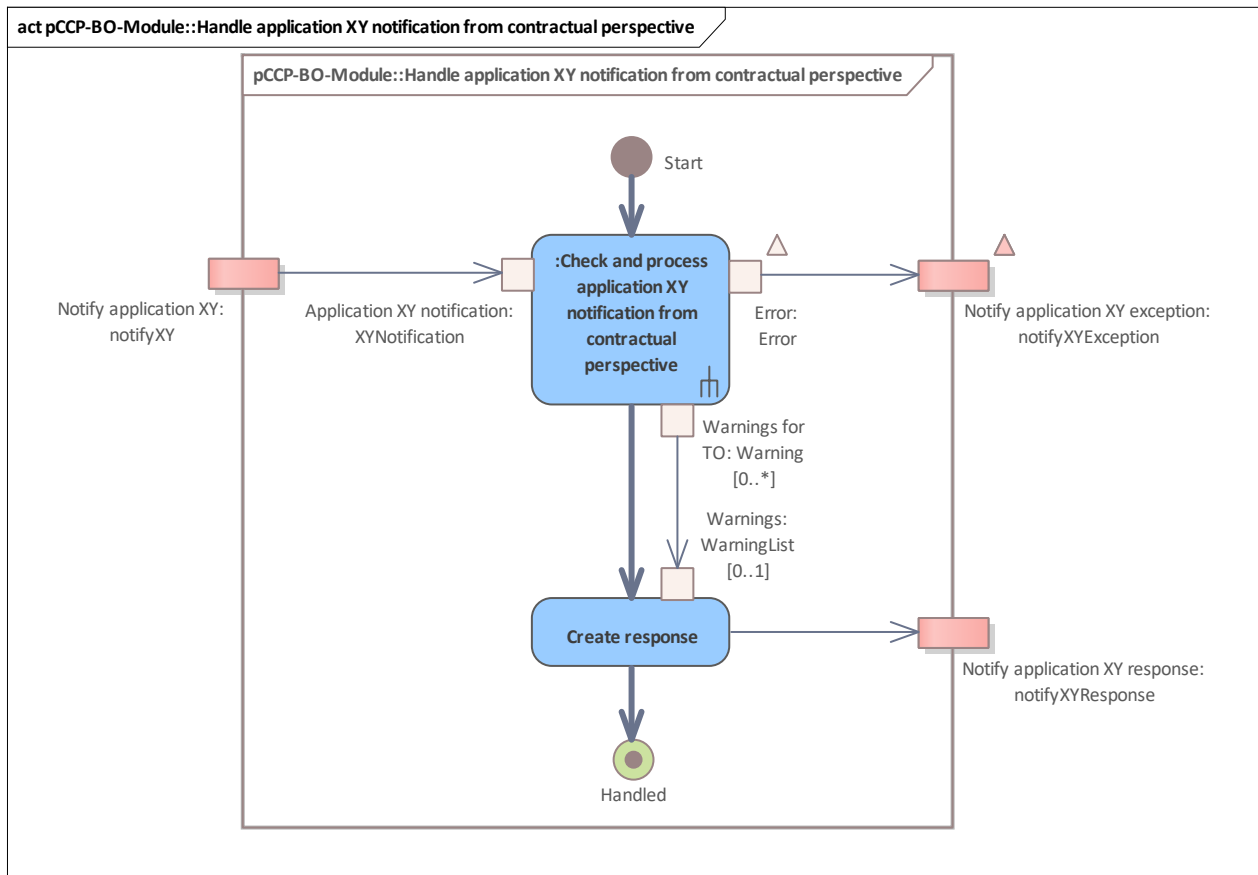


Figure 42: pCCP-BO-Module::Handle application XY notification from contractual perspective

### 3.9.3 Check and process application XY notification from contractual perspective

This activity performs the system internal processing of an application based notification either sent from another terminal operator system (SO or sCCP) or triggered directly after the executed use case [Handle application XY notification from operational perspective](#), if the current actor is the owner of the application instance. All necessary monitoring checks are performed, divided into general checks and notification specific checks.

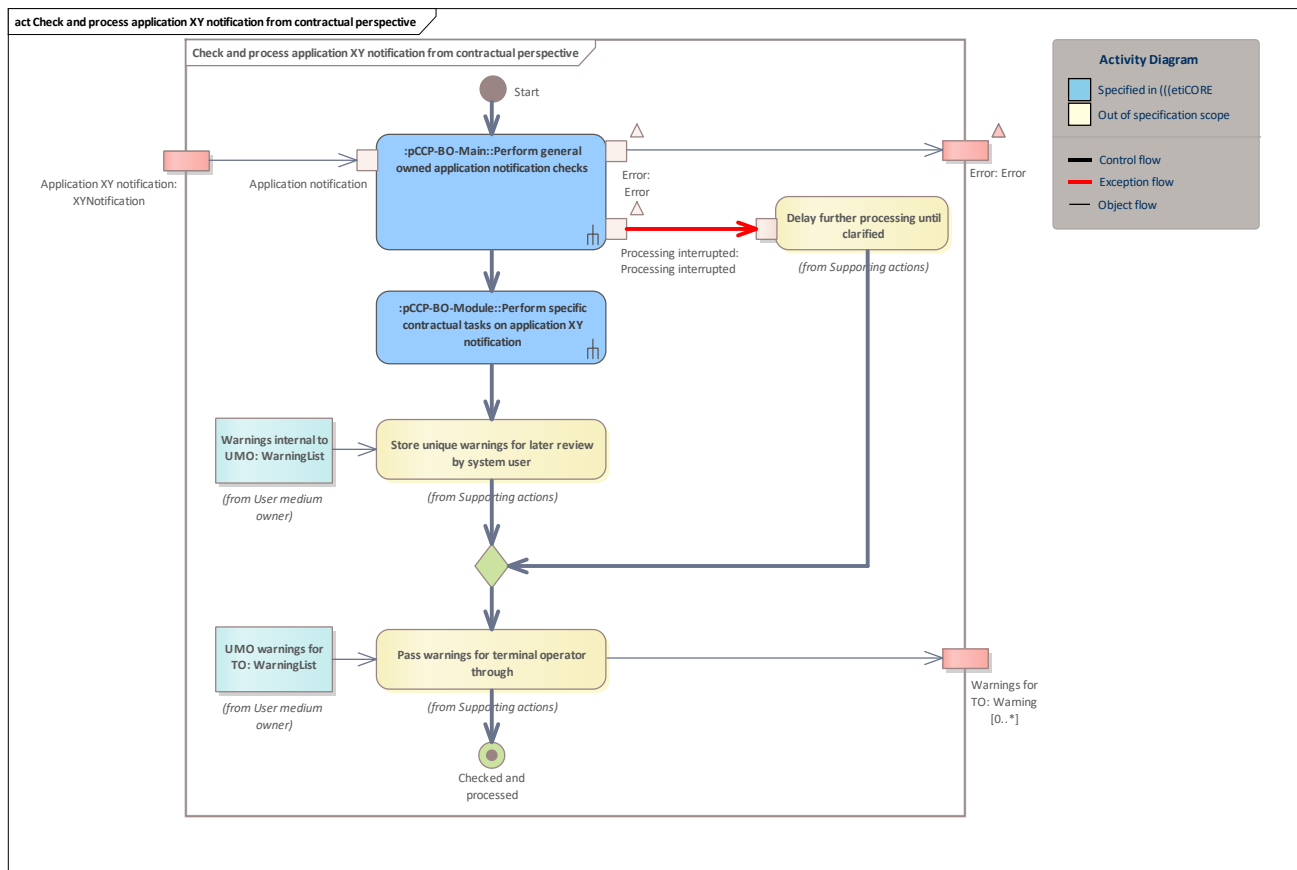


Figure 43: Check and process application XY notification from contractual perspective

### 3.9.4 pCCP-BO-Module::Perform specific contractual tasks on application XY notification

See [Handle application XY notification from contractual perspective](#)

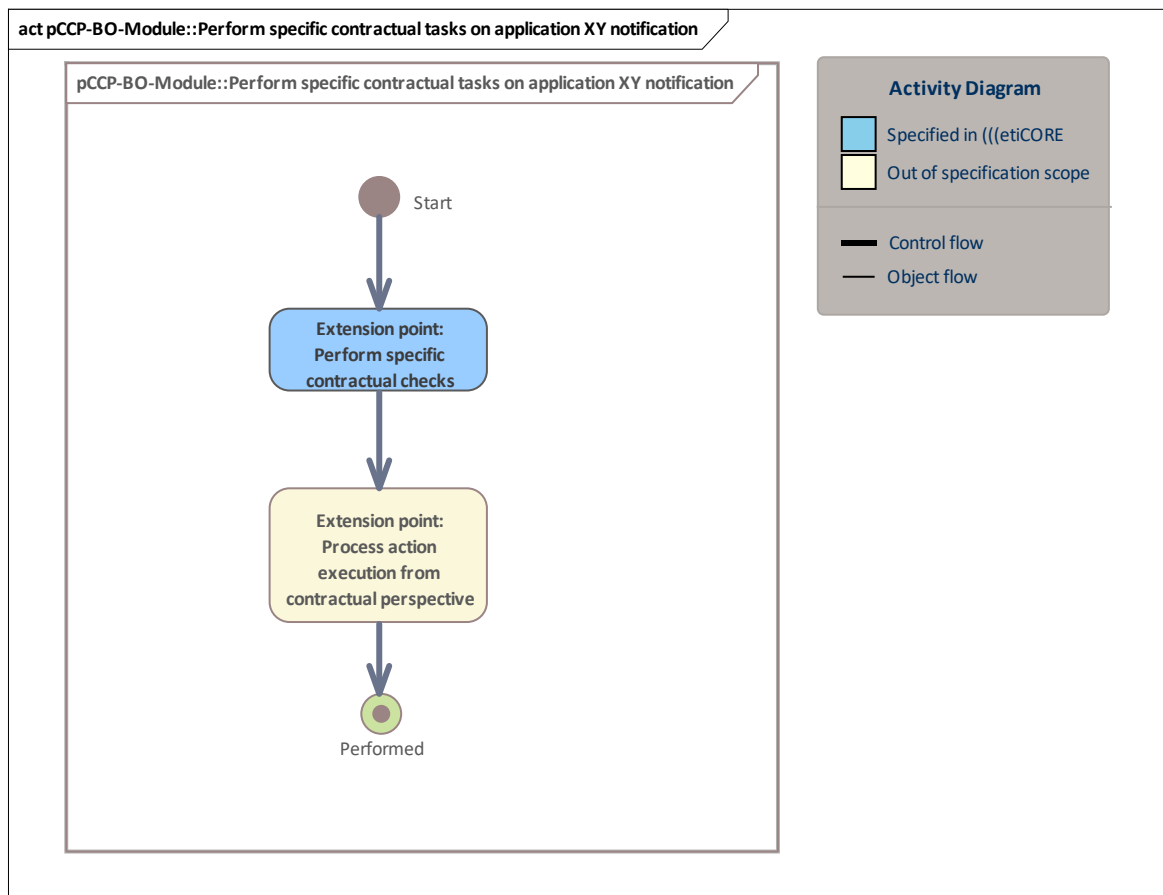


Figure 44: pCCP-BO-Module::Perform specific contractual tasks on application XY notification

## 4 Verifying Lists Updated via Increments

Incremental lists are used for large lists containing thousands of entries. These lists occur in the hotlist service for lists of hotlisted application instances and entitlements, as well as in the action list service when fetching the newest action list.

Instead of fetching a full list, an incremental list can be used. This list only contains the differences between a specified last list and the current one.

These differences have to be merged into the current inventory. To verify that the updated inventory is equal to the inventory of a full list, a certain checksum algorithm is used.

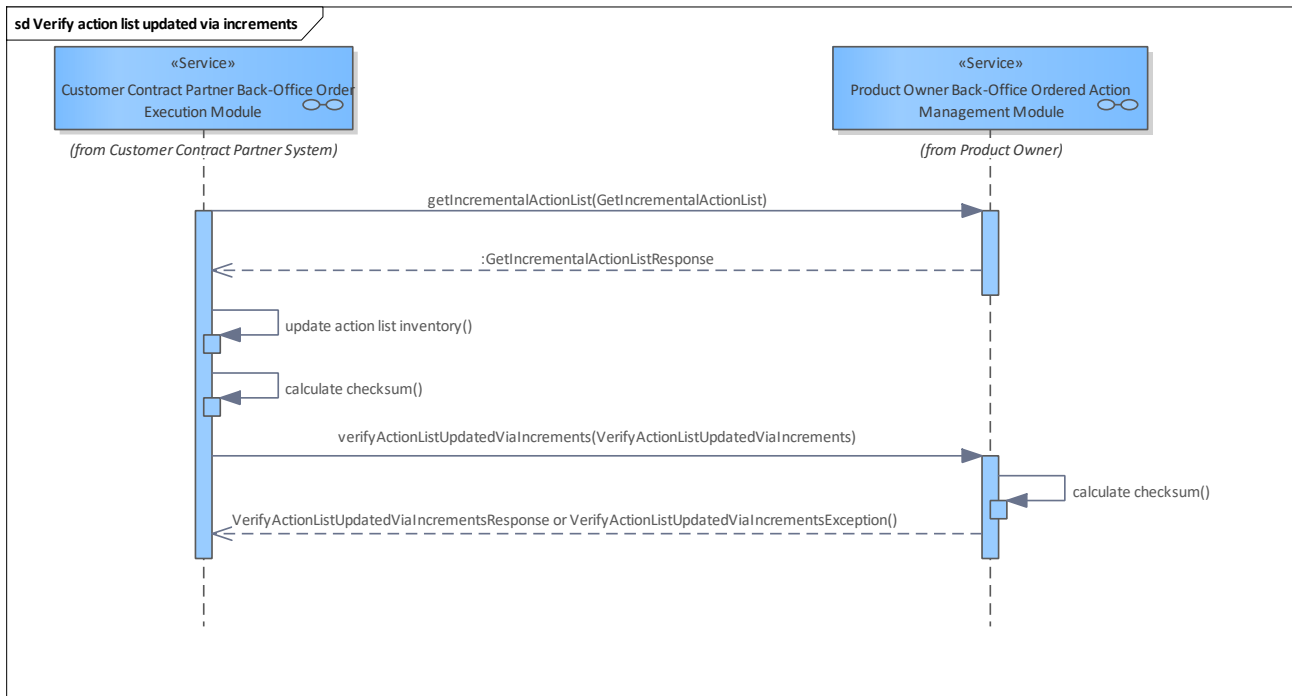


Figure 45: Verify action list updated via increments

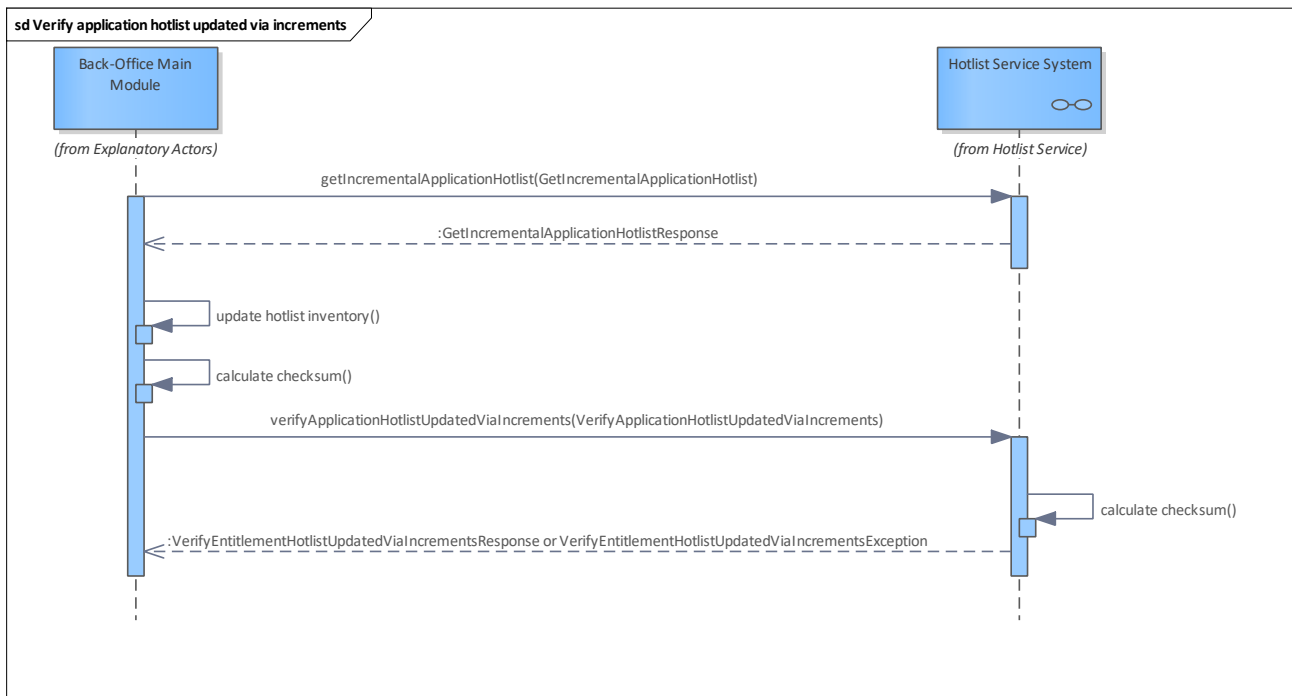


Figure 46: Verify application hotlist updated via increments





Figure 47: Verify entitlement hotlist updated via increments

## 4.1 Checksum calculation for hotlist and action list verification

The actual calculation of the checksum differs per list and is given in the corresponding use cases. Here, we describe some aspects that apply to all of them.

The data objects are encoded according to the distinguished encoding rules (DER) per the corresponding ASN.1 schemata including tag and length.

The hash algorithm to use is SHA-256.

We always sort lexicographically in ascending order.

Pseudo code to calculate the hash value:

```
hash( concatenate( sort( [ serialise(entry) for entry in entries ] ) ) )
```

## 4.2 Example calculation for an action list inventory

Action list inventory (reduced to the parts relevant to hashing) to verify:

```
[
  {
    "entitlementUnblockingActionListEntry": {
      "actionListEntryBaseInformation": {
        "orderId": {
          "orderNumber": 108,
          "orderingCcp": 123
        }
      }
    }
  },
  ...
]
```

```
{
  "entitlementIssuanceActionListEntry": {
    "actionListEntryBaseInformation": {
      "orderId": {
        "orderNumber": 109,
        "orderingCcp": 123
      }
    }
  },
  "entitlementTerminationActionListEntry": {
    "actionListEntryBaseInformation": {
      "orderId": {
        "orderNumber": 111,
        "orderingCcp": 123
      }
    }
  },
  "entitlementBlockingActionListEntry": {
    "actionListEntryBaseInformation": {
      "orderId": {
        "orderNumber": 110,
        "orderingCcp": 123
      }
    }
  }
}
```

Intermediate result after serialisation:

```
[0x4D017B5A016C,
0x4D017B5A016D,
0x4D017B5A016F,
0x4D017B5A016E]
```

Intermediate result after sorting:

```
[0x4D017B5A016C,
0x4D017B5A016D,
0x4D017B5A016E,
0x4D017B5A016F]
```

Final result after applying SHA-256:

```
0x1376FF1481CB9B73DDADDC5F4588478A506EE4129AA345B2D408E37DF1FB51E8
```

## 4.3 Example calculation for an application hotlist inventory

Application hotlist inventory to verify:

```
[
  {
    "applicationInstanceId": {
      "registrationAuthorityId": 10004,
```

```
        "subjectRole": "0x1F",
        "subjectNumber": "0x00000004"
    },
    "applicationTransitionCounter": 0,
    "applicationBlockingMode": 10
},
{
    "applicationInstanceId": {
        "registrationAuthorityId": 10004,
        "subjectRole": "0x1B",
        "subjectNumber": "0x00000002"
    },
    "applicationTransitionCounter": 6,
    "applicationBlockingMode": 0
},
{
    "applicationInstanceId": {
        "registrationAuthorityId": 10004,
        "subjectRole": "0x1B",
        "subjectNumber": "0x00000001"
    },
    "applicationTransitionCounter": 0,
    "applicationBlockingMode": 0
}
]
```

Intermediate result after serialisation:

```
[0x640D4D02271480011F82040000000458010002010A,
0x640D4D02271480011B820400000002580106020100,
0x640C4D02271480011B8203000001580100020100]
```

Intermediate result after sorting:

```
[0x640C4D02271480011B8203000001580100020100,
0x640D4D02271480011B820400000002580106020100,
0x640D4D02271480011F82040000000458010002010A]
```

Final result after applying SHA-256:

```
0x98152207B15DE600F0A98974CA510DDB6A3034E661BFE4287BF808128C61C6E2
```

## 4.4 Example calculation for an entitlement hotlist inventory

Entitlement hotlist inventory to verify:

```
[
    {
        "entitlementBlockingMode": 0,
        "entitlementId": {
            "ccpOrgId": 348,
            "entitlementIssuanceCounter": 1005,
            "samId": {
                "registrationAuthorityId": 2713,
                "subjectNumber": "0x00000002",
                "subjectRole": "0x13"
            }
        }
    },
    ...
]
```

```
"entitlementTransitionCounter": 0
},
{
  "entitlementBlockingMode": 20,
  "entitlementId": {
    "ccpOrgId": 5730,
    "entitlementIssuanceCounter": 120,
    "samId": {
      "registrationAuthorityId": 2713,
      "subjectNumber": "0x00000002",
      "subjectRole": "0x13"
    }
  }
},
"entitlementTransitionCounter": 0
},
{
  "entitlementBlockingMode": 0,
  "entitlementId": {
    "ccpOrgId": 36,
    "entitlementIssuanceCounter": 48905,
    "samId": {
      "registrationAuthorityId": 2713,
      "subjectNumber": "0x00000002",
      "subjectRole": "0x13"
    }
  }
},
"entitlementTransitionCounter": 6
}
]
```

Intermediate result after serialisation:

```
[0x6E174D02015C640D4D020A998001138204000000024B0203ED580100020100,
0x6E164D021662640D4D020A998001138204000000024B0178580100020114,
0x6E174D0124640D4D020A998001138204000000024B0300BF09580106020100]
```

Intermediate result after sorting:

```
[0x6E164D021662640D4D020A998001138204000000024B0178580100020114,
0x6E174D0124640D4D020A998001138204000000024B0300BF09580106020100,
0x6E174D02015C640D4D020A998001138204000000024B0203ED580100020100]
```

Final result after applying SHA-256:

```
0xC58276BE0B189DEAB6350B85588F80EF016153D01002B14F192B85B1FAA38124
```

## 5 Basic Bundle Back-Office System

This functionality bundle contains use cases that all back-office systems of the CCP, SO and PO must implement.

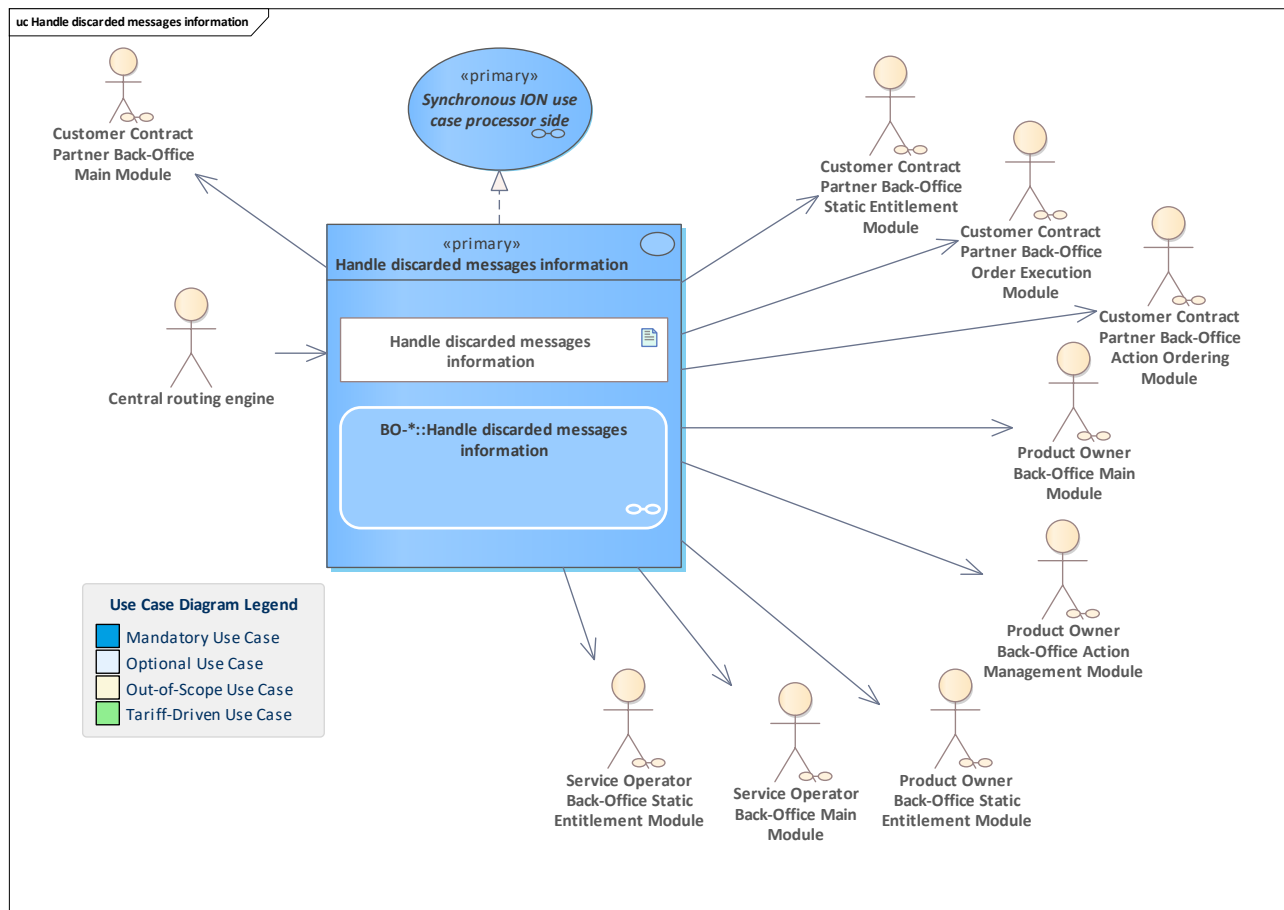
### 5.1 Overview

[Handle discarded messages information](#)  
[Set service as available for a participant](#)  
[Set service as unavailable for a participant](#)  
[Retrieve CV certificate over signing key](#)

[Retrieve and distribute the CA certificate repository](#)  
[Retrieve and distribute the CV certificate revocation list](#)  
[Notify events](#)  
[Handle events notification](#)  
[Demand application hotlisting](#)  
[Determine user medium owner](#)  
[Demand entitlement hotlisting](#)  
[Demand SAM hotlisting](#)  
[Determine SAM owner](#)  
[Revoke application hotlisting demand](#)  
[Revoke entitlement hotlisting demand](#)  
[Retrieve entitlement hotlist](#)  
[Optional: Retrieve entitlement hotlist with product information](#)  
[Optional: Retrieve incremental entitlement hotlist](#)  
[Optional: Verify entitlement hotlist updated via increments](#)  
[Update organisation hotlist inventory](#)  
[Update SAM hotlist inventory](#)  
[Retrieve and distribute organisation list](#)

## 5.2 Use Cases

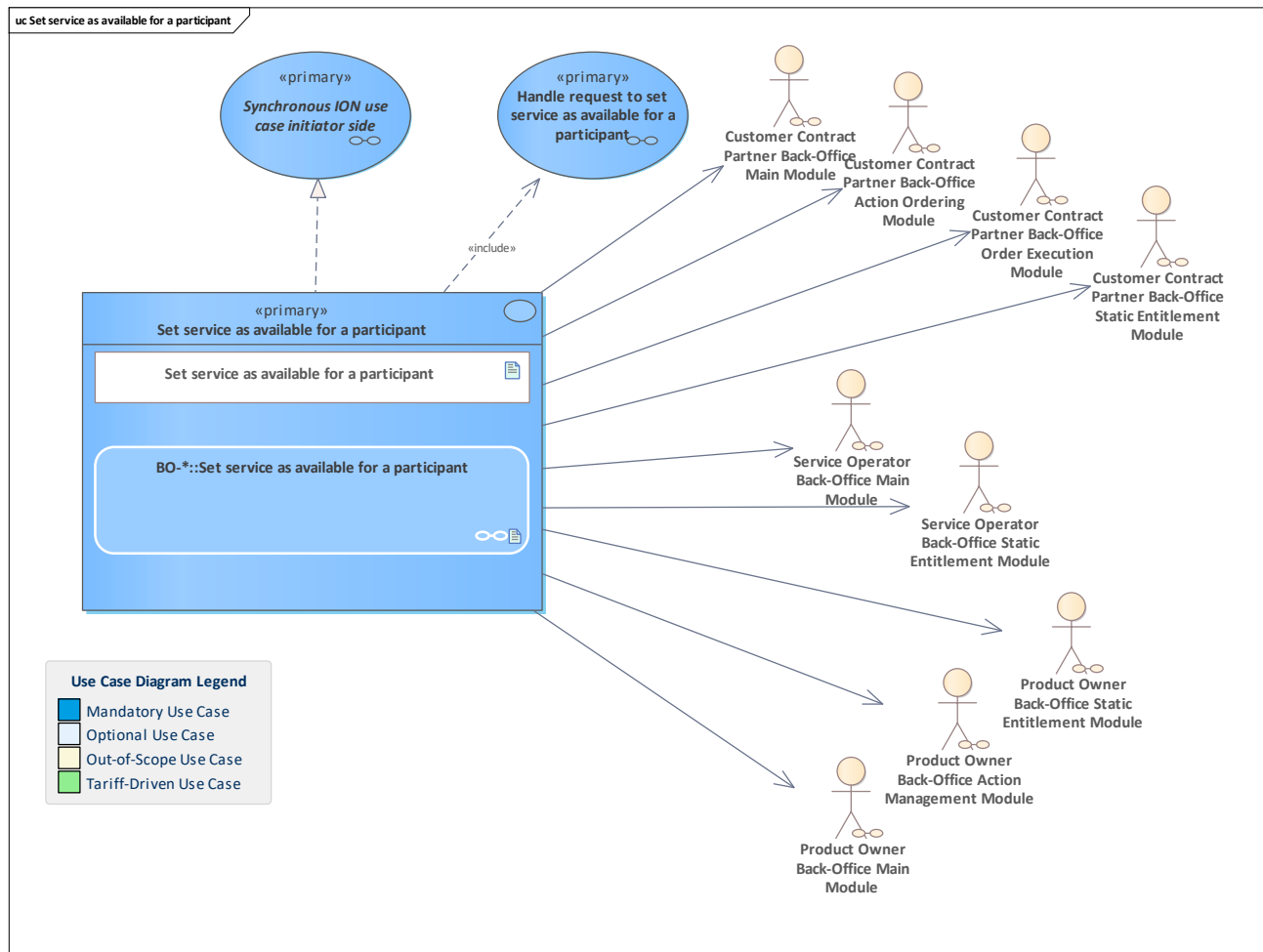
### 5.2.1 Handle discarded messages information





<b>Description</b>	<p>Process information about discarded asynchronous messages which were provided for store &amp; forward but could not be delivered to the specified recipient.</p> <p>The CRE will send message information to a</p> <ul style="list-style-type: none"><li>• <a href="#">Customer Contract Partner System</a></li><li>• <a href="#">Ordering Customer Contract Partner System</a></li><li>• <a href="#">Executing Customer Contract Partner System</a></li><li>• <a href="#">Customer Contract Partner STE System</a></li><li>• <a href="#">Service Operator System</a></li><li>• <a href="#">Service Operator STE System</a></li><li>• <a href="#">Product Owner System</a></li><li>• <a href="#">Product Owner Action Management System</a></li><li>• <a href="#">Product Owner STE System</a></li></ul> <p>These systems and modules work with asynchronous messages which can possibly be discarded.</p>
<b>Initiating Actor</b>	<a href="#">Central routing engine</a>
<b>Reacting Actor</b>	<a href="#">Initiator</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Action Management Module</a> <a href="#">Customer Contract Partner Back-Office Static Entitlement Module</a> <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> <a href="#">Product Owner Back-Office Static Entitlement Module</a> <a href="#">Service Operator Back-Office Static Entitlement Module</a> <a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">discarded messages information : notifyDiscardedMessages</a>
<b>Outputs</b>	<a href="#">response : notifyDiscardedMessagesResponse</a>
<b>Error Cases</b>	<a href="#">business exception : notifyDiscardedMessagesException</a>
<b>Activity Diagram</b>	<a href="#">BO-*::Handle discarded messages information</a>

## 5.2.2 Set service as available for a participant



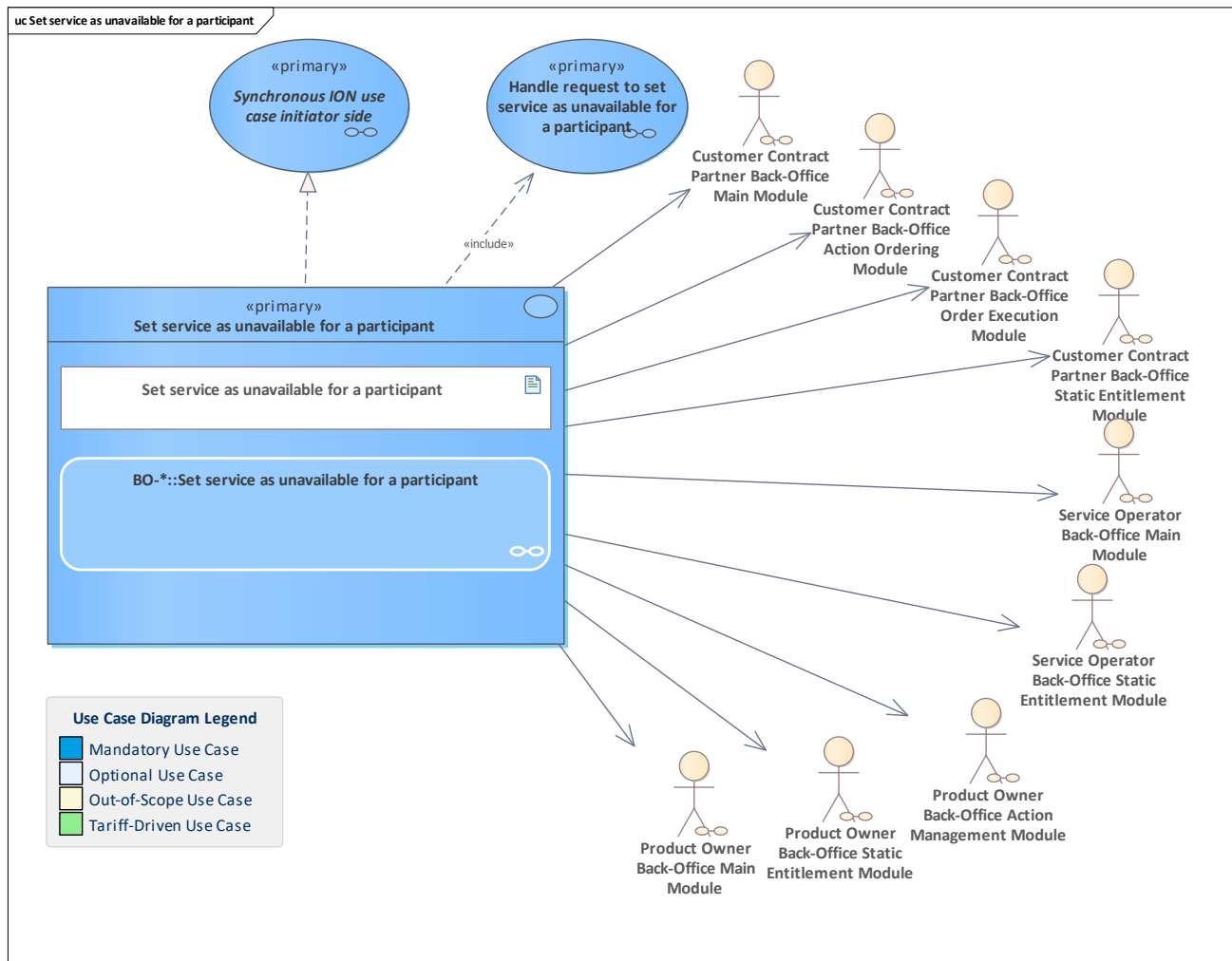
<b>Use Case</b>	<a href="#">Set service as available for a participant</a>
<b>Description</b>	In this use case, a participant sets a certain service as being available in the central routing engine (CRE). The purpose of this use case is to enable receiving messages for use cases in an asynchronous context. Note: if a service is announced to be available (again) the CRE will send stored messages to this service if any stored messages are still in the CRE's message queue.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Action Ordering Module</a> <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> <a href="#">Customer Contract Partner Back-Office Static Entitlement Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Service Operator Back-Office Static Entitlement Module</a> <a href="#">Product Owner Back-Office Action Management Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Static Entitlement Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle request to set service as available for a participant</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-*::Set service as available for a participant</a>



## 5.2.3 Set service as unavailable for a participant

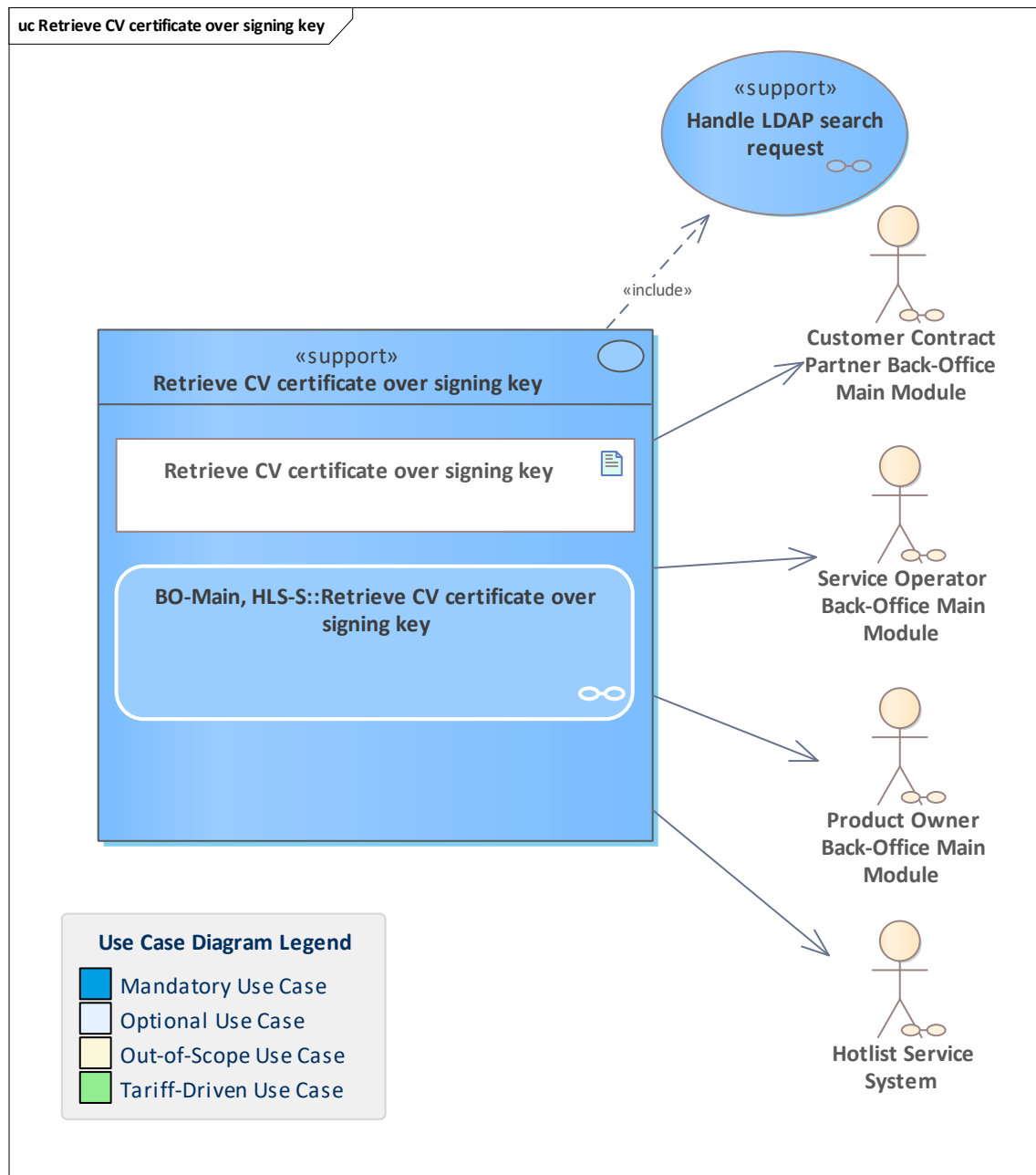


<b>Use Case</b>	<u>Set service as unavailable for a participant</u>
<b>Description</b>	<p>In this use case, a participant sets a certain service as being unavailable in the central routing engine (CRE).</p> <p>The purpose of this use case is to disable receiving messages for use cases in an asynchronous context. This means, that from now on, the CRE will store messages for a later delivery instead of routing them directly to the system that implements the service.</p> <p><b>Note:</b> participants must configure their services via the ESH in a preparatory step.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u> <u>Customer Contract Partner Back-Office Action Ordering Module</u> <u>Customer Contract Partner Back-Office Order Execution Module</u> <u>Customer Contract Partner Back-Office Static Entitlement Module</u> <u>Service Operator Back-Office Main Module</u> <u>Service Operator Back-Office Static Entitlement Module</u> <u>Product Owner Back-Office Static Entitlement Module</u> <u>Product Owner Back-Office Main Module</u> <u>Product Owner Back-Office Action Management Module</u>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle request to set service as unavailable for a participant</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-*::Set service as unavailable for a participant</a>

## 5.2.4 Retrieve CV certificate over signing key

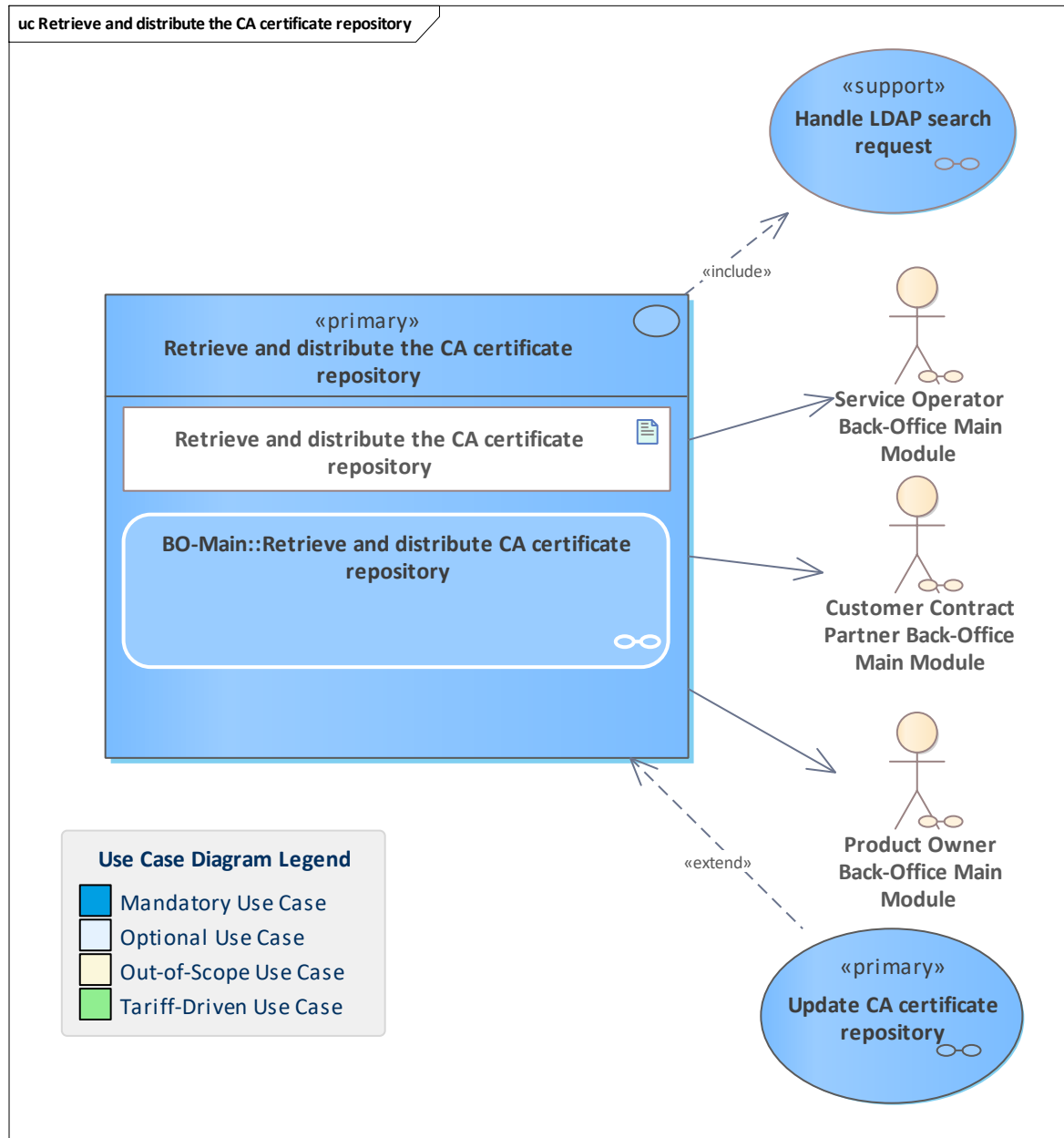


<b>Use Case</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Description</b>	Retrieve the latest certificate over the signing key of an end entity. Note that there might be several certificates for this end entity that are not relevant here: superseded certificates and certificates over keys not used for signature purposes.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Hotlist Service System</a>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">CV certificate over signing key : CvCertificate</a>
<b>Error Cases</b>	<a href="#">M-PKI not reachable</a> <a href="#">Unknown app instance ID</a>
<b>Activity Diagram</b>	<a href="#">BO-Main, HLS-S::Retrieve CV certificate over signing key</a>

## 5.2.5 Retrieve and distribute the CA certificate repository

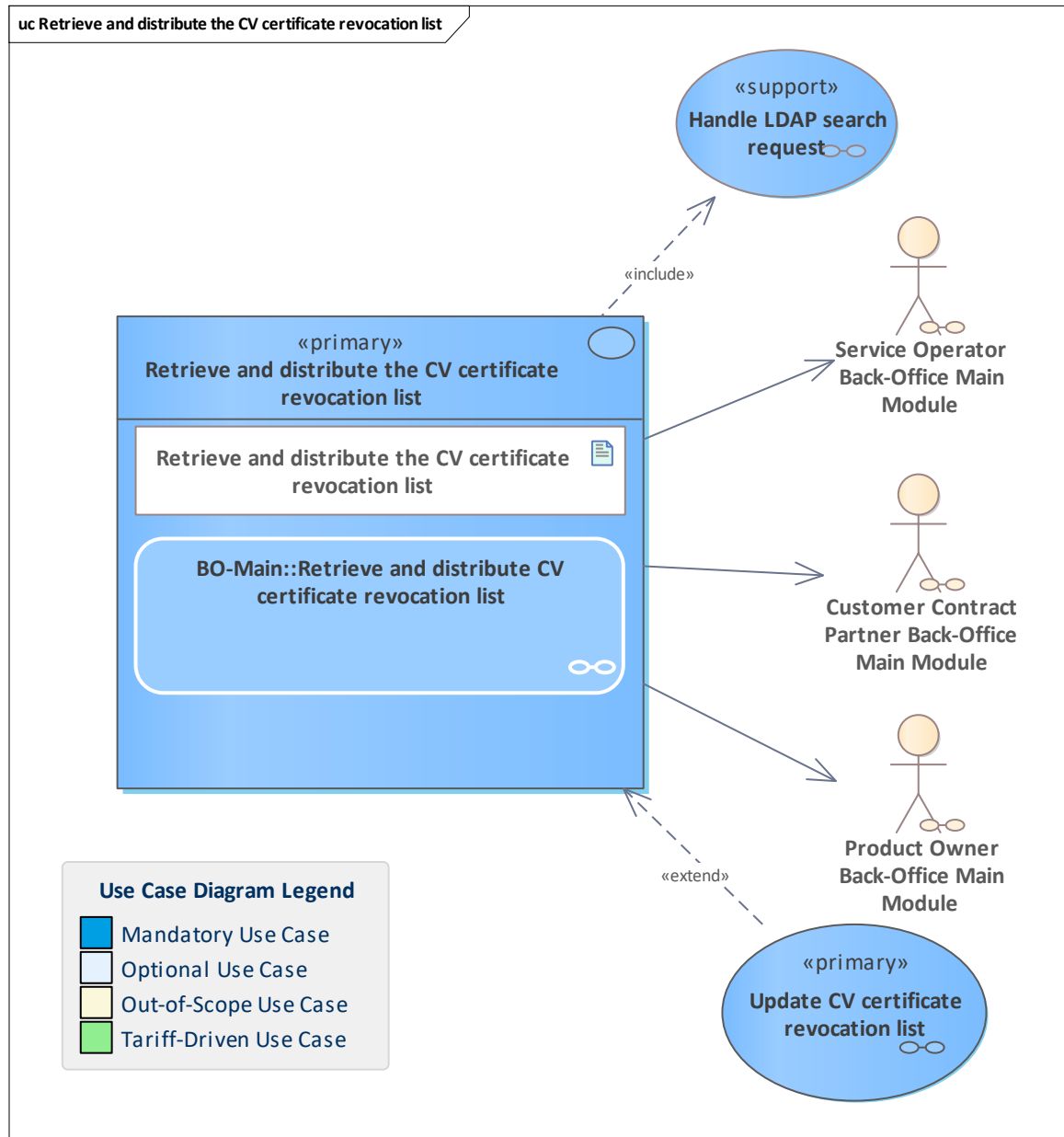


<b>Use Case</b>	<a href="#">Retrieve and distribute the CA certificate repository</a>
<b>Description</b>	The back-office system retrieves the CA certificate repository from the media-PKI (M-PKI). If the requestor operates terminals that belong to back-office this system, the CA certificate repository is also updated in all of them. This process needs to run periodically to keep the CA certificate repository up to date.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>



	Product Owner Back-Office Main Module
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Update CA certificate repository</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Retrieve and distribute CA certificate repository</a>

## 5.2.6 Retrieve and distribute the CV certificate revocation list



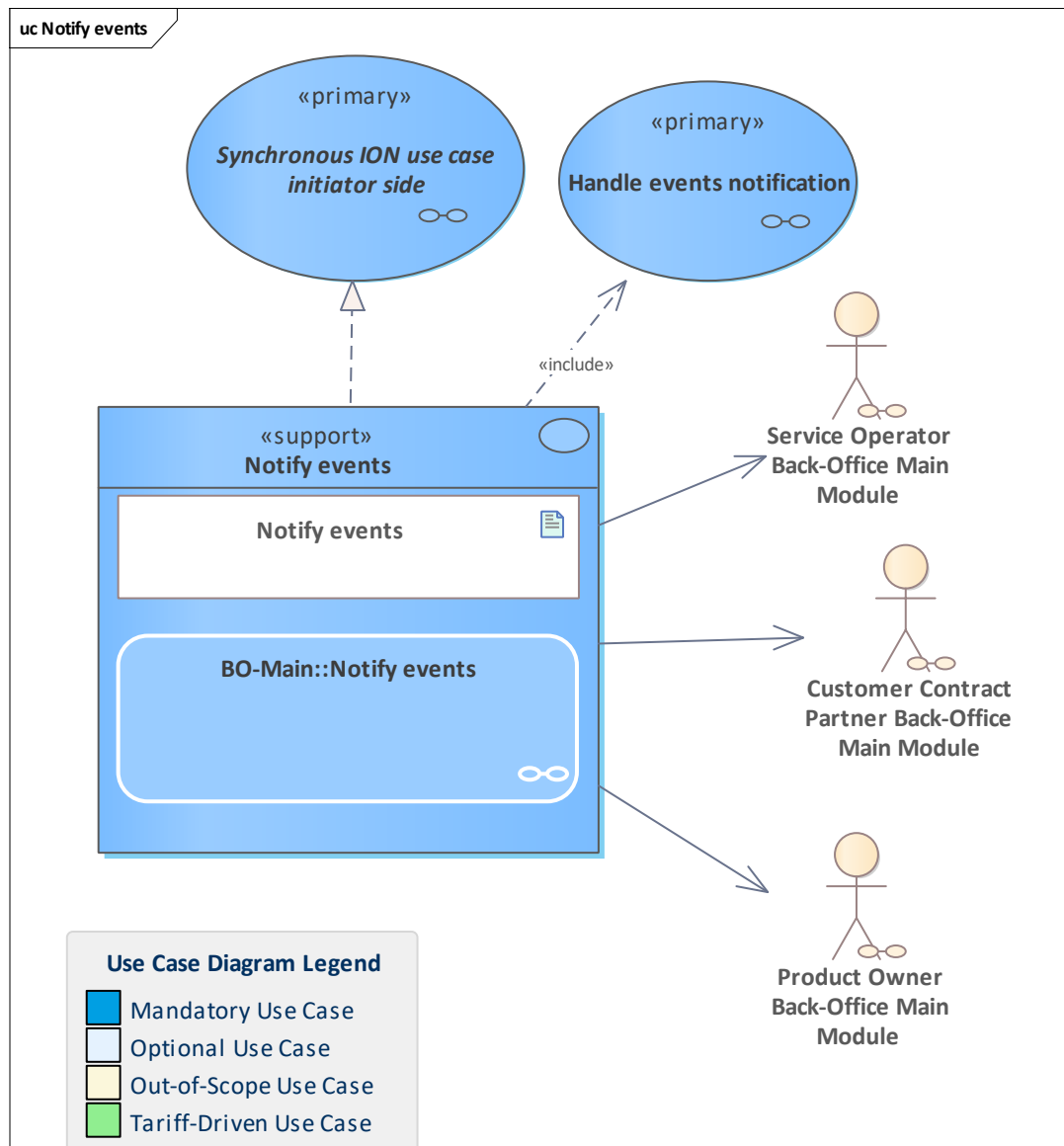
<b>Use Case</b>	<a href="#">Retrieve and distribute the CV certificate revocation list</a>
<b>Description</b>	The back-office system retrieves the CV certificate revocation list from the media-PKI (M-PKI). If the requestor operates terminals that belong to back-office this system, the CV certificate revocation list is also updated in all of them. This process needs to run periodically to keep the CV certificate revocation list up to date.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Product Owner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>



<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Update CV certificate revocation list</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Retrieve and distribute CV certificate revocation list</a>



## 5.2.7 Notify events

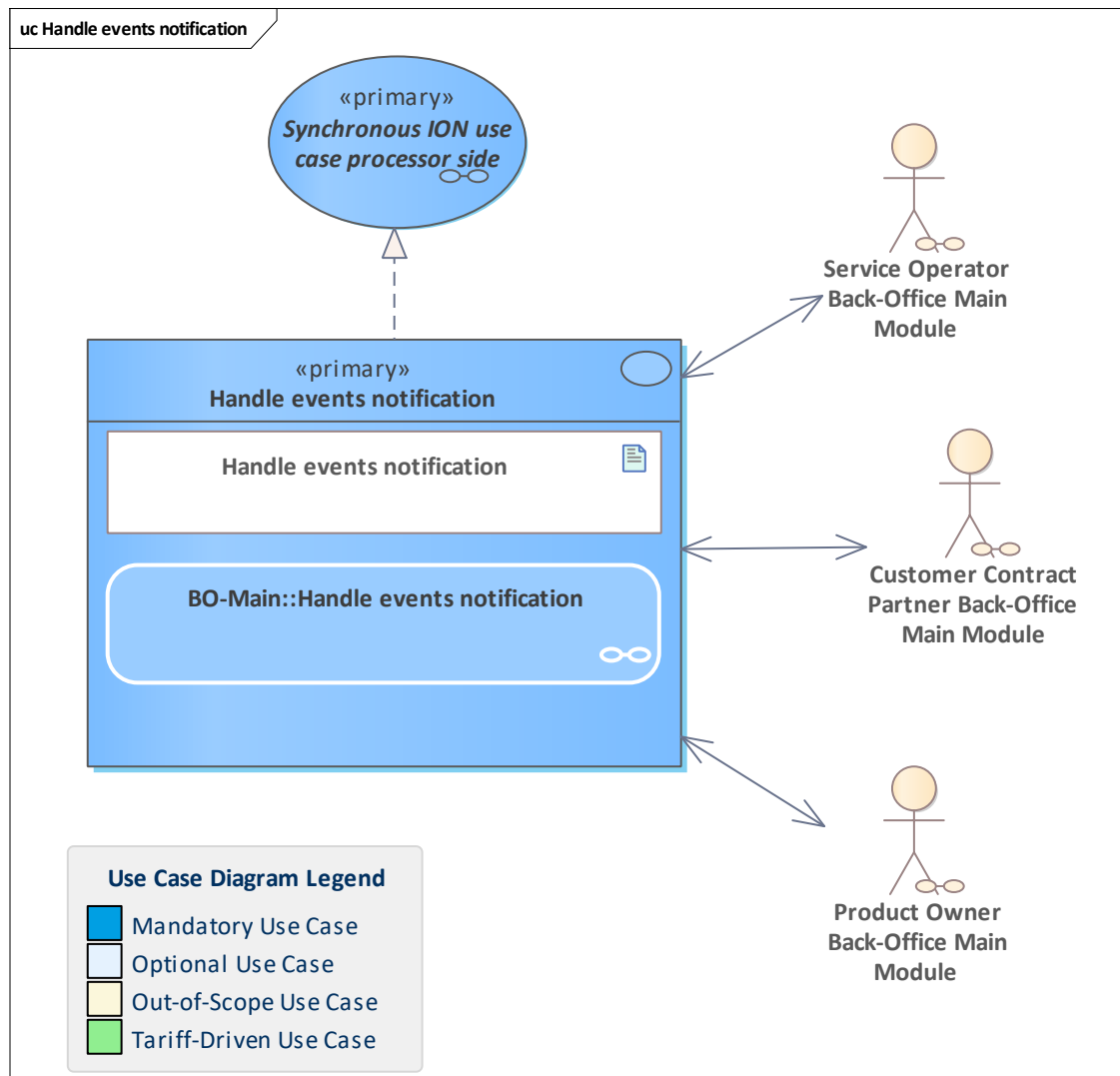


<b>Use Case</b>	<a href="#">Notify events</a>
<b>Description</b>	Notify a participant about warnings which occurred during the downstream monitoring or similar.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle events notification</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>



<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Role : PartnerRoleCode</a> <a href="#">Warnings : Warning</a> <a href="#">Receiver : OrganisationId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Notify events</a>

## 5.2.8 Handle events notification

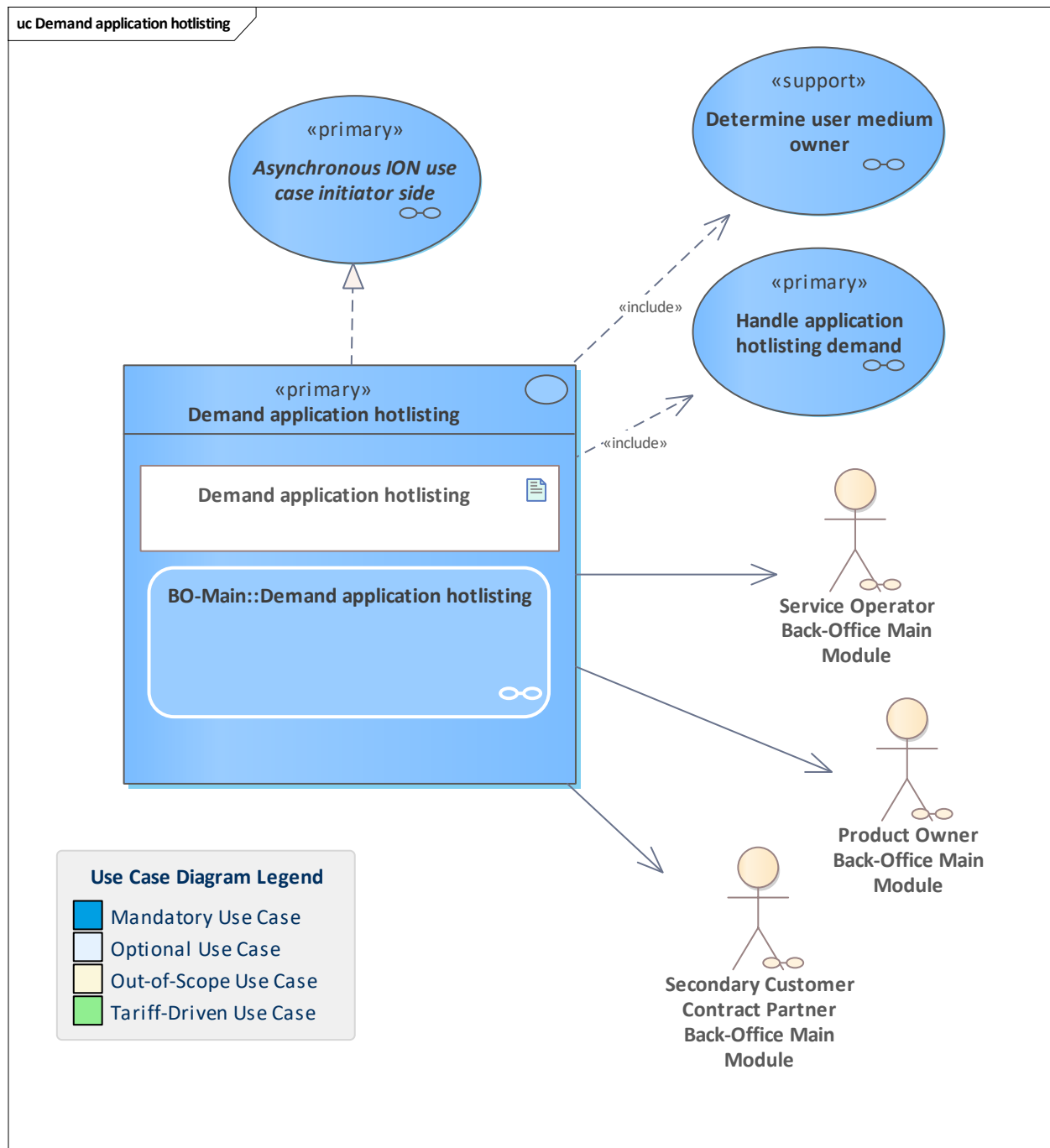


<b>Use Case</b>	<a href="#">Handle events notification</a>
<b>Description</b>	A participant is informed about warnings.
<b>Initiating Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case processor side</a>
<b>Base Activity</b>	



<b>Inputs</b>	<a href="#">Notify events : notifyEvents</a>
<b>Outputs</b>	<a href="#">Notify events response : notifyEventsResponse</a>
<b>Error Cases</b>	<a href="#">Notify events exception : notifyEventsException</a>
<b>Activity Diagram</b>	<a href="#">BO-Main::Handle events notification</a>

## 5.2.9 Demand application hotlisting

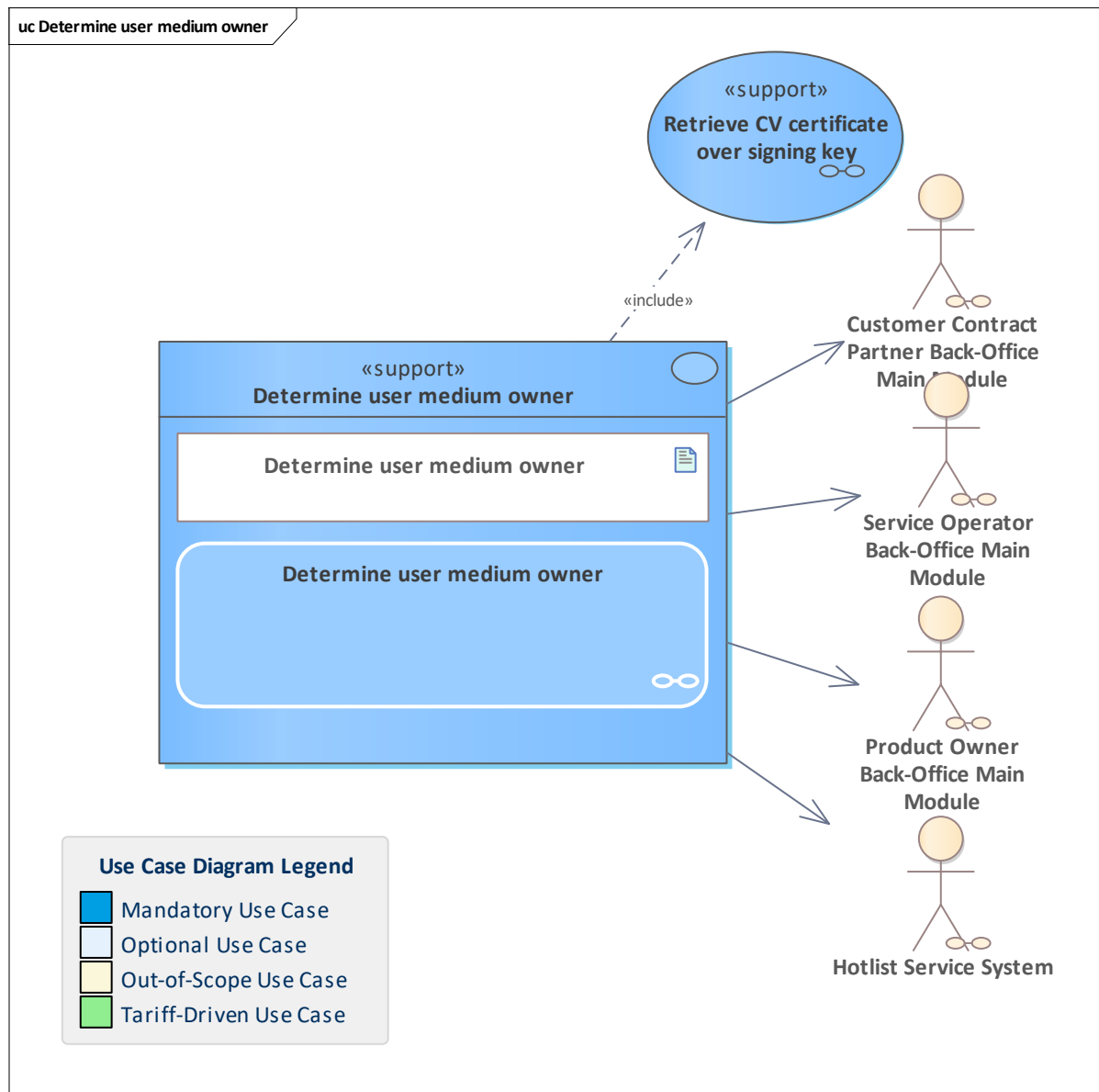


Use Case	Demand application hotlisting
Description	<p>The PO, sCCP or SO as sender demands the pCCP that issued the application to the customer to hotlist the application in question. The sender adds a reason (SO e.g. in case of a defective user medium, sCCP e.g. in case of a lost user medium) and further hotlisting parameters.</p> <p>This application can be either a user medium application with an application instance ID or a MOTICS app with an SCE ID.</p>



	In most cases, the hotlist demand from a third party will be caused by monitoring.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a> / <a href="#">Determine user medium owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Application transition counter : TransitionCounter</a> <a href="#">Blocking reason : BlockingReason</a> <a href="#">Hotlist entry expiration time : ZonedDateTime</a> <a href="#">Application blocking mode : ApplicationBlockingModeCode</a> <a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Demand application hotlisting</a>

## 5.2.10 Determine user medium owner



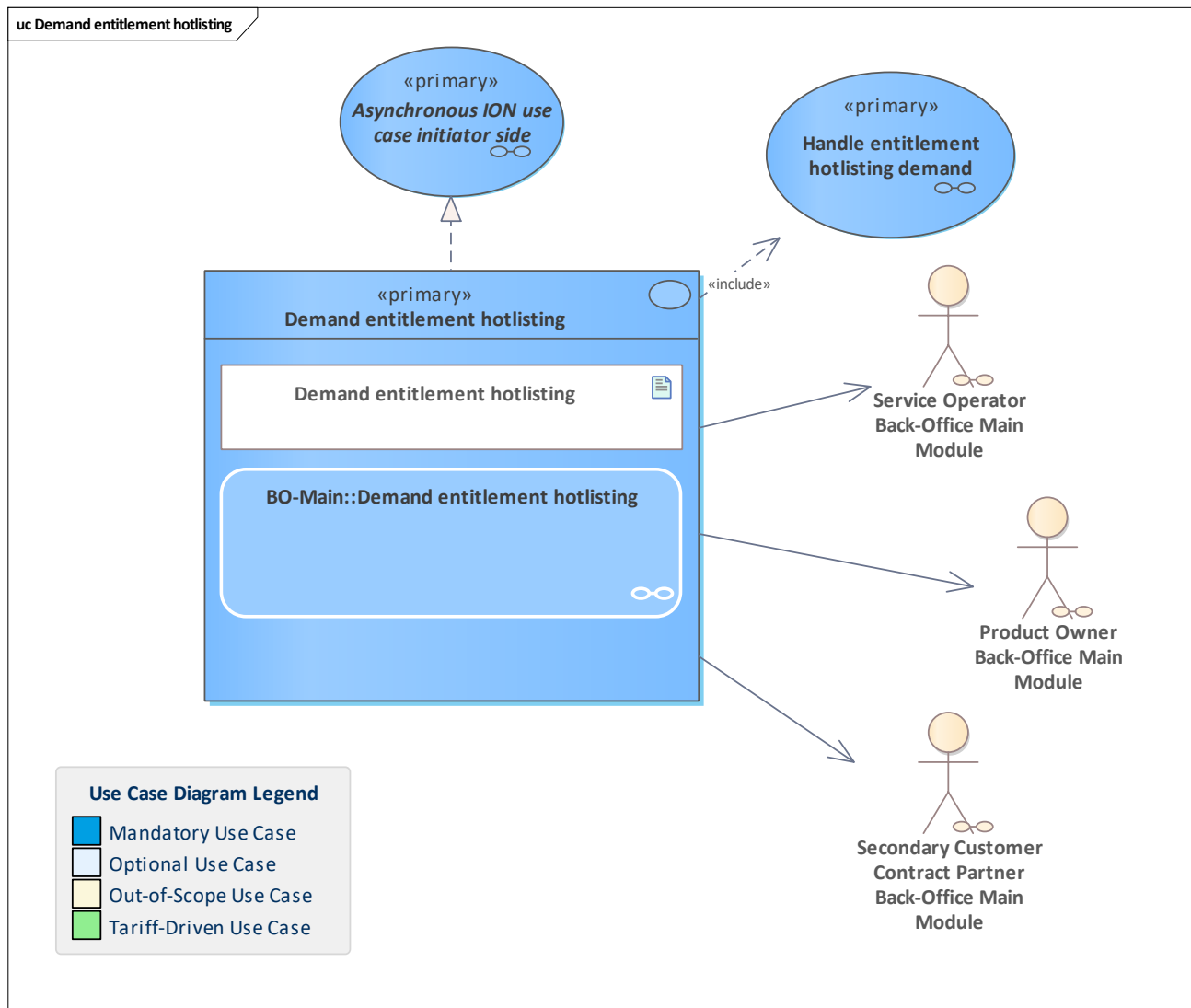
<b>Use Case</b>	<u>Determine user medium owner</u>
<b>Description</b>	<p>Determine the owner of a user medium with an application or SCE (in case of MOTICS with static entitlement) in a back-office system using the ownership information contained in the corresponding certificate.</p> <p>To determine the user medium owner, the matching CV certificate is to be fetched, so the caller retrieves the latest certificate over the signing key of an end entity.</p> <p>Note that there might be several certificates for this end entity, that are not relevant here: superseded certificates and certificates for keys not used for signature purposes. Thus, the right certificate that delivers the owner organisation ID has to be filtered.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u>



	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Hotlist Service System</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">Org ID of user medium owner : OrganisationId</a>
<b>Error Cases</b>	<a href="#">M-PKI not reachable</a> <a href="#">Unknown app instance ID</a>
<b>Activity Diagram</b>	<a href="#">Determine user medium owner</a>



## 5.2.11 Demand entitlement hotlisting

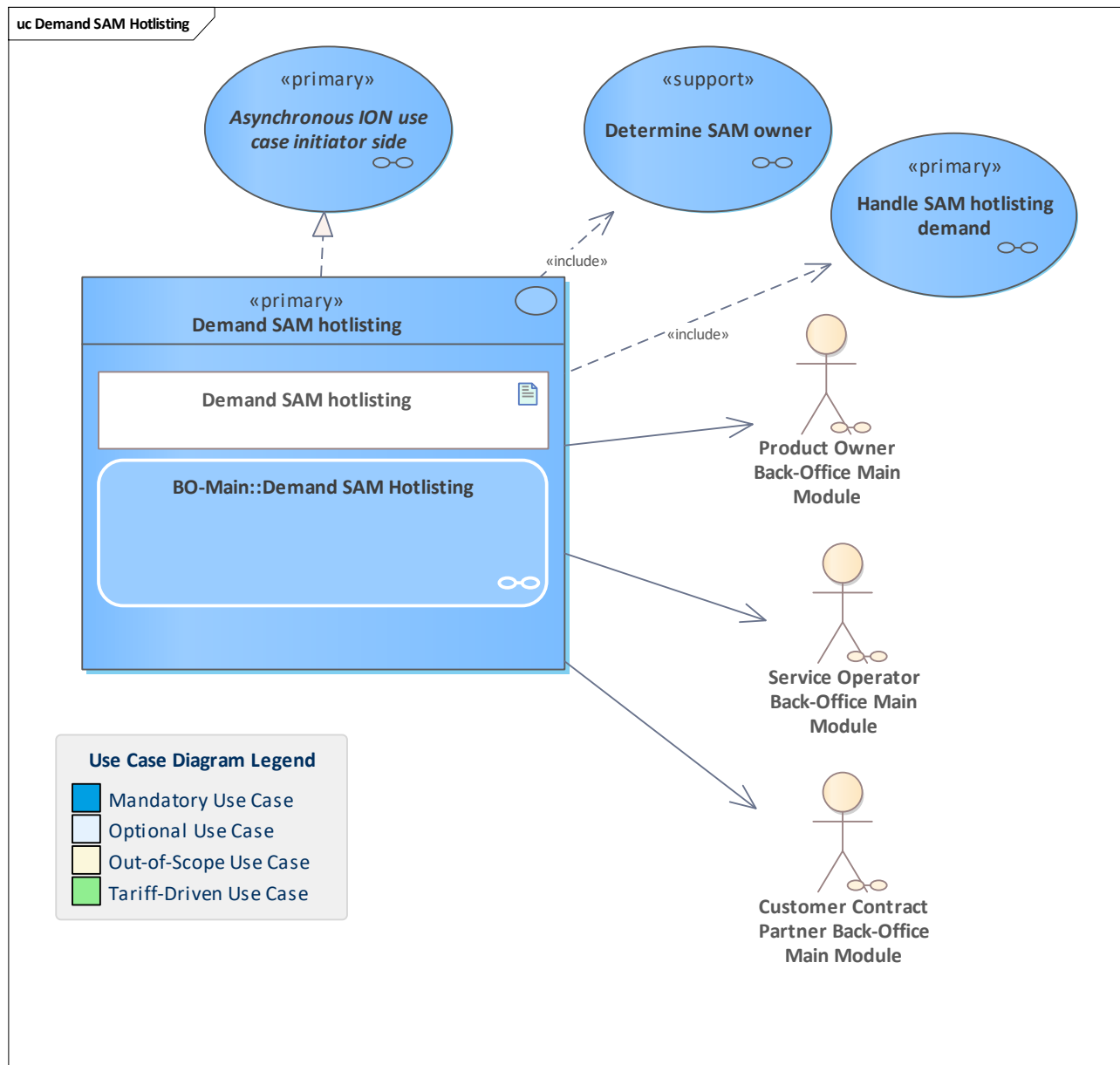


Use Case	Demand entitlement hotlisting
Description	<p>In this use case, the PO, sCCP or SO as sender demands the pCCP that issued the entitlement to the customer to hotlist the entitlement in question.</p> <p>The sender adds a reason and further hotlisting parameters. This entitlement can be either located on a user medium with an application or a static entitlement coming from a barcode or a MOTICS app.</p> <p>In most cases, the hotlisting demand from a third party will be caused by monitoring.</p> <p>Possible reasons are:</p> <ul style="list-style-type: none"> <li>Inconsistencies have occurred during monitoring (the most likely reason)</li> <li>Offence against terms of carriage</li> <li>Payment delay</li> <li>Referenced product was deactivated</li> </ul>



	<ul style="list-style-type: none"><li>Entitlement with the same ID already exists</li><li>User medium/application which contains entitlements of a secondary customer contract partner (CCP) was replaced</li></ul>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Blocking reason : BlockingReason</a> <a href="#">Hotlist entry expiration time : ZonedDateTime</a> <a href="#">Entitlement blocking mode : EntitlementBlockingModeCode</a> <a href="#">Hotlist entry effective time : ZonedDateTime</a> <a href="#">UM or SCE-ID : AppInstanceId</a> <a href="#">Entitlement transition counter : TransitionCounter</a> <a href="#">Product ID : ProductId</a> <a href="#">Entitlement ID : EntitlementId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Demand entitlement hotlisting</a>

## 5.2.12 Demand SAM hotlisting

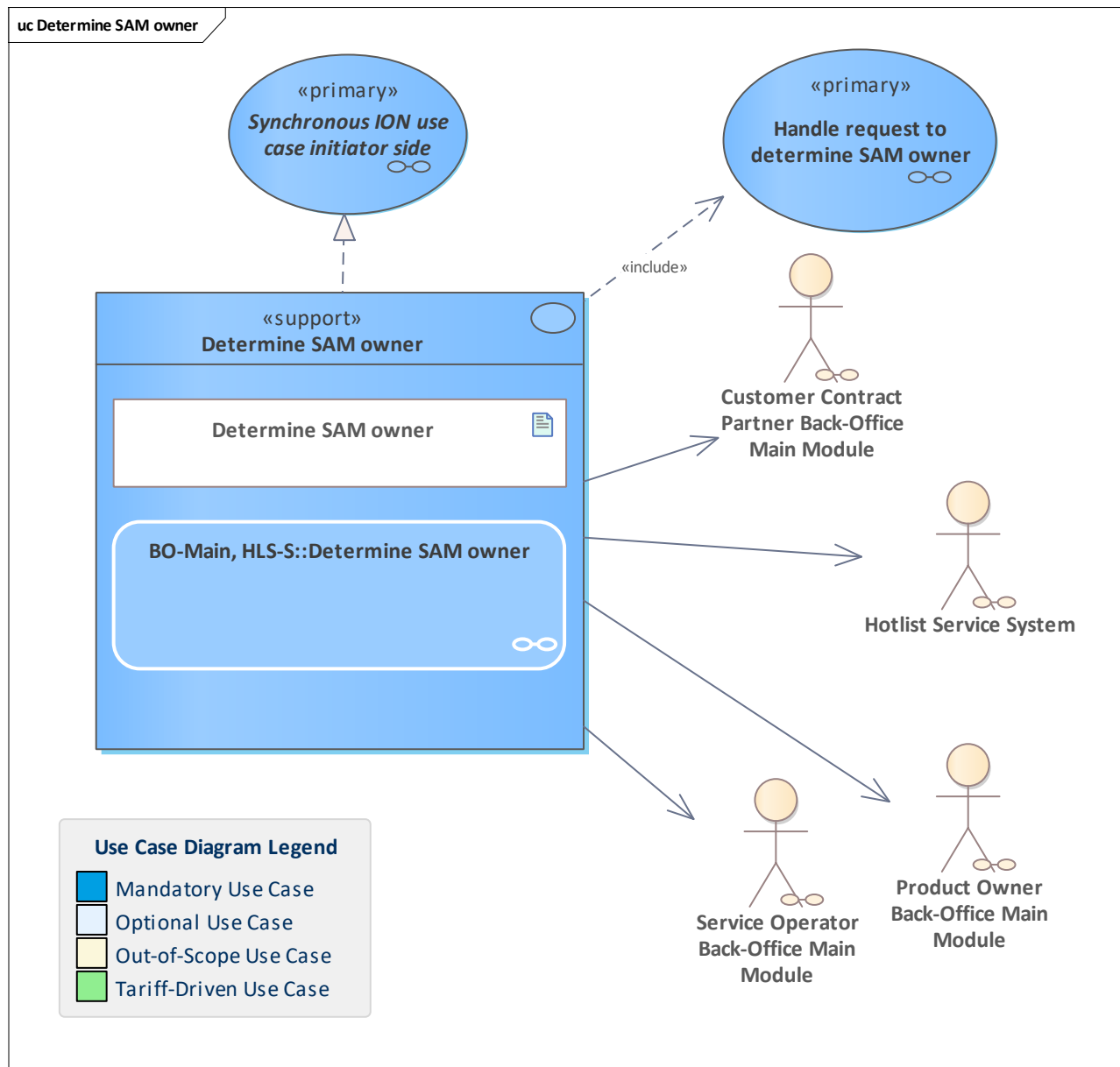


<b>Use Case</b>	<a href="#">Demand SAM hotlisting</a>
<b>Description</b>	<p>A PO, SO or CCP sends a demand for hotlisting to the SAM Owner (SO or CCP). Reasons for this demand could either be loss or theft of a SAM that was used in one of the SO or CCP terminals. Another reason could be suspected fraud, which could be detected by monitoring.</p> <p>Before demanding the SAM hotlisting, the demander must find out the SAM owner to place the demand to the right receiver.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle SAM hotlisting demand</a> / <a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Blocking reason : BlockingReason</a> <a href="#">SAM action counter : ActionCounter</a> <a href="#">SAM ID : AppInstanceId</a> <a href="#">SAM entitlement issuance counter : EntitlementIssuanceCounter</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Demand SAM Hotlisting</a>

## 5.2.13 Determine SAM owner



<b>Use Case</b>	<a href="#">Determine SAM owner</a>
<b>Description</b>	Determine the SAM Owner (organisation ID and role) for a given SAM ID using the service provided by the ESH.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Hotlist Service System</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases</b>	<a href="#">Handle request to determine SAM owner</a>



<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">SAM ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">SAM owner : OrganisationalUnit</a>
<b>Error Cases</b>	<a href="#">E_CO_APP_INSTANCE_ID_UNKNOWN</a> <a href="#">Unknown SAM : E_CO_APP_INSTANCE_ID_UNKNOWN</a> <a href="#">Timeout</a>
<b>Activity Diagram</b>	<a href="#">BO-Main, HLS-S::Determine SAM owner</a>

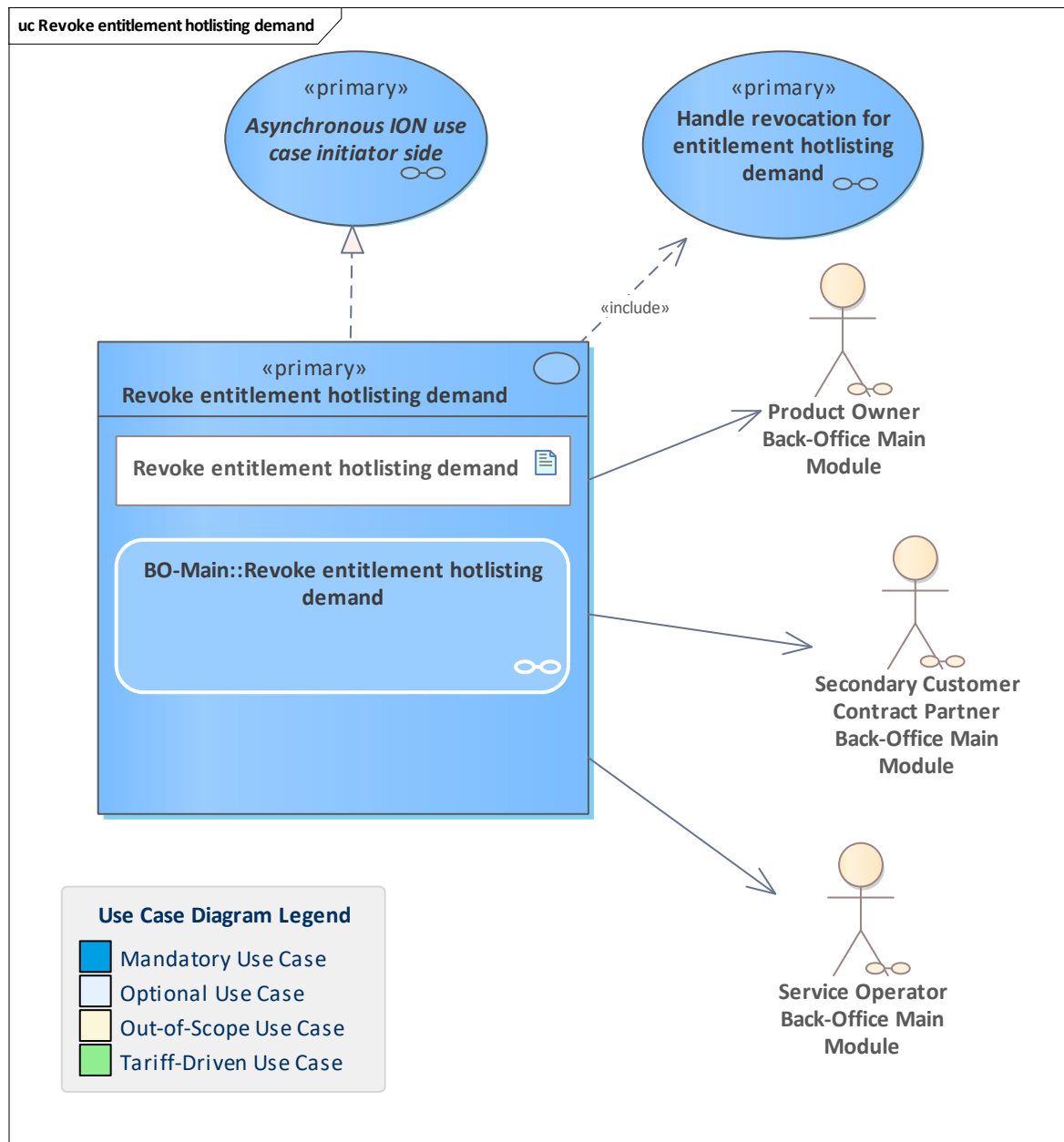




	The new application owner must have transferred the information about old hotlisting demands (if any), especially the ION message references.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle revocation for application hotlisting demand</a> / <a href="#">Determine user medium owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">ION message id : IonMessageId</a> <a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Revoke application hotlisting demand</a>



## 5.2.15 Revoke entitlement hotlisting demand

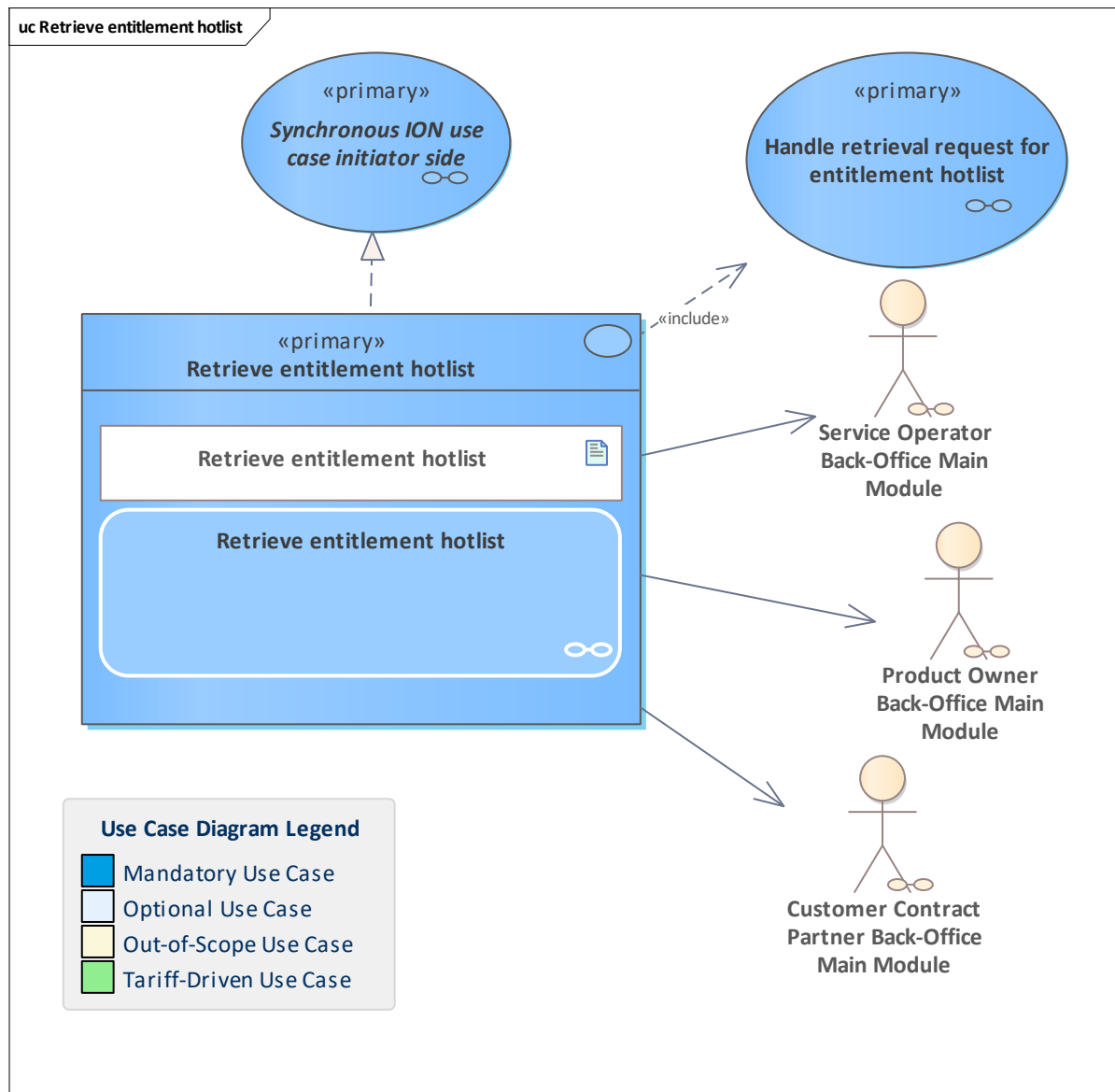


<b>Use Case</b>	<a href="#">Revoke entitlement hotlisting demand</a>
<b>Description</b>	<p>A PO, SO or sCCP sends a demand for hotlisting revocation to the entitlement owner (pCCP).</p> <p>The pCCP will check the revocation demand and, in case of a positive decision, have the entitlement removed from the hotlist. The revocation references the ION message of the previous hotlisting demand.</p> <p>This is a rarely used use case.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>



<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle revocation for entitlement hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Entitlement Id : EntitlementId</a> <a href="#">ION message ID : IonMessageId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Revoke entitlement hotlisting demand</a>

## 5.2.16 Retrieve entitlement hotlist

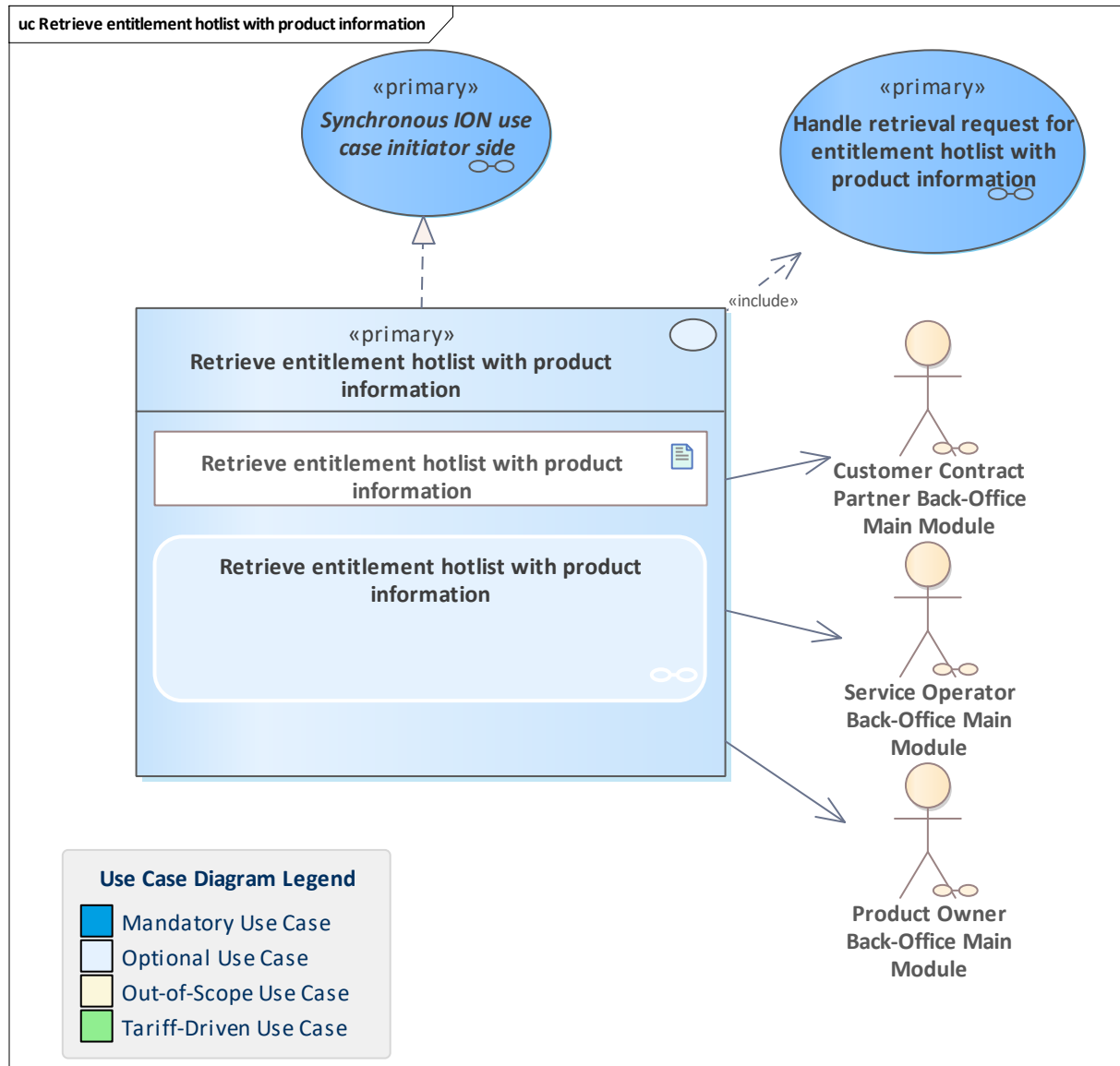


<b>Use Case</b>	<u>Retrieve entitlement hotlist</u>
<b>Description</b>	The SO, PO and CCP want to update their entitlement hotlist inventory by retrieving the current entitlement hotlist from the hotlist service system. Please note that the entitlement hotlist can be retrieved either as an incremental or a total hotlist, as well as a total hotlist with product information.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u> <u>Service Operator Back-Office Main Module</u> <u>Product Owner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	



<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for entitlement hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Entitlement hotlist : EntitlementHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve entitlement hotlist</a>

## 5.2.17 Optional: Retrieve entitlement hotlist with product information

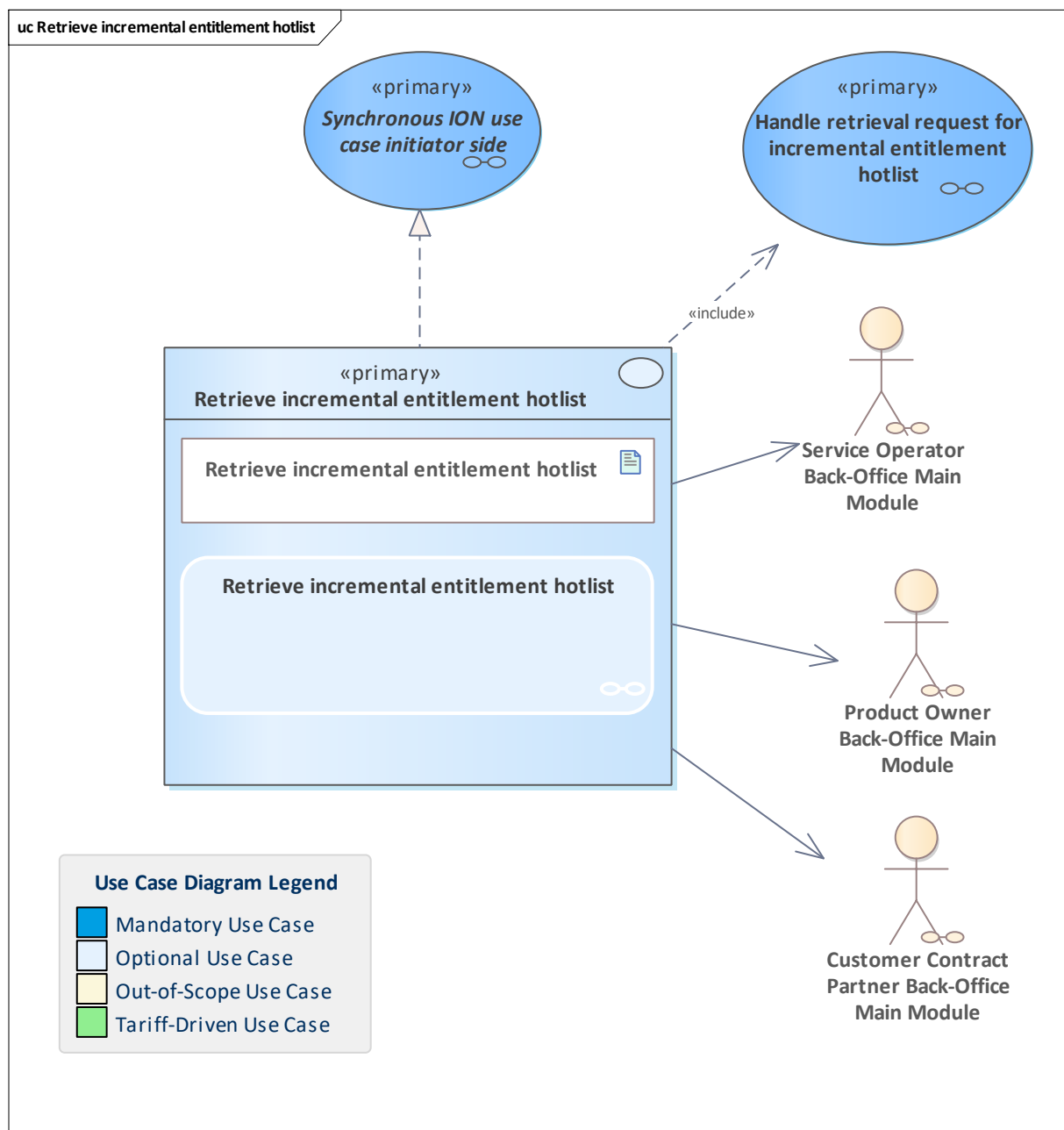


<b>Use Case</b>	<a href="#">Retrieve entitlement hotlist with product information</a>
<b>Description</b>	The SO, PO and CCP want to update their entitlement hotlist inventory by retrieving the entitlement hotlist with additionally contained product information from the hotlist service system. Please note that the entitlement hotlist can be retrieved either as an incremental or a total hotlist, as well as a hotlist with product information (this use case).
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for entitlement hotlist with product information</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Entitlement hotlist : EntitlementHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve entitlement hotlist with product information</a>

## 5.2.18 Optional: Retrieve incremental entitlement hotlist



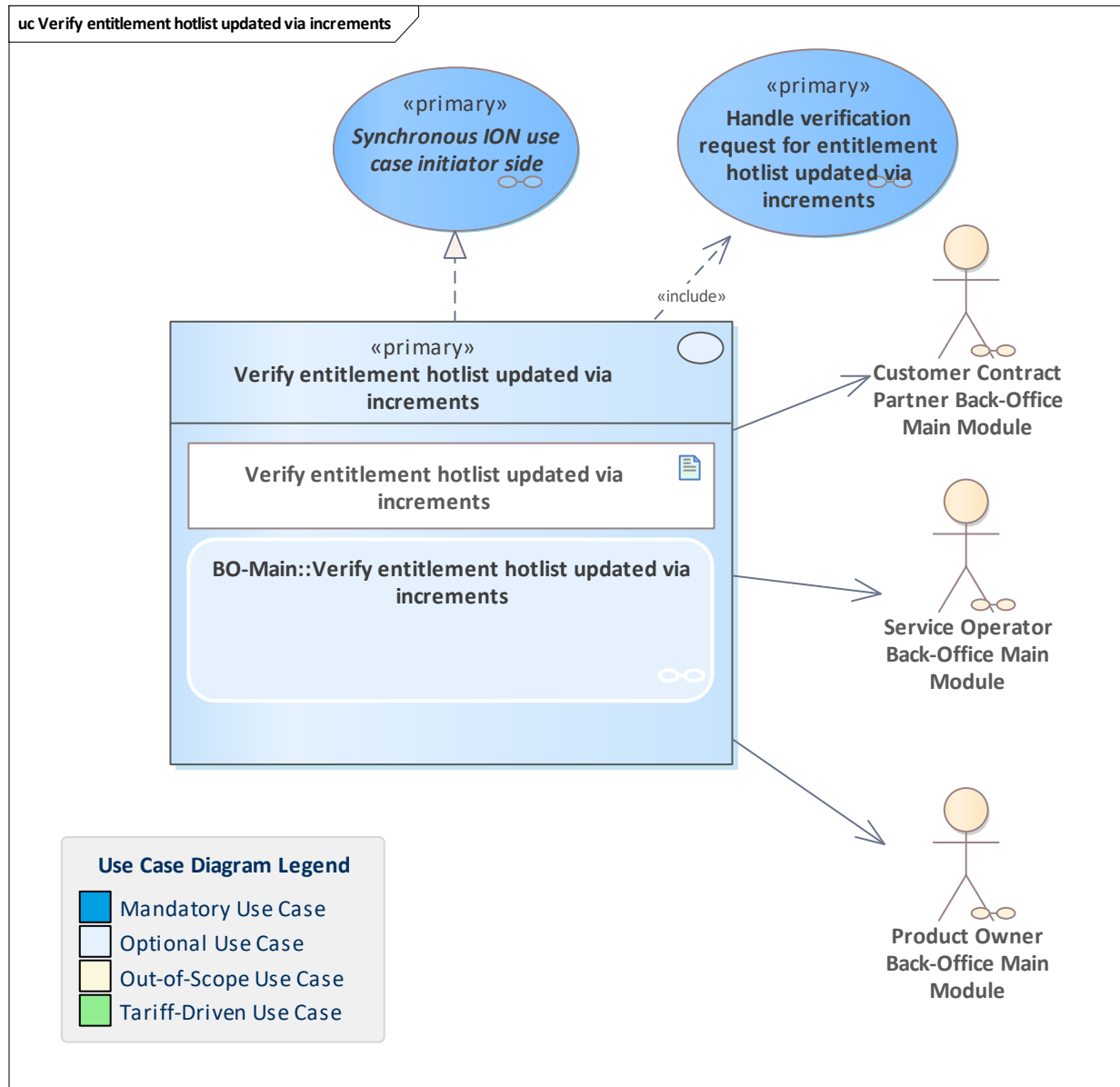
<b>Use Case</b>	<a href="#">Retrieve incremental entitlement hotlist</a>
<b>Description</b>	The SO, PO and CCP want to update their entitlement hotlist inventory by retrieving the current incremental entitlement hotlist from the hotlist service system. Please note that the entitlement hotlist can be retrieved either as an incremental (this use case) or a total hotlist, as well as a hotlist with product information.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a>



	<a href="#">Product Owner Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for incremental entitlement hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Updated entitlement hotlist : EntitlementHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve incremental entitlement hotlist</a>



## 5.2.19 Optional: Verify entitlement hotlist updated via increments

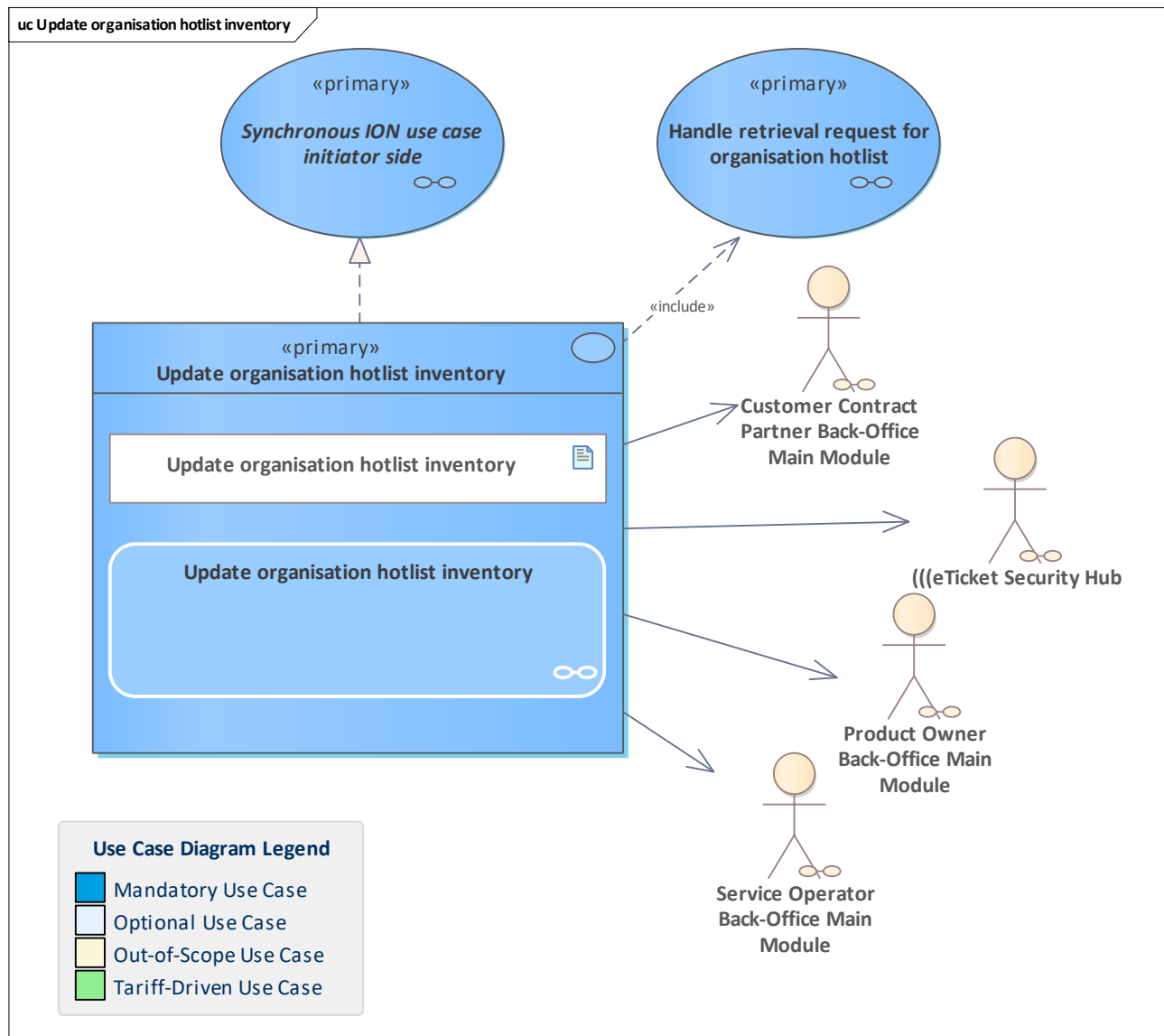


<b>Use Case</b>	<a href="#">Verify entitlement hotlist updated via increments</a>
<b>Description</b>	After updating the entitlement hotlist inventory via the last incremental entitlement hotlist, the participant must ensure that the updated hotlist inventory is the same as the hotlist inventory of the hotlist service system (filtered to the participant's products). For that reason, participants compute the checksum of all the entitlement IDs in their inventory and send it to the hotlist service system to verify the value of the checksum. See also <a href="#">Checksum calculation for hotlist and action list verification</a> and <a href="#">Example calculation for an entitlement hotlist inventory</a> .
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>



	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle verification request for entitlement hotlist updated via increments</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a> <a href="#">Updated entitlement hotlist : EntitlementHotlist</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Verify entitlement hotlist updated via increments</a>

## 5.2.20 Update organisation hotlist inventory

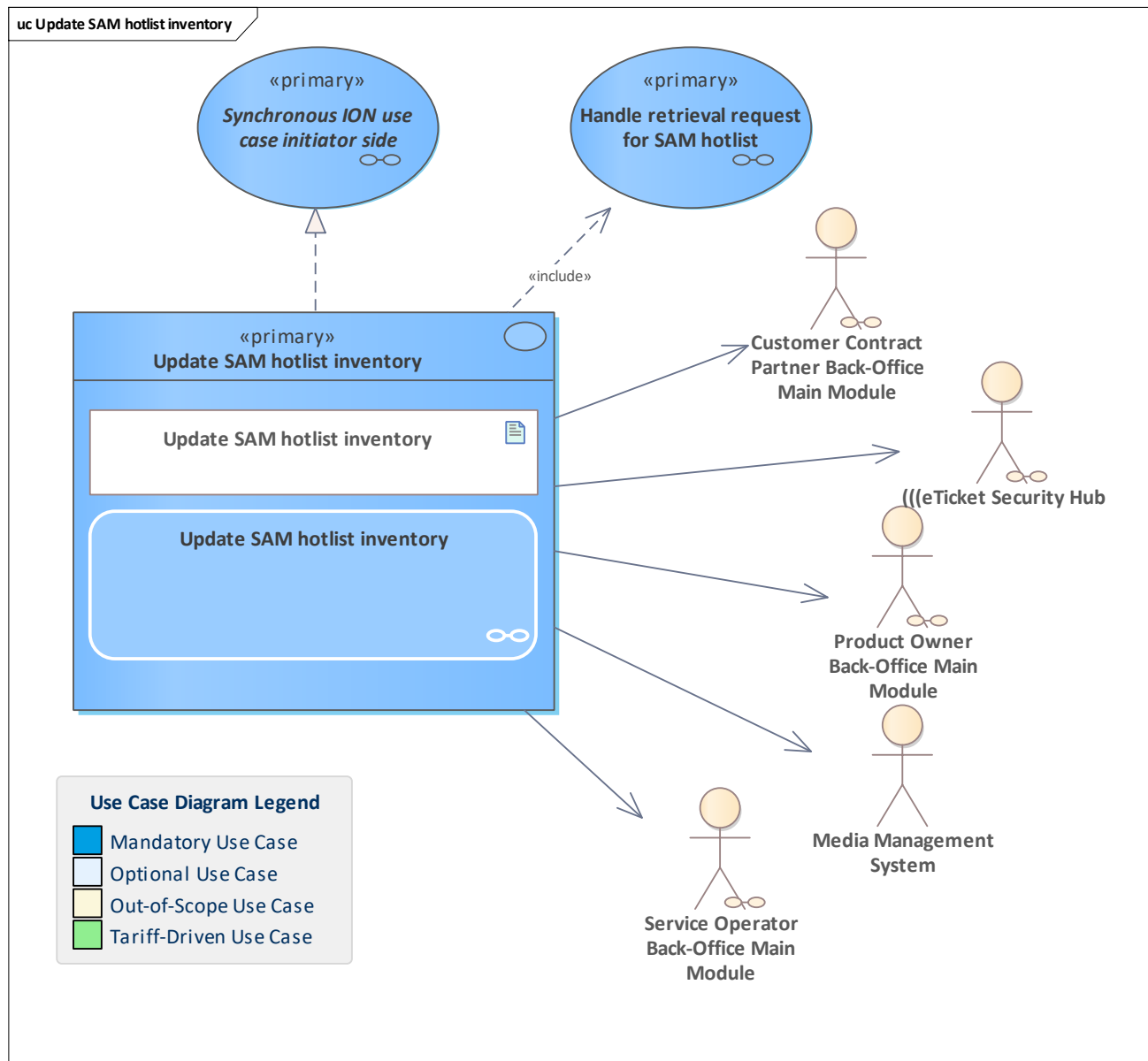


<b>Use Case</b>	<a href="#">Update organisation hotlist inventory</a>
<b>Description</b>	The SO, CCP, PO and the scheme manager's ESH want to update their organisation hotlist inventory by retrieving the current organisation hotlist from the hotlist service system and processing it into their organisation hotlist inventory.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">(((eTicket Security Hub</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases</b>	<a href="#">Handle retrieval request for organisation hotlist</a>



<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	<a href="#">Updated organisation hotlist inventory</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Update organisation hotlist inventory</a>

## 5.2.21 Update SAM hotlist inventory

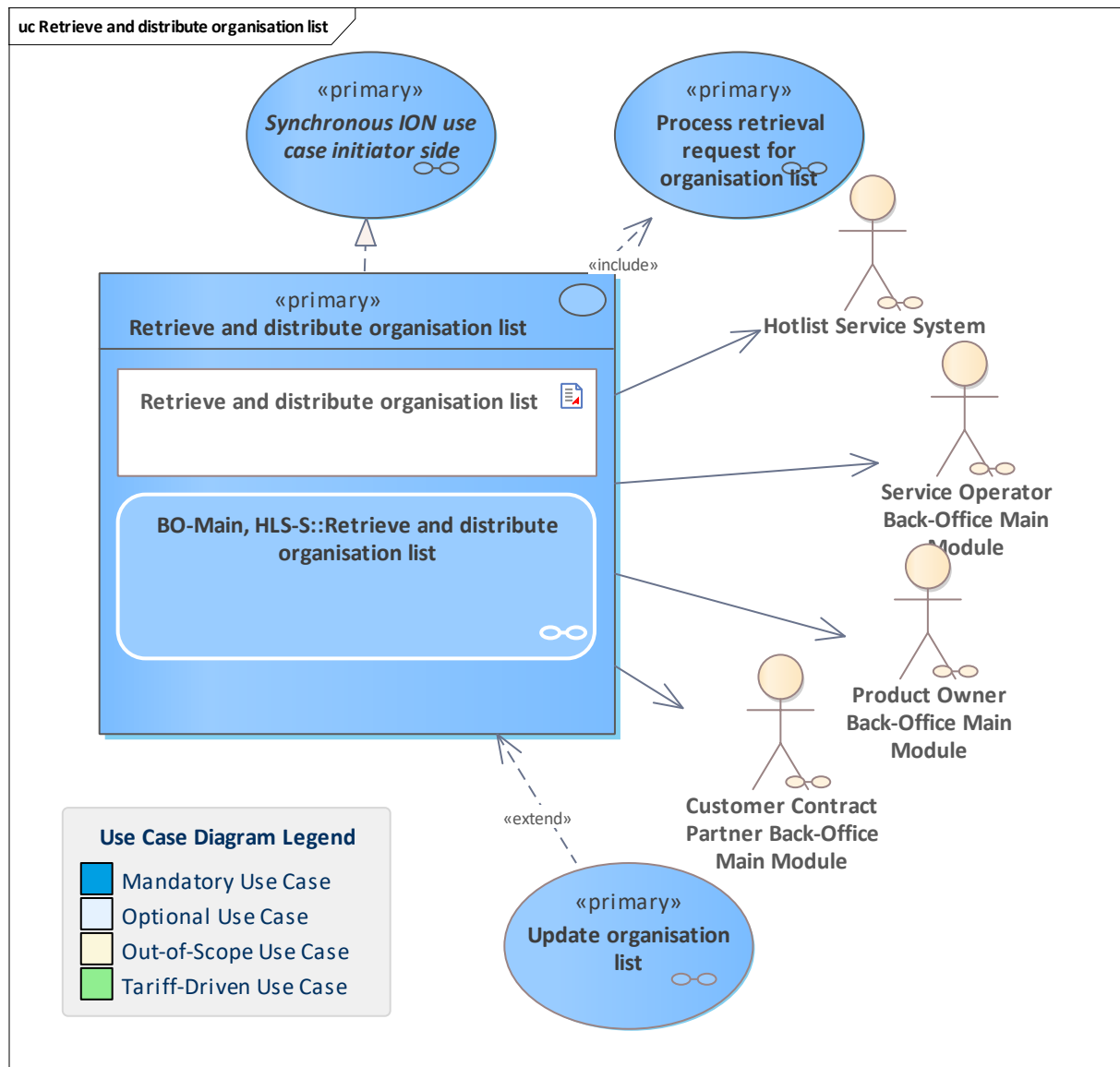


<b>Use Case</b>	<u>Update SAM hotlist inventory</u>
<b>Description</b>	The SO, CCP, PO and the scheme manager's ESH and MMS want to update their SAM hotlist inventory by retrieving the current SAM hotlist from the hotlist service system and processing it into the SAM hotlist inventory.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u> <u>Service Operator Back-Office Main Module</u> <u>Product Owner Back-Office Main Module</u> <u>Media Management System</u> <u>(((eTicket Security Hub</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for SAM hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	<a href="#">Updated SAM hotlist inventory</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Update SAM hotlist inventory</a>

## 5.2.22 Retrieve and distribute organisation list



<b>Use Case</b>	<a href="#">Retrieve and distribute organisation list</a>
<b>Description</b>	The registrar, as part of the scheme manager, provides a list of organisations. This list can be retrieved by all participants daily. An organisation list is distributed to the terminals by CCP and SO. Please note that the list does not involve any IP addresses of the organisations due to data protection.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Hotlist Service System</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Update organisation list</a>



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Process retrieval request for organisation list</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main, HLS-S::Retrieve and distribute organisation list</a>



## 6 Basic Bundle Terminal Operator System - Foundation

This functionality bundle contains use cases that all back-office systems of CCP and SO must implement. CCP and SO are terminal operators whose systems interact with the respective terminals.

### 6.1 Overview

Handle SAM hotlisting demand

Check and add SAM to hotlist

Retrieve application hotlist

Optional: Retrieve incremental application hotlist

Optional: Verify application hotlist updated via increments

Update authentication key hotlist inventory

Update hotlist inventory from operational perspective

Distribute SAM configuration

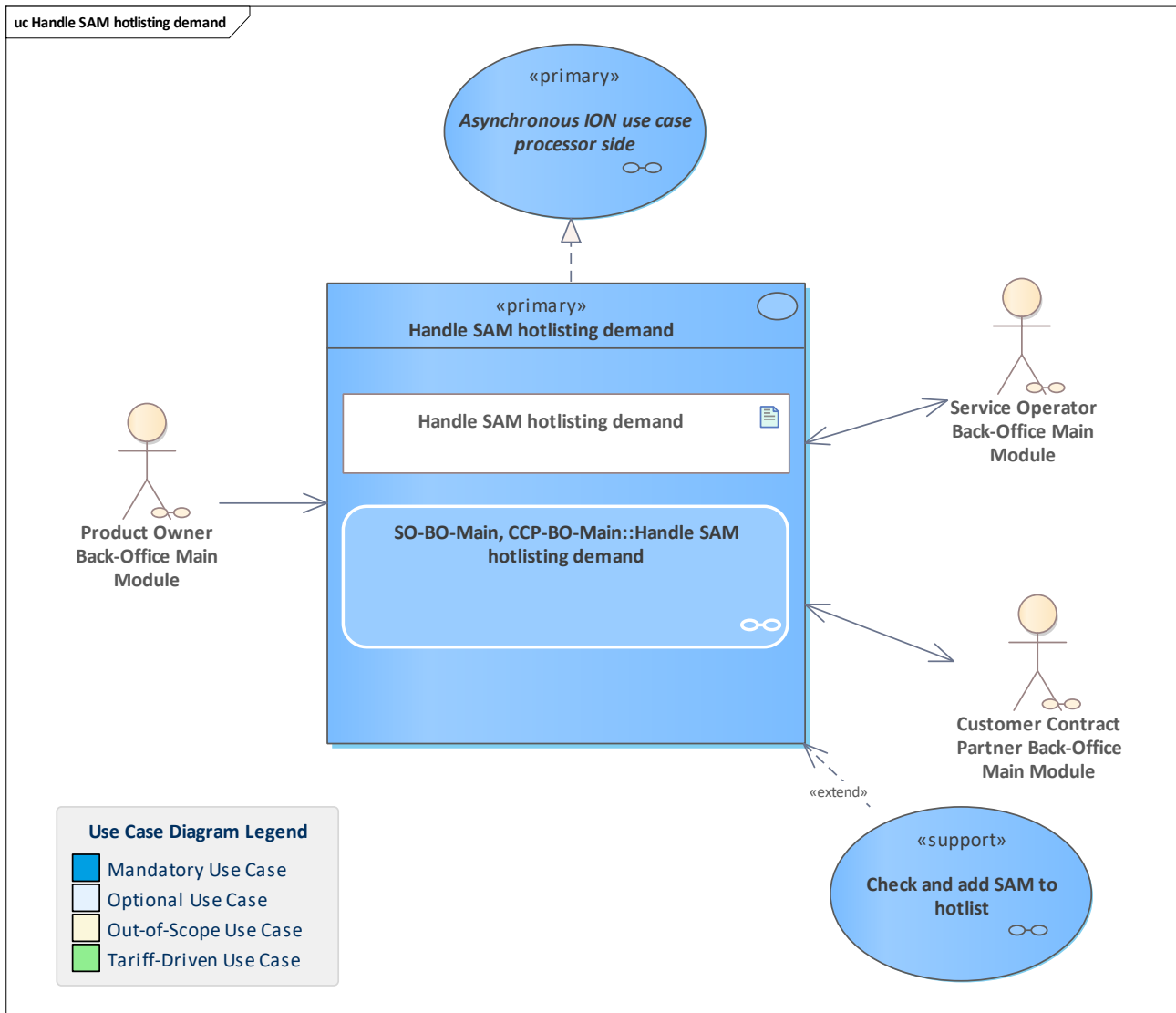
Optional: Distribute the SAM reset script

Distribute tariff module

Monitor SAMs from operational perspective

### 6.2 Use Cases

#### 6.2.1 Handle SAM hotlisting demand

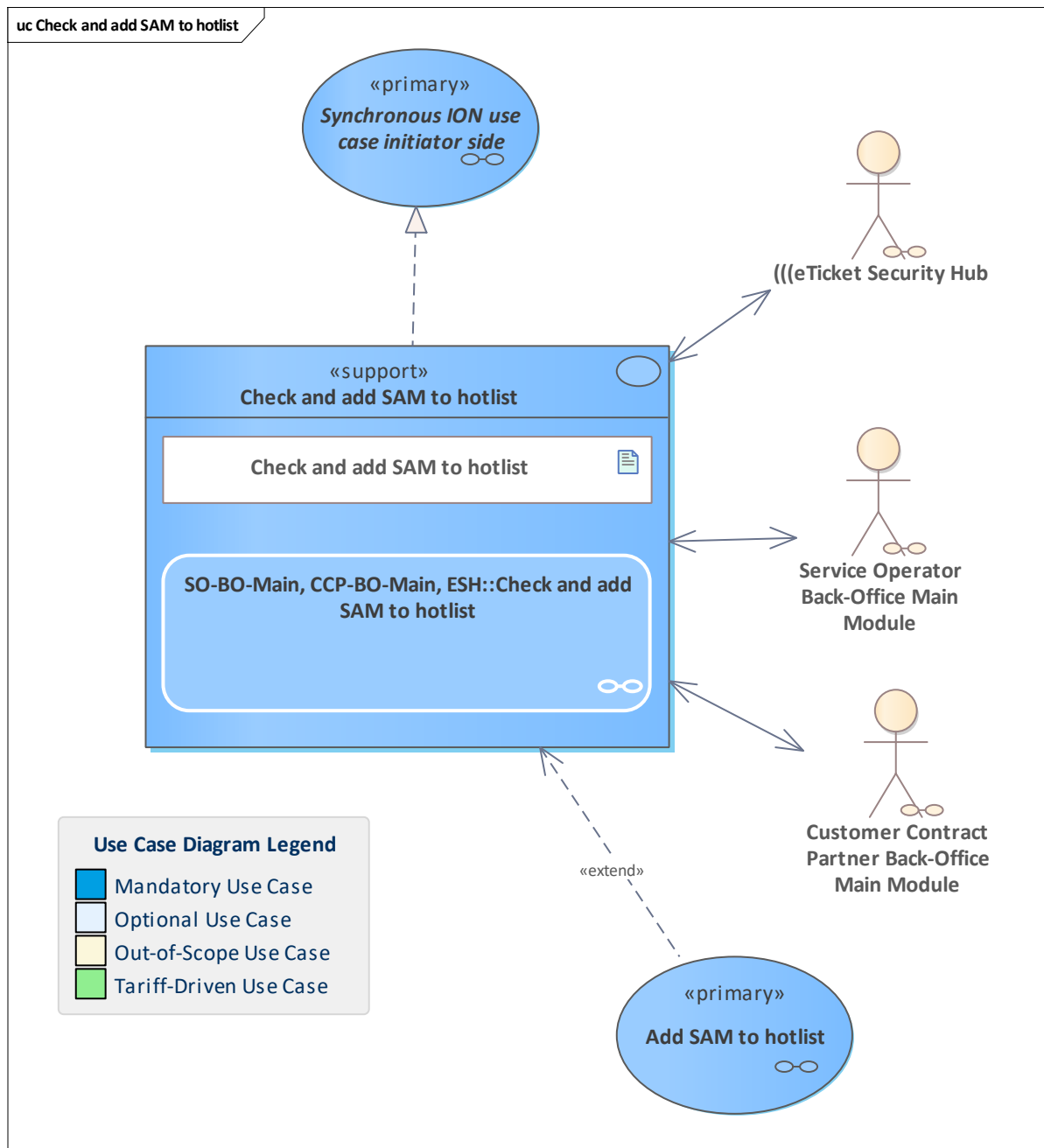


<b>Use Case</b>	<a href="#">Handle SAM hotlisting demand</a>
<b>Description</b>	<p>The owner (SO or CCP) of a SAM to be hotlisted checks the hotlisting demand. If the result of the check is such that a hotlisting is required, it will inform the hotlist service system to have the SAM added to the SAM hotlist. Otherwise, the demand will be rejected and there is no need to communicate with the hotlist service system.</p> <p>Please note that if there is more than one demand for hotlisting the same SAM, this has to be considered in the check for a required hotlisting but will not result in an exception. Especially for the monitoring of a third-party system, it must be possible to demand hotlisting even if the same object was demanded for hotlisting in the past.</p> <p>Please note that the scheme manager is allowed to hotlist SAMs, for example in case of a hotlisted organisation or as an escalation instance.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>



	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Check and add SAM to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Demand SAM hotlisting : demandSamHotlisting</a>
<b>Outputs</b>	<a href="#">Demand SAM hotlisting response : demandSamHotlistingResponse</a>
<b>Error Cases</b>	<a href="#">E_CO_APP_INSTANCE_ID_UNKNOWN</a> <a href="#">E_CO_WRONG_SECURITY_LEVEL</a> <a href="#">Demand SAM hotlisting exception : demandSamHotlistingException</a>
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle SAM hotlisting demand</a>

## 6.2.2 Check and add SAM to hotlist

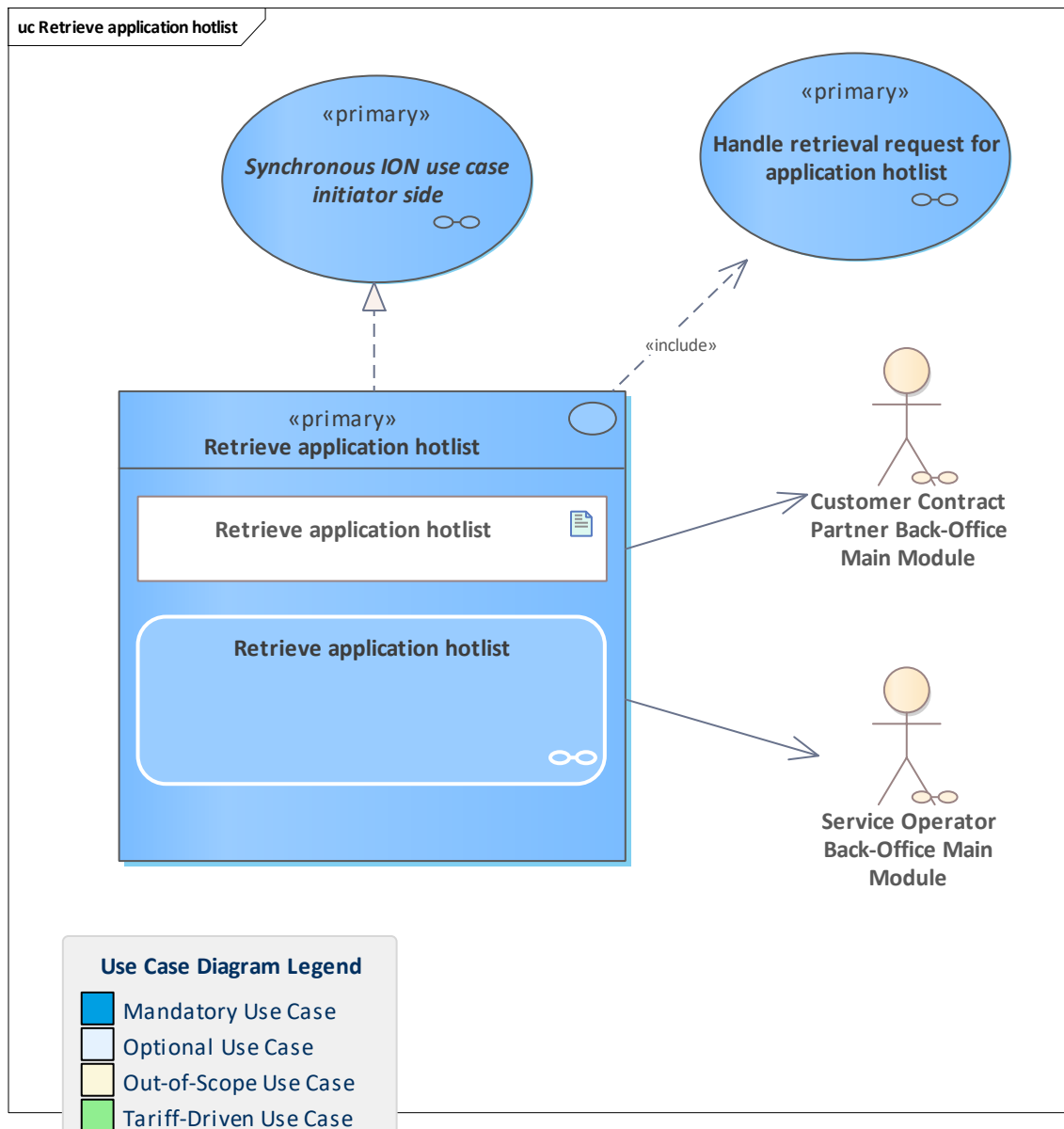


Use Case	Check and add SAM to hotlist
Description	<p>Supporting use case for the Scheme Manager, SO or CCP.</p> <p>The parameters required to add a SAM to the hotlist are checked, especially if another system has already requested the SAM to be added to the hotlist.</p> <p>In addition, the SAM owner may add specific counter information that is not available to a third party system.</p> <p>The hotlist service system is requested to add the SAM to the SAM</p>



	hotlist.
<b>Initiating Actor</b>	<a href="#">(((eTicket Security Hub</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">(((eTicket Security Hub</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Add SAM to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Blocking Reason : BlockingReason</a> <a href="#">SAM action counter : ActionCounter</a> <a href="#">SAM ID : AppInstanceId</a> <a href="#">SAM entitlement issuance counter : EntitlementIssuanceCounter</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main, ESH::Check and add SAM to hotlist</a>

## 6.2.3 Retrieve application hotlist

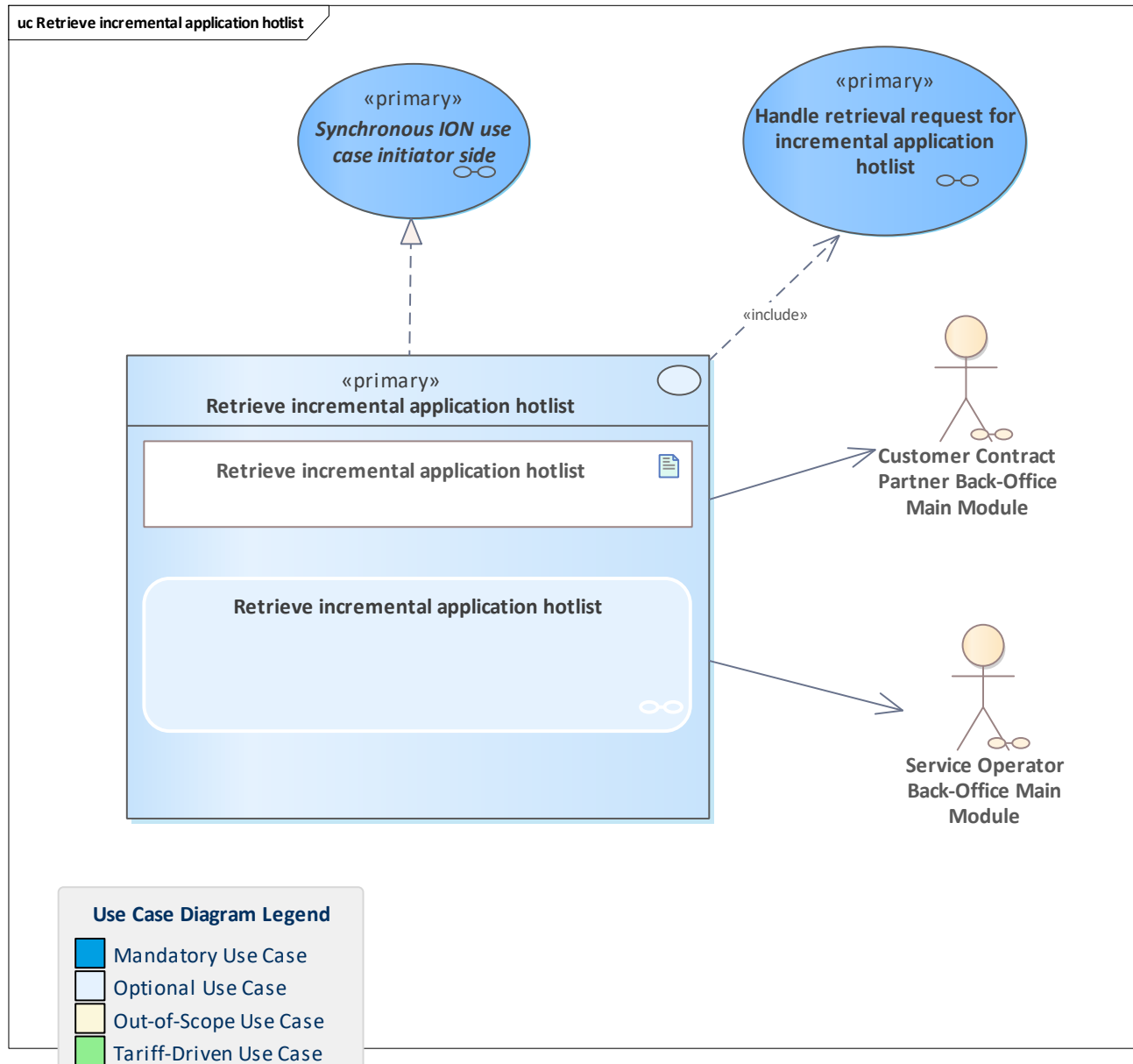


<b>Use Case</b>	<a href="#">Retrieve application hotlist</a>
<b>Description</b>	The SO and CCP want to update their application hotlist inventory by retrieving the application hotlist from the hotlist service system. Please note that the application hotlist can be retrieved as an incremental or a total hotlist.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for application hotlist</a>



<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Application hotlist : ApplicationHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve application hotlist</a>

## 6.2.4 Optional: Retrieve incremental application hotlist



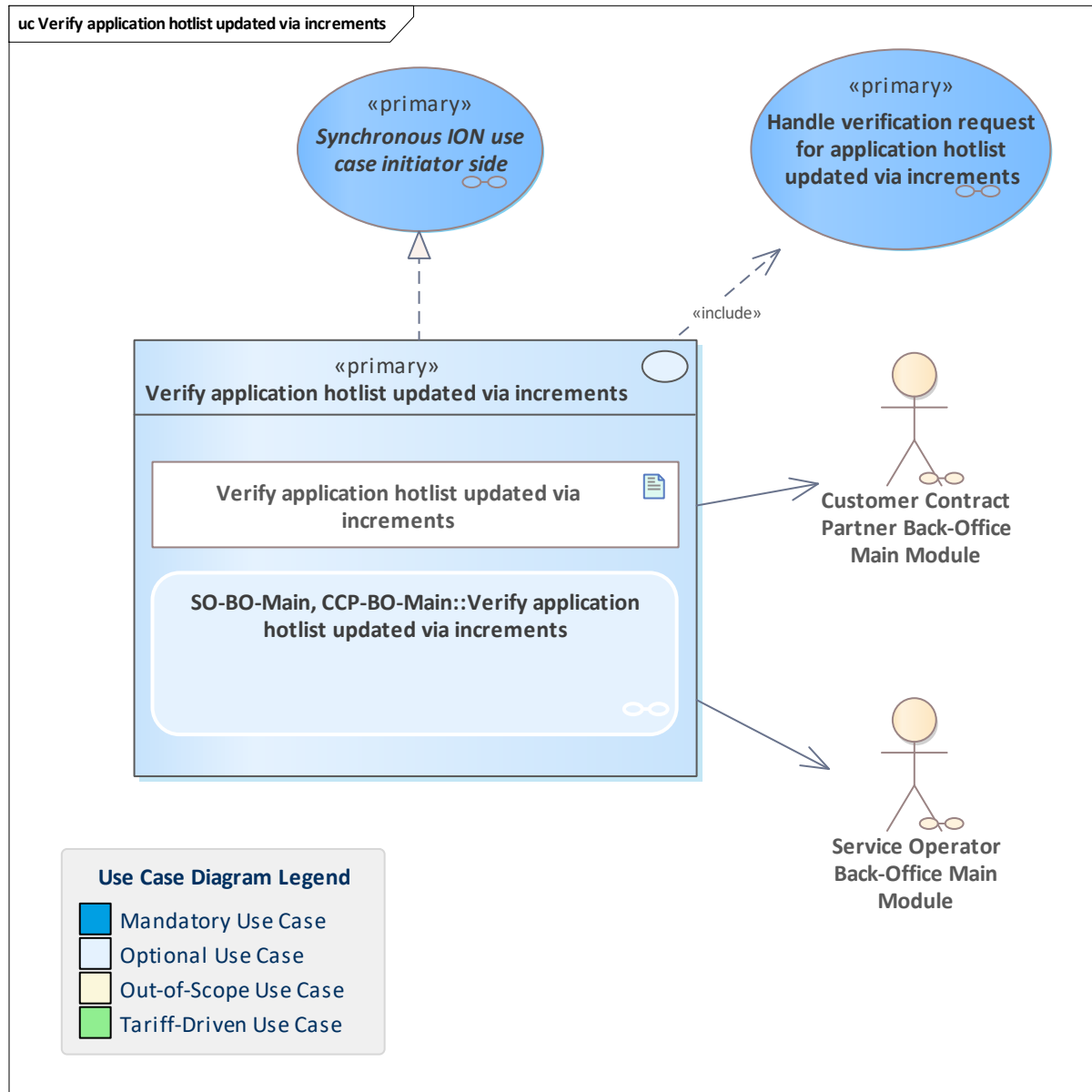
<b>Use Case</b>	<a href="#">Retrieve incremental application hotlist</a>
<b>Description</b>	The SO and CCP want to update their application hotlist inventory by retrieving the current incremental application hotlist from the hotlist service system. Please note that the application hotlist can be retrieved as an incremental or a total hotlist.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	





<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for incremental application hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Updated application hotlist : ApplicationHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve incremental application hotlist</a>

## 6.2.5 Optional: Verify application hotlist updated via increments

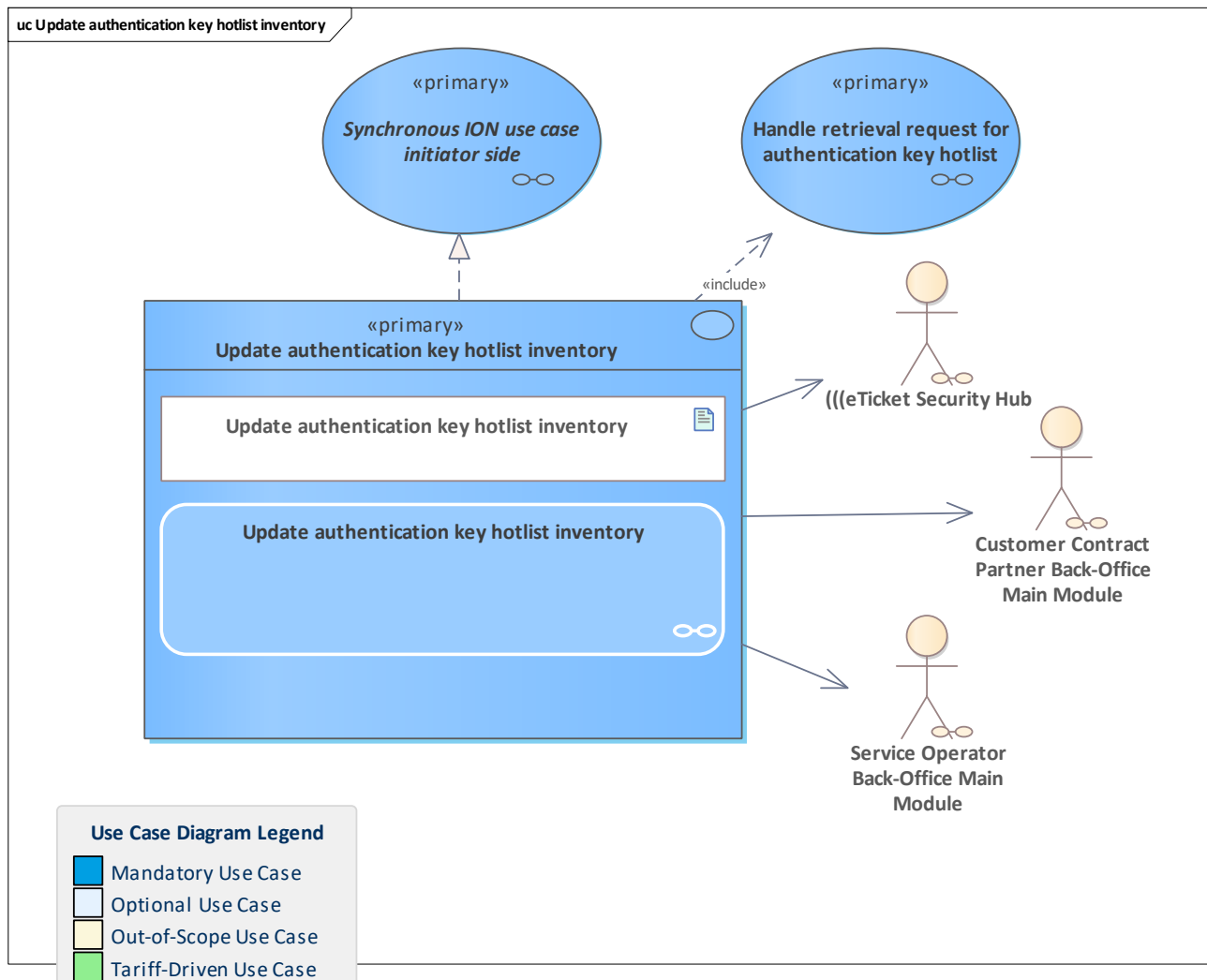


<b>Use Case</b>	<u>Verify application hotlist updated via increments</u>
<b>Description</b>	<p>After updating the application hotlist inventory via the last incremental application hotlist, the participant must ensure that the updated hotlist inventory is the same as the hotlist inventory of the hotlist service system.</p> <p>For that reason, participants compute the checksum of all the application instance IDs in their inventory and send it to the hotlist service system to verify the value of the checksum. See also <a href="#">Checksum calculation for hotlist and action list verification</a> and <a href="#">Example calculation for an application hotlist inventory</a>.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u>



	<a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle verification request for application hotlist updated via increments</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a> <a href="#">Updated application hotlist : ApplicationHotlist</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Verify application hotlist updated via increments</a>

## 6.2.6 Update authentication key hotlist inventory

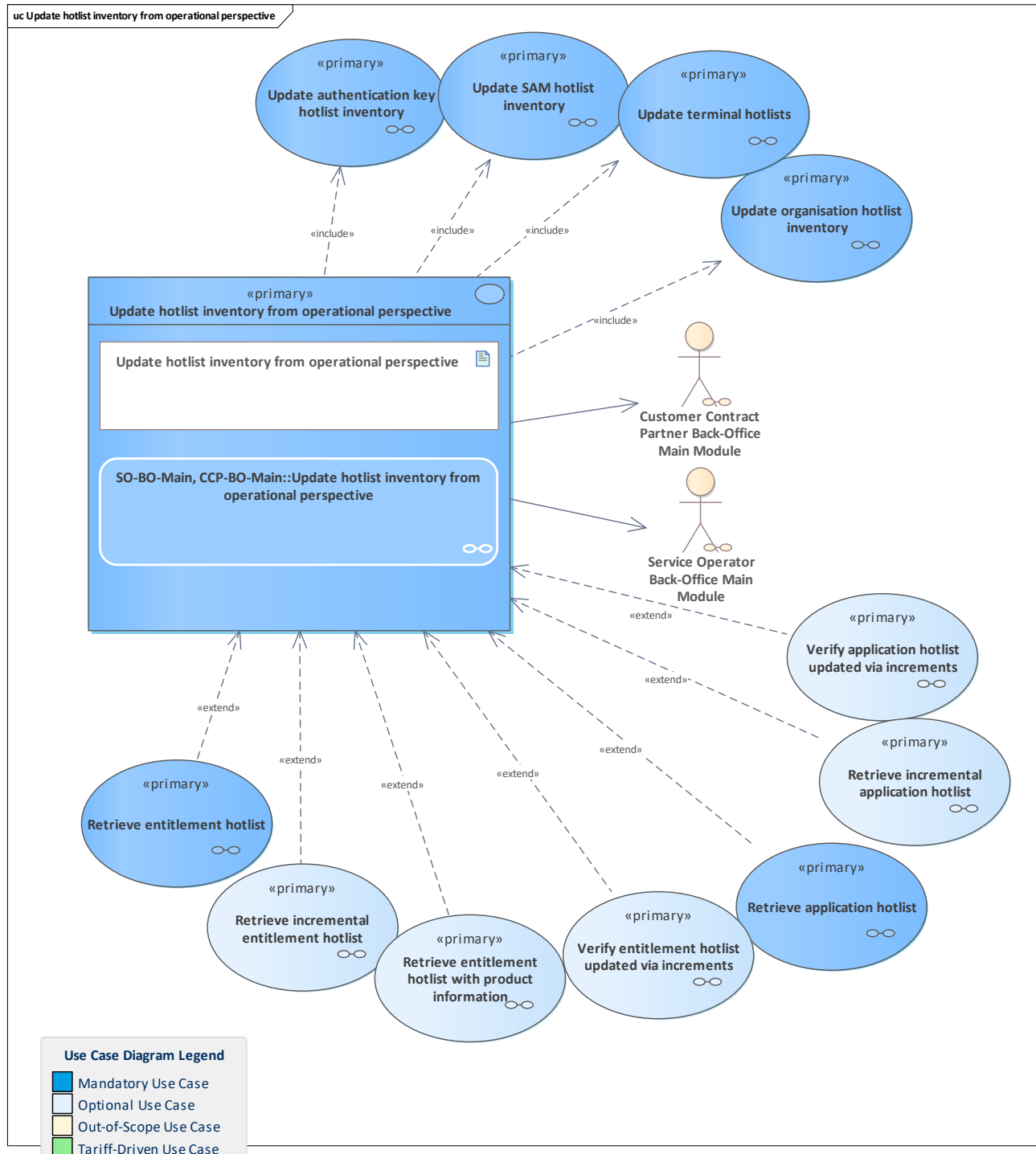


<b>Use Case</b>	<a href="#">Update authentication key hotlist inventory</a>
<b>Description</b>	The SO, CCP and the scheme manager's ESH want to update the authentication key hotlist inventory by retrieving the current authentication key hotlist from the hotlist service system and processing it into their authentication key hotlist inventory.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">(((eTicket Security Hub</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for authentication key hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	



<b>Inputs</b>	
<b>Outputs</b>	<a href="#">Updated authentication key hotlist inventory</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Update authentication key hotlist inventory</a>

## 6.2.7 Update hotlist inventory from operational perspective

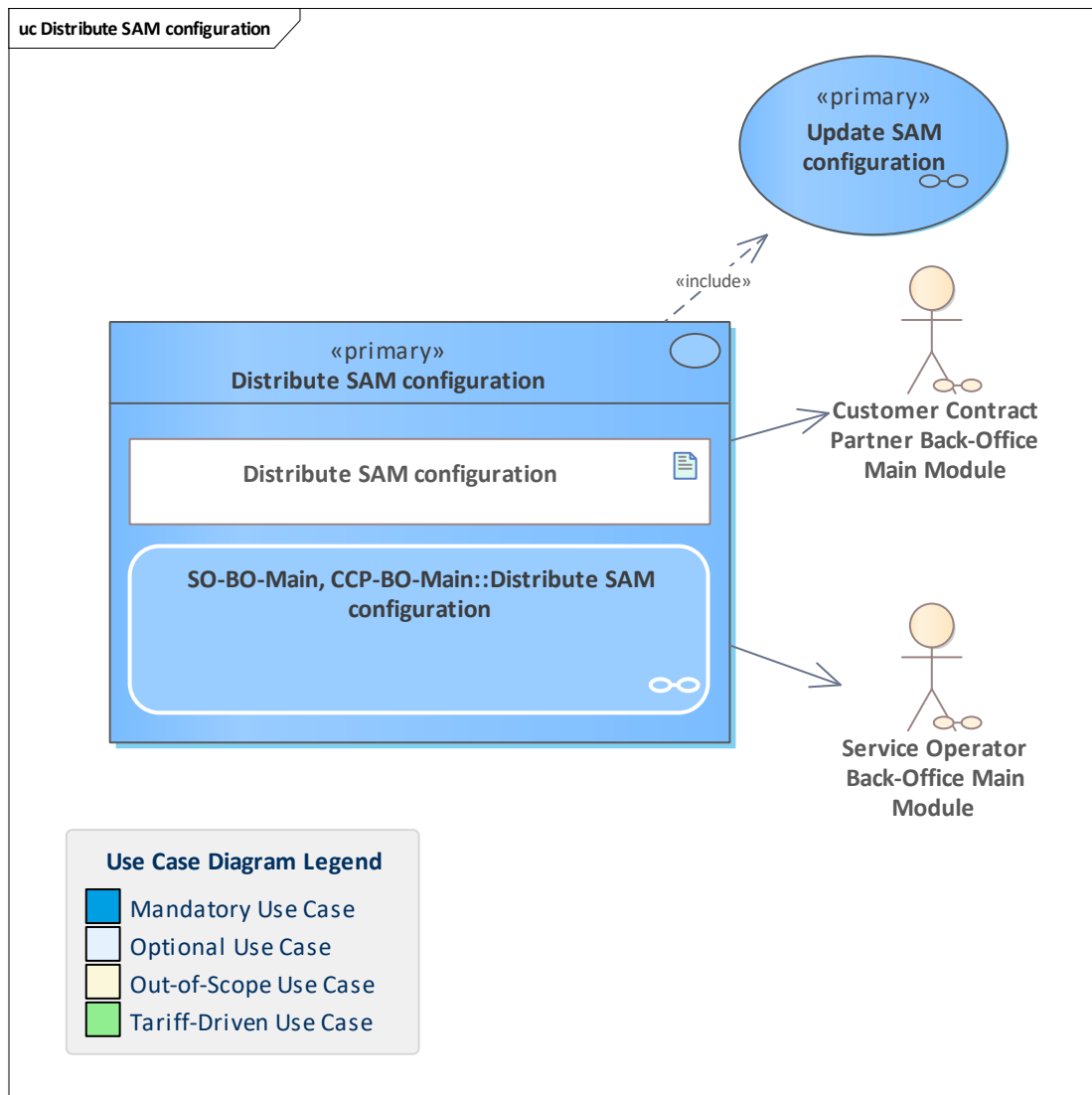


Use Case	Update hotlist inventory from operational perspective
Description	A CCP or SO wants to retrieve the currently available hotlists and distribute them to its terminals to inspect accessed entitlements and applications, as well as to deactivate a terminal if its SAM has been hotlisted.



	<p>The available hotlists to be updated and distributed are:</p> <ul style="list-style-type: none"><li>• Application hotlist:  either total application hotlist or incremental application hotlist</li><li>• Entitlement hotlist:  either total entitlement hotlist or incremental entitlement hotlist or total entitlement hotlist with product information</li><li>• SAM hotlist</li><li>• Organisation hotlist</li><li>• Authentication key hotlist</li></ul>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Retrieve incremental application hotlist</a> / <a href="#">Verify application hotlist updated via increments</a> / <a href="#">Retrieve application hotlist</a> / <a href="#">Verify entitlement hotlist updated via increments</a> / <a href="#">Retrieve entitlement hotlist</a> / <a href="#">Retrieve incremental entitlement hotlist</a> / <a href="#">Retrieve entitlement hotlist with product information</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Update SAM hotlist inventory</a> / <a href="#">Update authentication key hotlist inventory</a> / <a href="#">Update organisation hotlist inventory</a> / <a href="#">Update terminal hotlists</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Update hotlist inventory from operational perspective</a>

## 6.2.8 Distribute SAM configuration



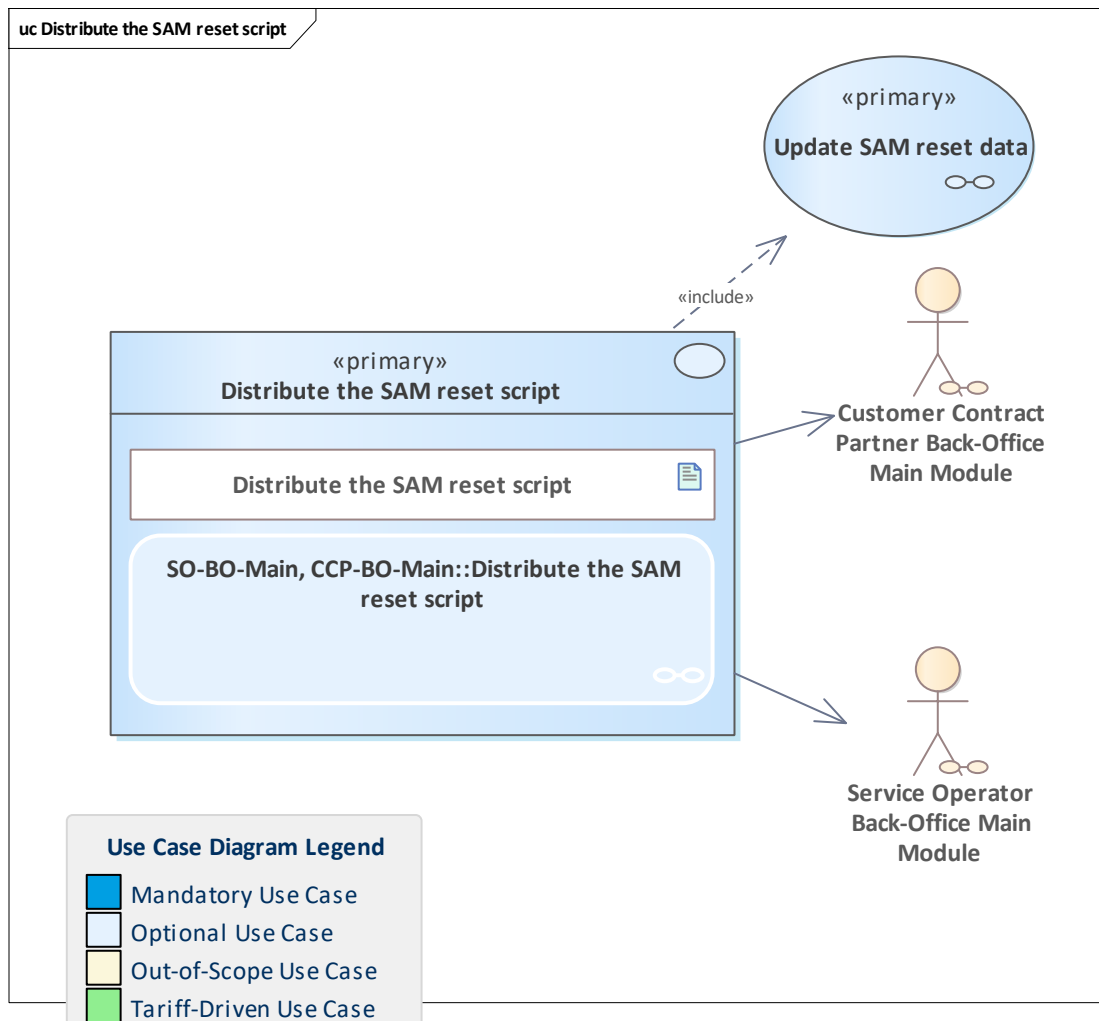
<b>Use Case</b>	<a href="#">Distribute SAM configuration</a>
<b>Description</b>	<p>Required SAM configuration data was received from the ESH. The back-office system distributes the script for each new SAM configuration.</p> <p>In this use case, it is assumed that only one SAM configuration per terminal is required for the sake of simplicity.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Update SAM configuration</a>
<b>Linked Use Cases</b>	





<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">List of SAM configuration</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Distribute SAM configuration</a>

## 6.2.9 Optional: Distribute the SAM reset script

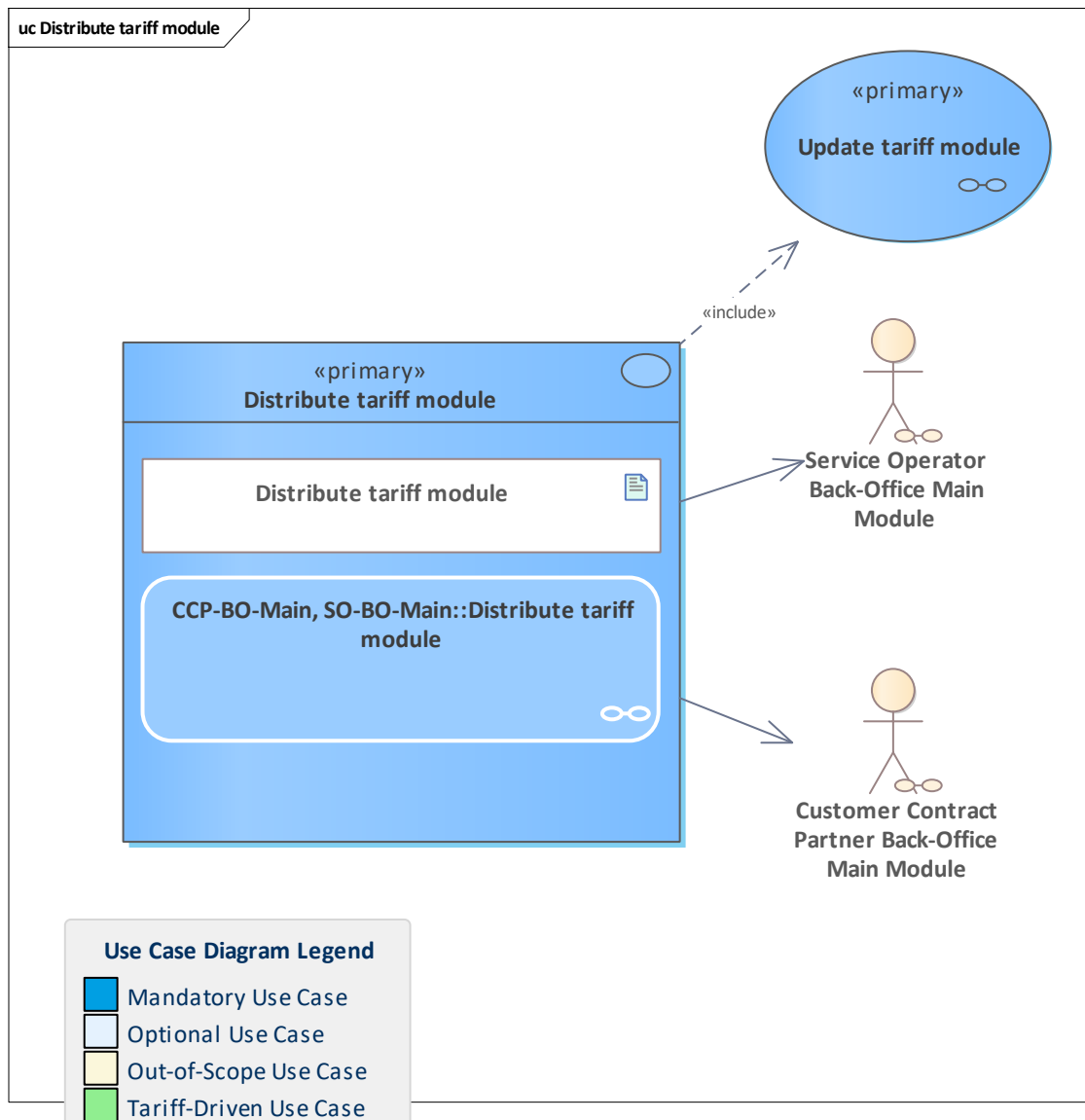


<b>Use Case</b>	<a href="#">Distribute the SAM reset script</a>
<b>Description</b>	<p>Required SAM reset data was received from the ESH. The back-office system distributes the script for each SAM that needs to be reset.</p> <p>In this use case it is assumed that only one SAM reset per terminal is required for the sake of simplicity.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Update SAM reset data</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	



<b>Inputs</b>	<a href="#">List of SAM reset data</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Distribute the SAM reset script</a>

## 6.2.10 Distribute tariff module

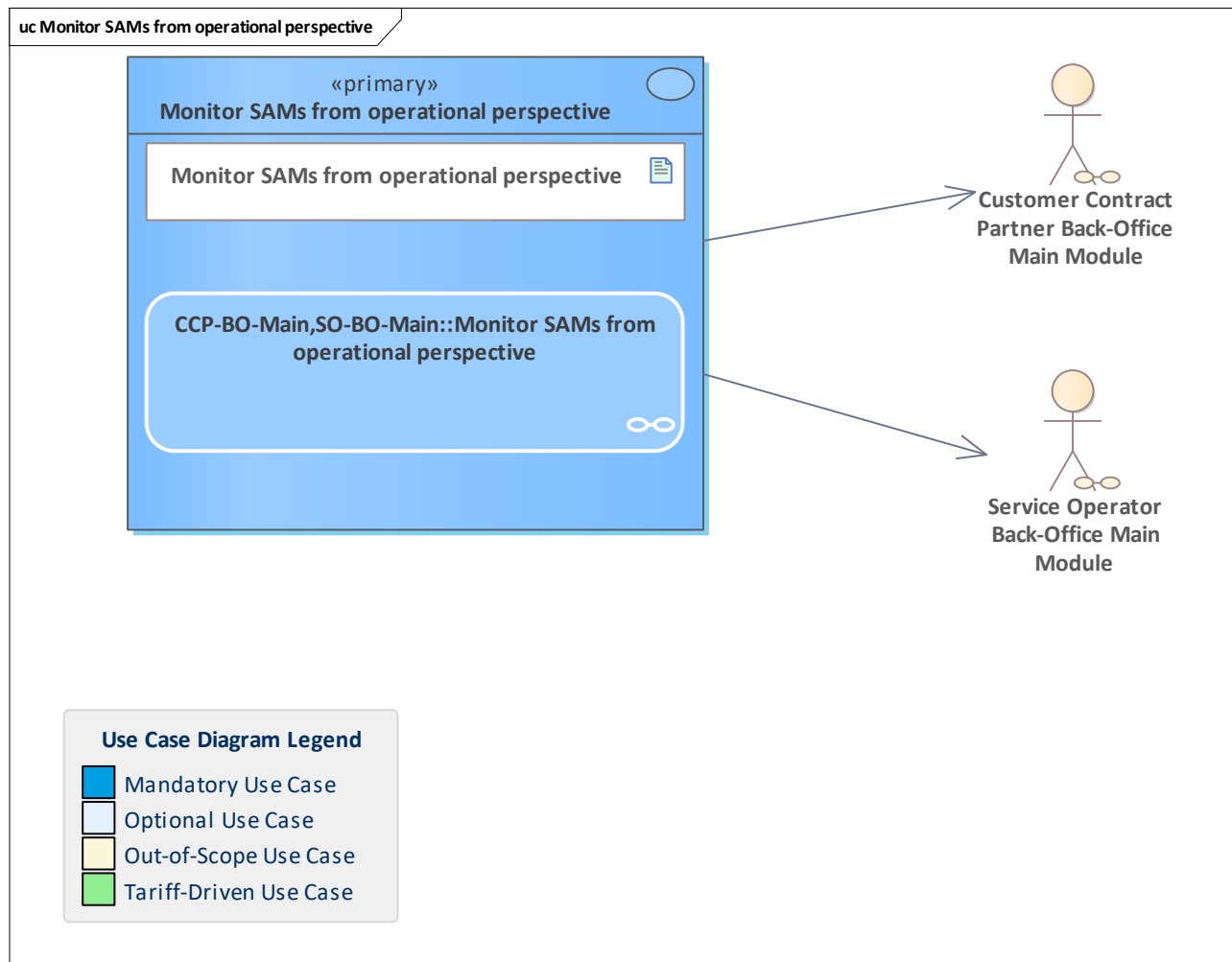


<b>Use Case</b>	<u>Distribute tariff module</u>
<b>Description</b>	<p>The CPP and SO back-office system retrieves tariff modules from POs (out of scope) and stores them in its data store.</p> <p>The SO and CCP create one tariff module for terminals based on the tariff modules received from the POs. The SO and CCP distributes this tariff module for its terminals.</p> <p>This process needs to run periodically to keep the tariff module up to date.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Service Operator Back-Office Main Module</u> <u>Customer Contract Partner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	



<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a> / <a href="#">Update tariff module</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main, SO-BO-Main::Distribute tariff module</a>

## 6.2.11 Monitor SAMs from operational perspective



<b>Use Case</b>	<a href="#">Monitor SAMs from operational perspective</a>
<b>Description</b>	The own SAMs need to be monitored from the operational perspective. All entitlement issuances and action authorisations have to be documented using notifications.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	



<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#"><u>CCP-BO-Main,SO-BO-Main::Monitor SAMs from operational perspective</u></a>

## 7 Basic Bundle Terminal Operator System - Extended Logging

This optional functionality bundle allows the processing of extended logging notifications for an application or an entitlement. The extended logging can also be used for static entitlements.

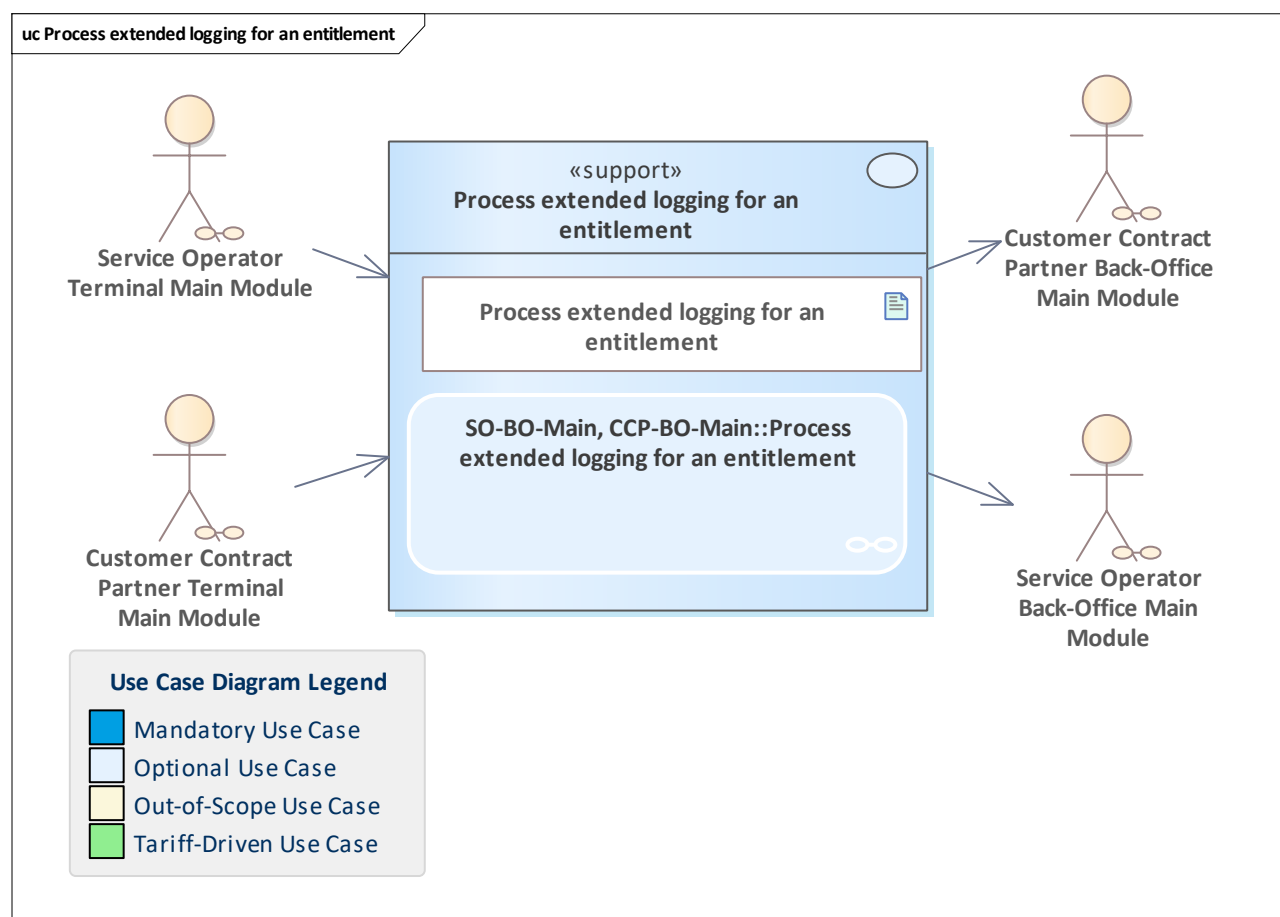
### 7.1 Overview

Optional: Process extended logging for an entitlement

Optional: Process extended logging for an application

### 7.2 Use Cases

#### 7.2.1 Optional: Process extended logging for an entitlement



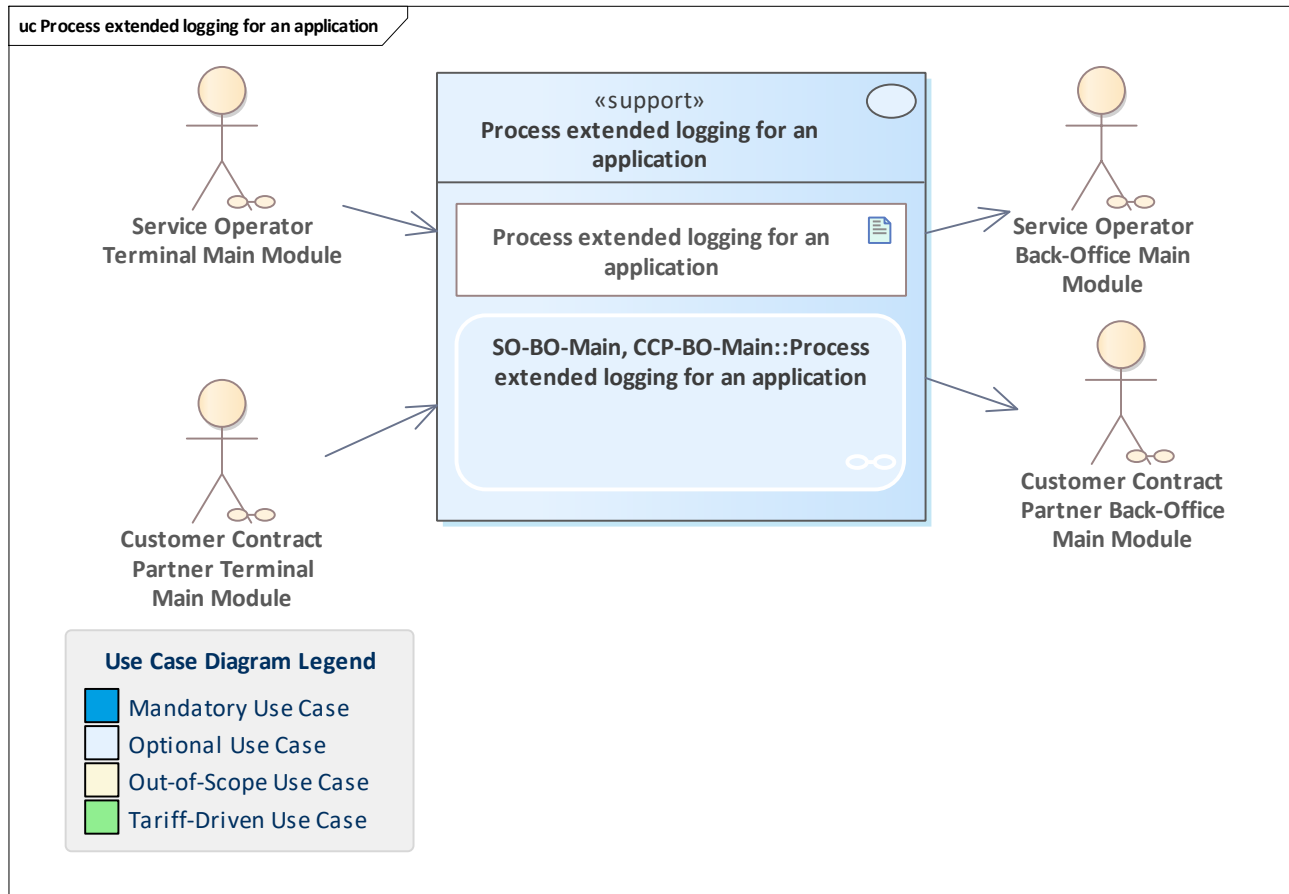
<b>Use Case</b>	<u>Process extended logging for an entitlement</u>
<b>Description</b>	Extended logging for an entitlement is received from a terminal by the back-office system and registered.





<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a> <a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Log for invalid entitlement from terminal : tLogInvalidEntitlement</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Process extended logging for an entitlement</a>

## 7.2.2 Optional: Process extended logging for an application



<b>Use Case</b>	<a href="#">Process extended logging for an application</a>
<b>Description</b>	Extended logging for an application is received from a terminal by the back-office system and registered.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Log for invalid application from terminal : tLogInvalidApplication</a>
<b>Outputs</b>	



<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#"><u>SO-BO-Main, CCP-BO-Main::Process extended logging for an application</u></a>

## 8 Basic Bundle Terminal Operator System - UM with Application

Functionality bundle that covers all the basic use cases required for a terminal operator system that works with user media with an application.

### 8.1 Overview

Handle defective user medium with application

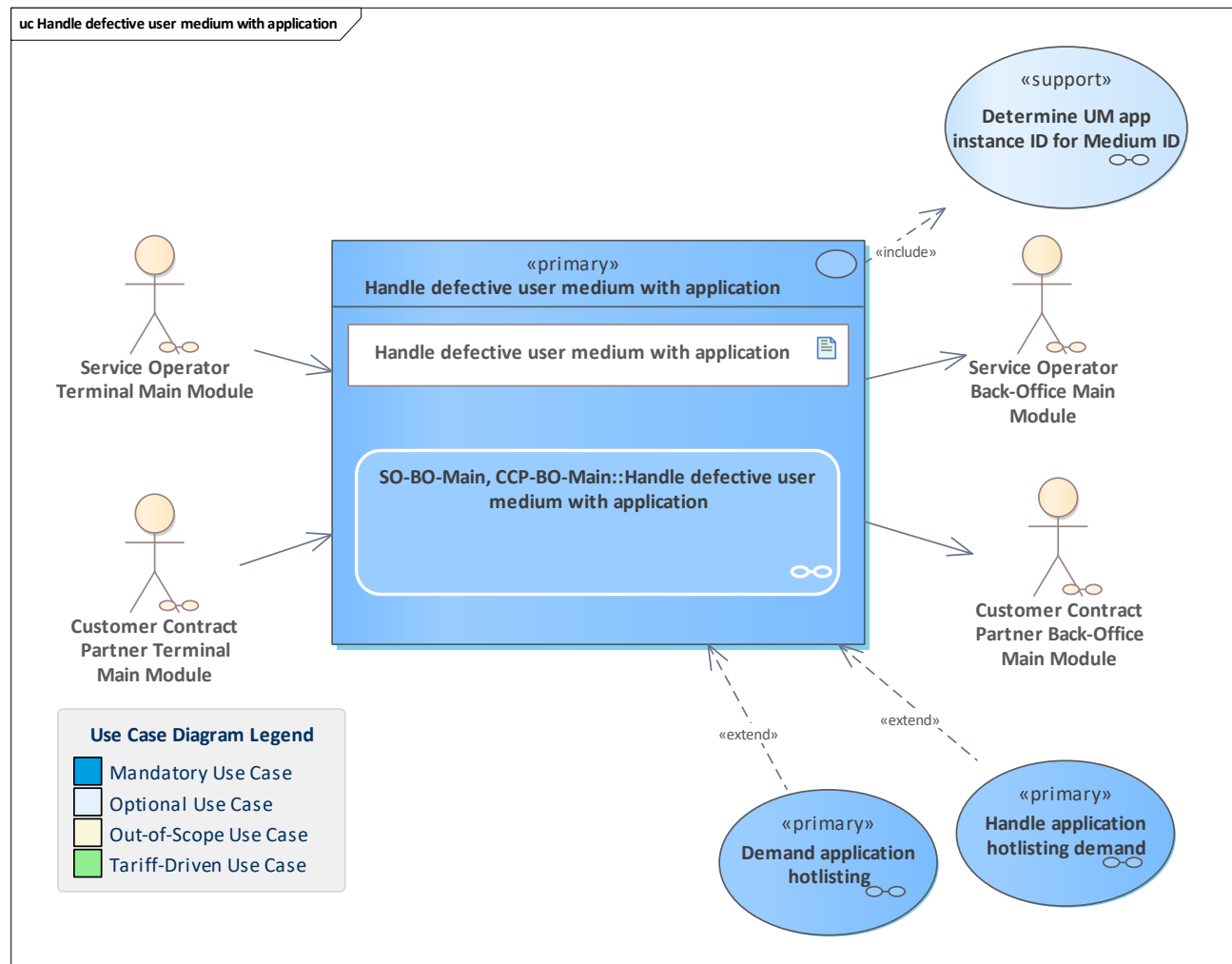
Optional: Determine UM app instance ID for Medium ID

Handle application blocked notification from operational perspective

Handle entitlement blocked notification from operational perspective

### 8.2 Use Cases

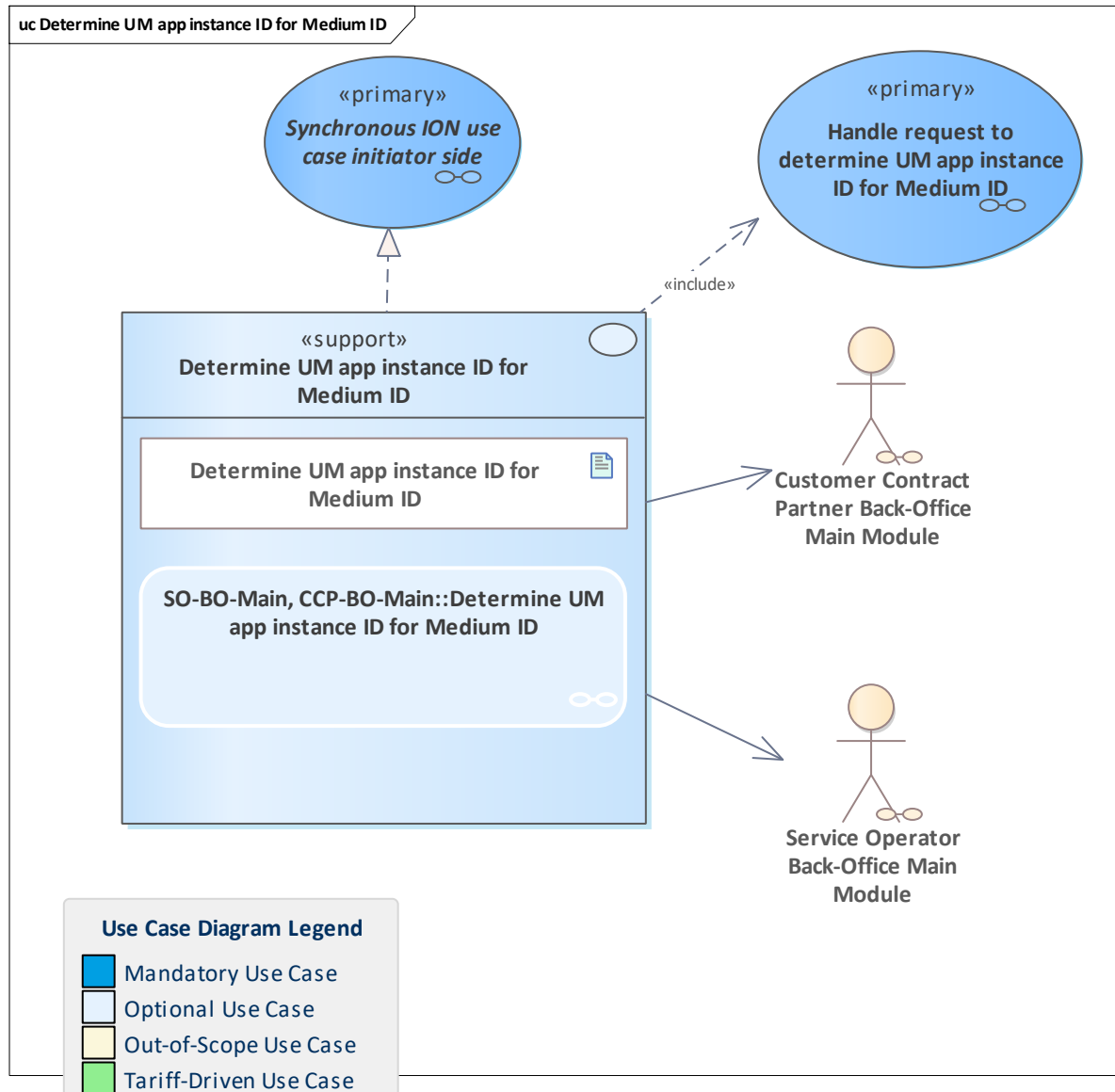
#### 8.2.1 Handle defective user medium with application





<b>Use Case</b>	<a href="#">Handle defective user medium with application</a>
<b>Description</b>	<p>The terminal operator (CCP or SO) back-office system receives the log data of the terminal which detected the defective user medium. The message also contains the medium ID which can be the application instance ID. If the medium ID is proprietary, the application instance ID must be determined for the received medium ID.</p> <p>The application of the application instance ID must be hotlisted. Depending on the role in the current process (pCCP versus sCCP/SO) either a hotlist demand is sent to the application instance owner, or the pCCP communicates directly with the hotlist service.</p>
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a> <a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Handle application hotlisting demand</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a> / <a href="#">Determine UM app instance ID for Medium ID</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Log defective user medium with application from terminal : tLogDefectiveUserMediumWithApplication</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle defective user medium with application</a>

## 8.2.2 Optional: Determine UM app instance ID for Medium ID

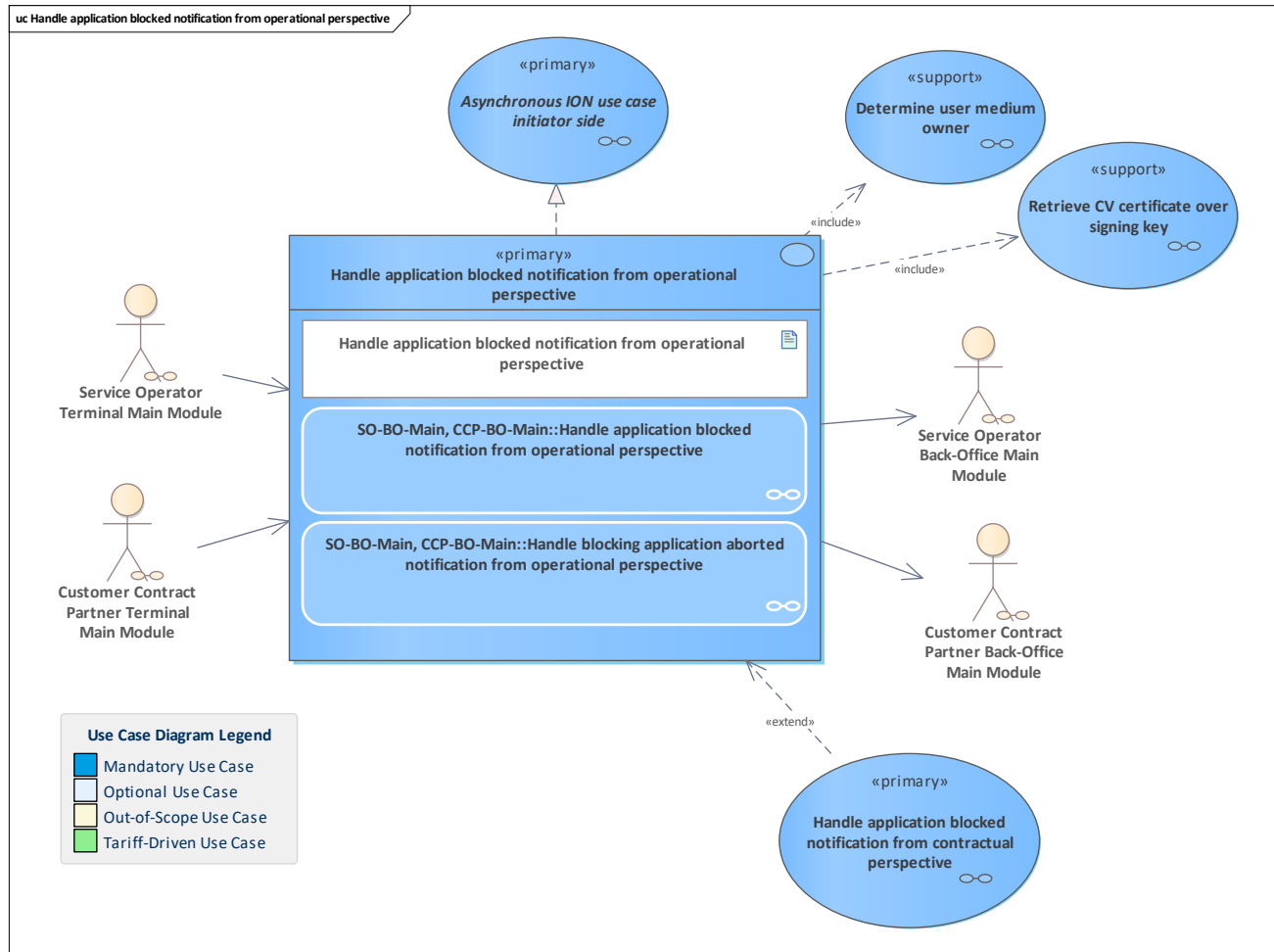


<b>Use Case</b>	<a href="#">Determine UM app instance ID for Medium ID</a>
<b>Description</b>	Determine the user medium application instance ID for a given medium ID using the service provided by the ESH. If the application instance ID is already printed on all user media of the transport company, then this use case is optional.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle request to determine UM app instance ID for Medium ID</a>



<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Medium ID : MediumId</a>
<b>Outputs</b>	<a href="#">UM app Instance ID : AppInstanceId</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Determine UM app instance ID for Medium ID</a>

## 8.2.3 Handle application blocked notification from operational perspective



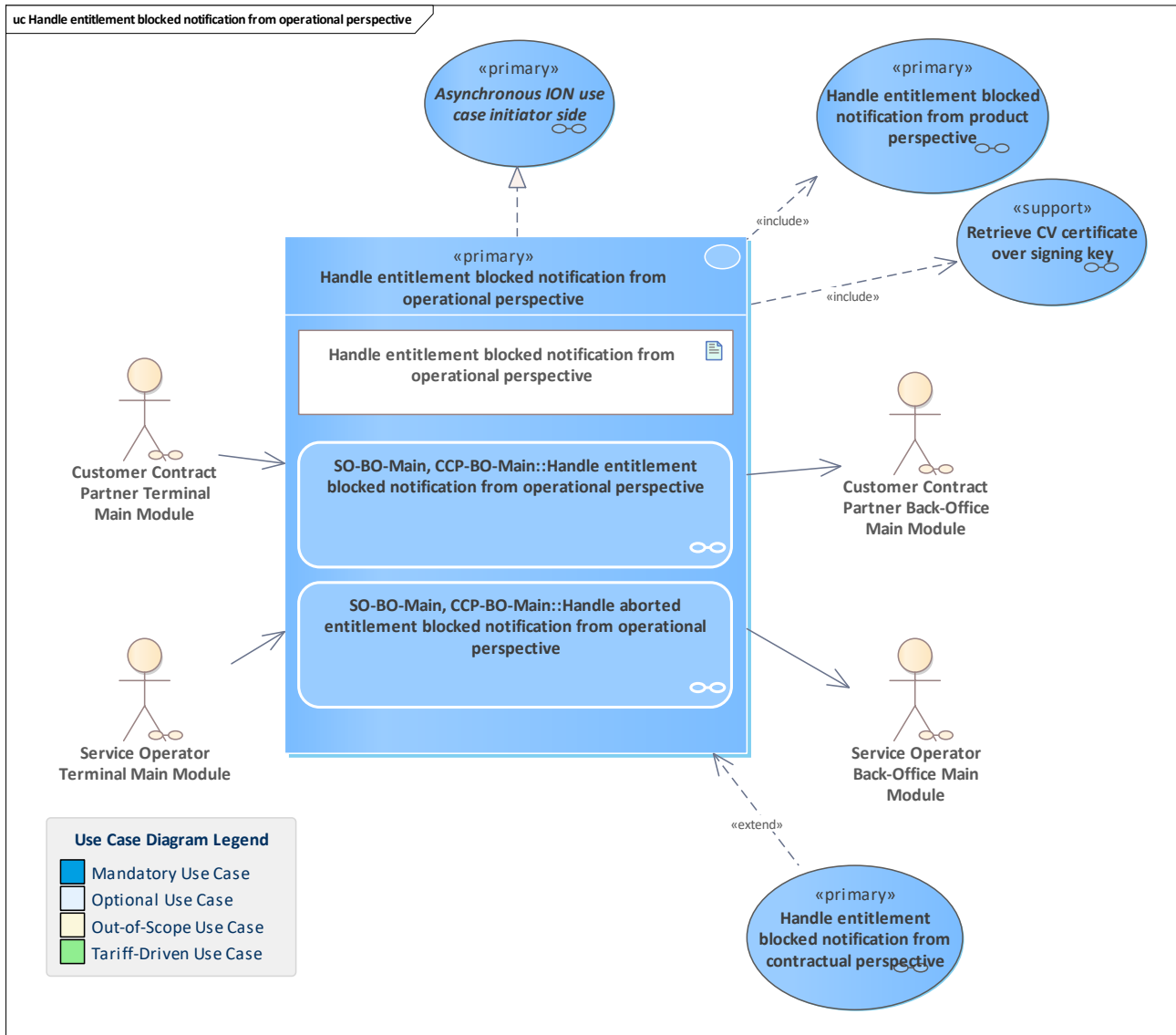
Use Case	<a href="#">Handle application blocked notification from operational perspective</a>
Description	<p>A terminal has blocked the application in a user medium with an application.</p> <p>In this use case, the SO and/or CCP back-office system receives the notification from the terminal and runs certain checks from the issuer perspective, such as the signature verification of the block application attestation.</p> <p>The pCCP of the application is informed if the current operator is a SO or sCCP.</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The SO or the CCP back-office system registers the abortion for internal monitoring.</p>
Initiating Actor	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>
Reacting Actor	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
Preconditions	
Postconditions	





<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle application blocked notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Determine user medium owner</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Application blocked notification from terminal : tNotifyApplicationBlocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle application blocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify application blocking aborted from terminal : tNotifyApplicationBlockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle blocking application aborted notification from operational perspective</a>

## 8.2.4 Handle entitlement blocked notification from operational perspective



<b>Use Case</b>	<a href="#">Handle entitlement blocked notification from operational perspective</a>
<b>Description</b>	<p>The entitlement blocked notification is sent by the terminal to the SO or CCP back-office system. The notification will be checked.</p> <p><u>If the pCCP blocked its own entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>request the hotlist service system to remove the entitlement from the entitlement hotlist.</li> </ul> <p><u>If the sCCP or SO blocked an entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO (and the PO will forward it later to the pCCP)</li> </ul>



	In case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The SO or CCP back-office system registers the abortion for internal monitoring and data consistency.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle entitlement blocked notification from contractual perspective</a> / <a href="#">Handle entitlement blocked notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement blocked notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify entitlement blocked from terminal : tNotifyEntitlementBlocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle entitlement blocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify entitlement blocked aborted from terminal : tNotifyEntitlementBlockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle aborted entitlement blocked notification from operational perspective</a>

## 9 Electronic Ticket Bundle SO-System

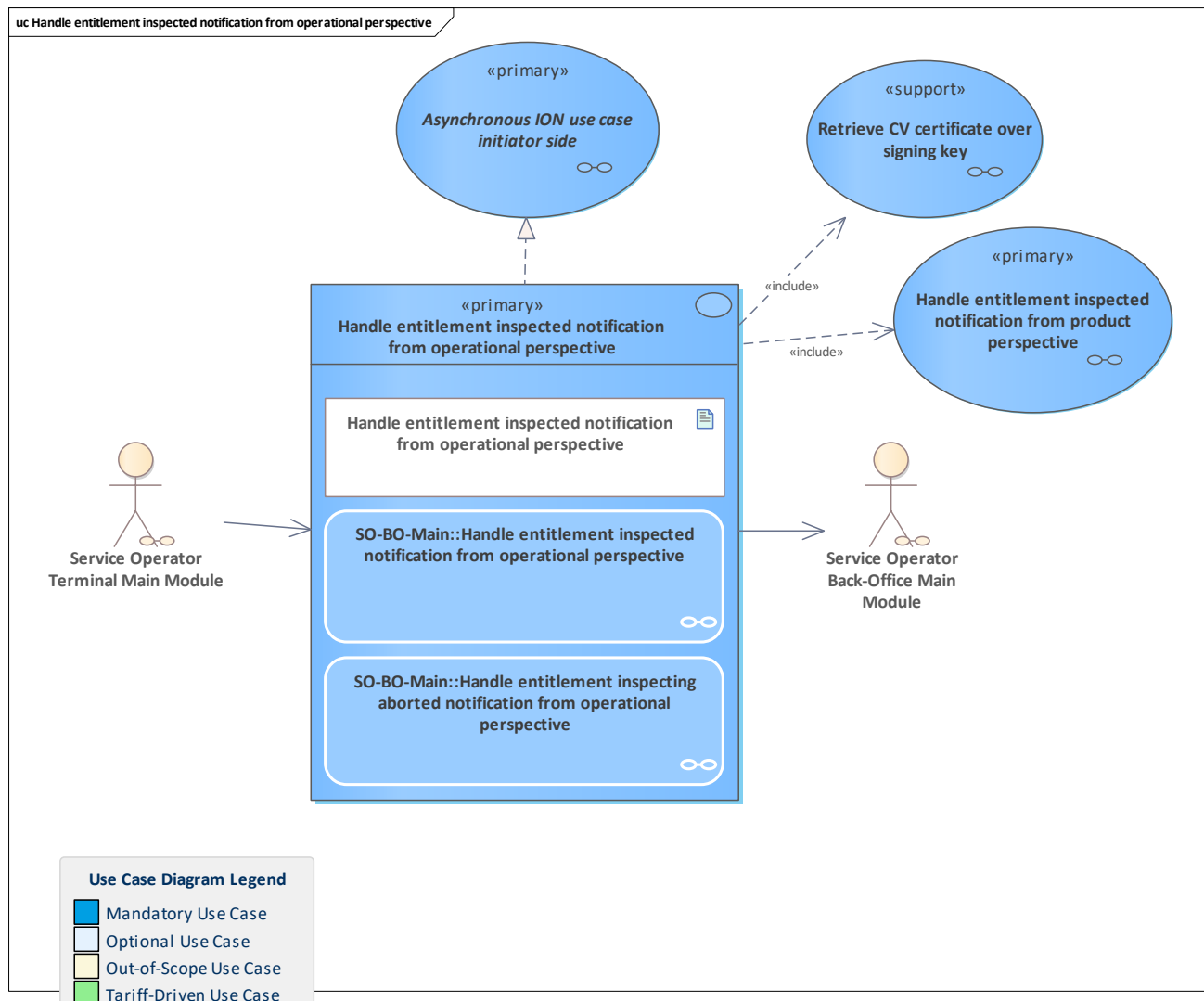
Functionality bundle that covers the use cases for a SO back-office system with electronic tickets placed on a user medium with application (e.g. chip card).

### 9.1 Overview

Handle entitlement inspected notification from operational perspective

### 9.2 Use Cases

#### 9.2.1 Handle entitlement inspected notification from operational perspective



#### Use Case

Handle entitlement inspected notification from operational perspective

<b>Description</b>	This use case describes the processing of the inspection notification received from the SO terminal in the back-office system of the SO. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation. Finally, the notification is forwarded to the PO system. This can be done either directly with a single message or in a scheduled process that sends a list of notifications.
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement inspected notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify entitlement inspecting aborted from terminal : tNotifyEntitlementInspectingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle entitlement inspecting aborted notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify entitlement inspected from terminal : tNotifyEntitlementInspected</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle entitlement inspected notification from operational perspective</a>



## 10 Account-Based Payment Bundle SO-System

No payment method use cases for a SO System.

## 11 Stored-Value Payment Bundle SO-System

No payment method use cases for a SO system.

## 12 Sale Electronic Ticket via Account-Based Payment Bundle SO-System

No payment method use cases for a SO System.

## 13 Sale Electronic Ticket via Stored-Value Payment Bundle SO-System

No payment method use cases for a SO System.

## 14 IN-OUT Bundle SO-System

Functionality bundle that provides SO back-office system use cases for IN-OUT functionality. The term CICO is also used.

### 14.1 Overview

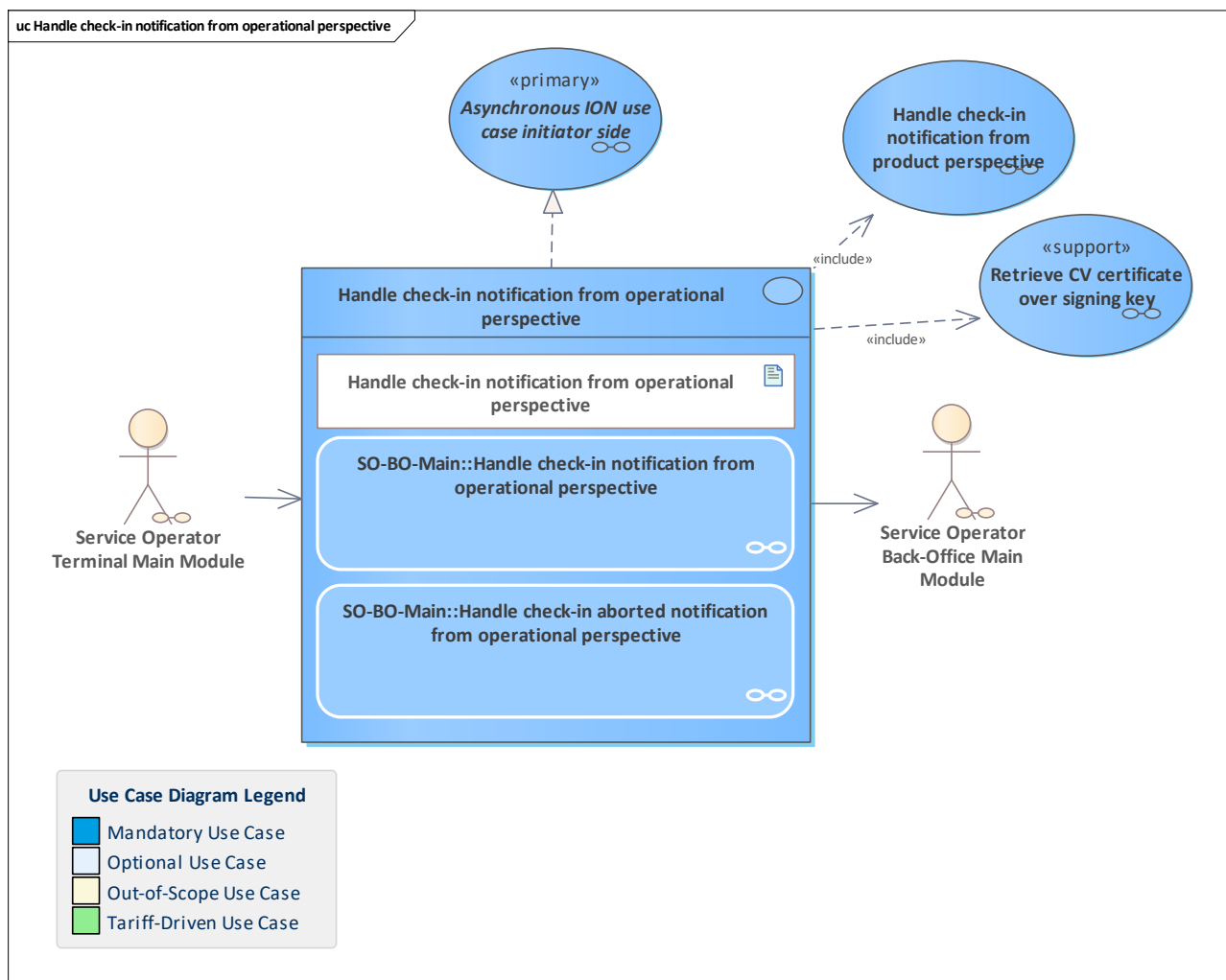
[Handle check-in notification from operational perspective](#)

[Handle check-out notification from operational perspective](#)

[Handle stored-value payment method recharged notification from operational perspective](#)

### 14.2 Use Cases

#### 14.2.1 Handle check-in notification from operational perspective



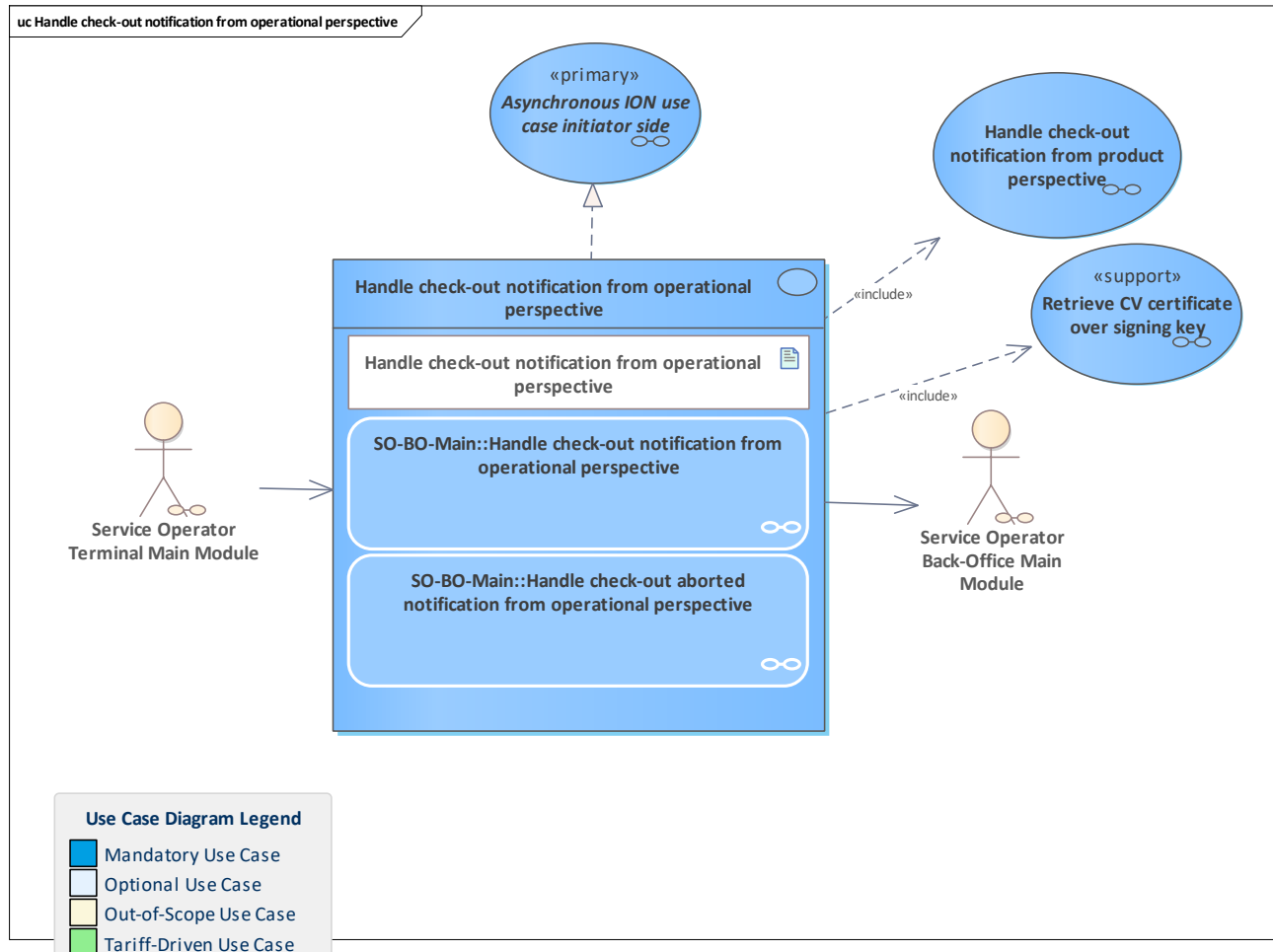
<b>Use Case</b>	<a href="#">Handle check-in notification from operational perspective</a>
<b>Description</b>	<p>Handle an check-in operation from the operational perspective. The check-in notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation. Finally, the notification is forwarded to the PO system. This can be done either directly with a single message or in a scheduled process that sends a list of notifications.</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency. Since no counters are affected which are important for the PO, in case of an abortion, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle check-in notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>



<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify check in from terminal : tNotifyCheckinAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle check-in aborted notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify check in from terminal : tNotifyCheckin</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle check-in notification from operational perspective</a>



## 14.2.2 Handle check-out notification from operational perspective

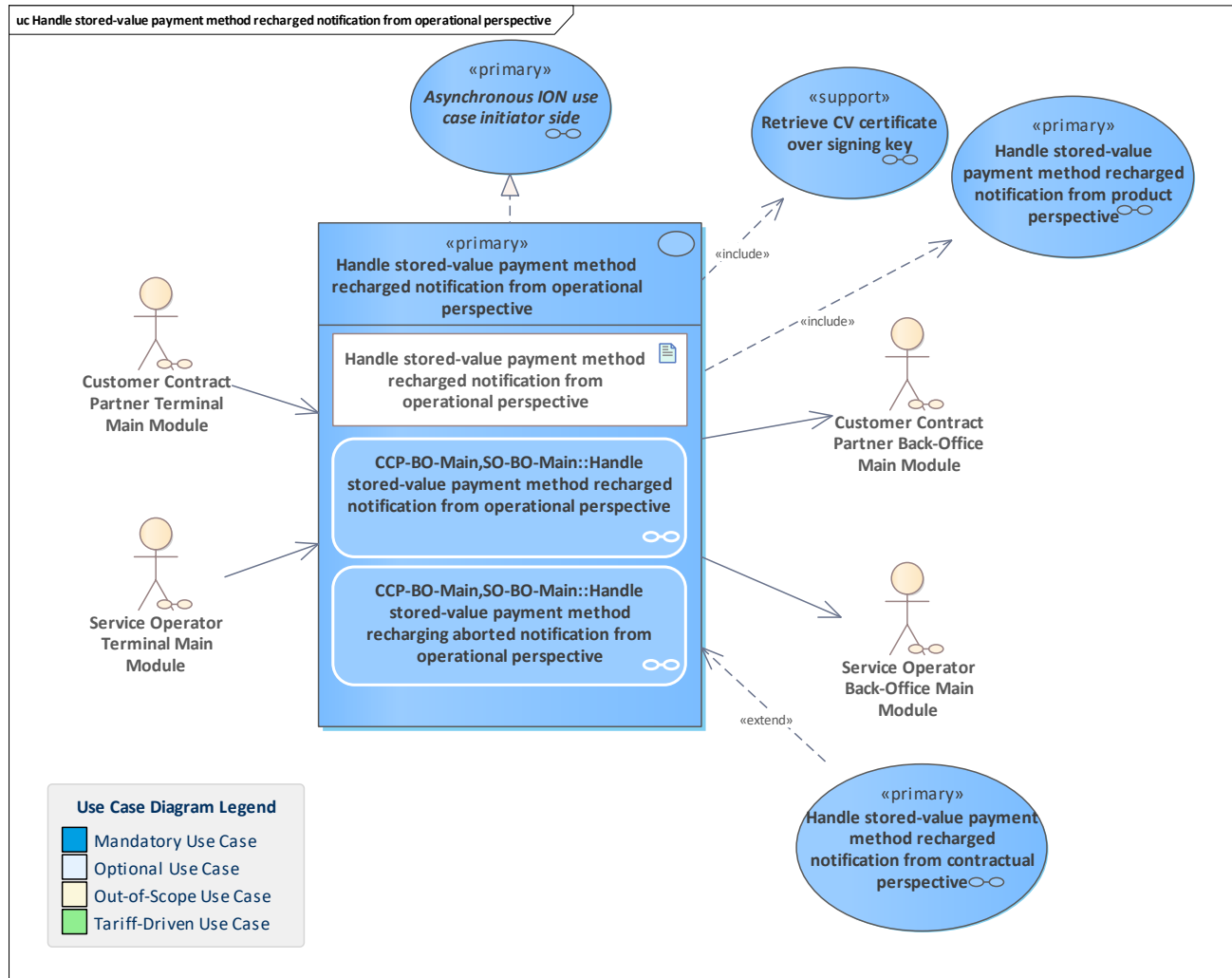


<b>Use Case</b>	<a href="#">Handle check-out notification from operational perspective</a>
<b>Description</b>	<p>Handle an check-out operation from the operational perspective. The check-out notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation. Finally, the notification is forwarded to the PO system. This can be done either directly with a single message or in a scheduled process that sends a list of notifications.</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency. Since no counters are affected which are important for the PO, in case of an abortion, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases</b>	<a href="#">Handle check-out notification from product perspective</a> / <a href="#">Retrieve</a>



<b>(Includes)</b>	<a href="#">CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify check out from terminal : tNotifyCheckoutAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle check-out aborted notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify check out from terminal : tNotifyCheckout</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle check-out notification from operational perspective</a>

## 14.2.3 Handle stored-value payment method recharged notification from operational perspective



<b>Use Case</b>	<a href="#">Handle stored-value payment method recharged notification from operational perspective</a>
<b>Description</b>	<p>Handle a stored-value payment method recharged notification from the operational perspective.</p> <p>The CCP back-office system receives the notification about the recharge action of a stored-value payment method and registers it. Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the recharge action attestation.</p> <p>Then, the notification is forwarded to the responsible PO system. If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In the case of a transaction abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important to the PO, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>



<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle stored-value payment method recharged notification from contractual perspective</a> / <a href="#">Handle stored-value payment method recharged notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle stored-value payment method recharged notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method recharged : tNotifyStoredValuePaymentMethodRecharged</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main,SO-BO-Main::Handle stored-value payment method recharged notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method recharging aborted : tNotifyStoredValuePaymentMethodRechargingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main,SO-BO-Main::Handle stored-value payment method recharging aborted notification from operational perspective</a>



## **15 Ordered Action Management Bundle SO-System**

No use cases in SO back-office system for ordered action management.

## **16 Static Entitlements Bundle SO-System**

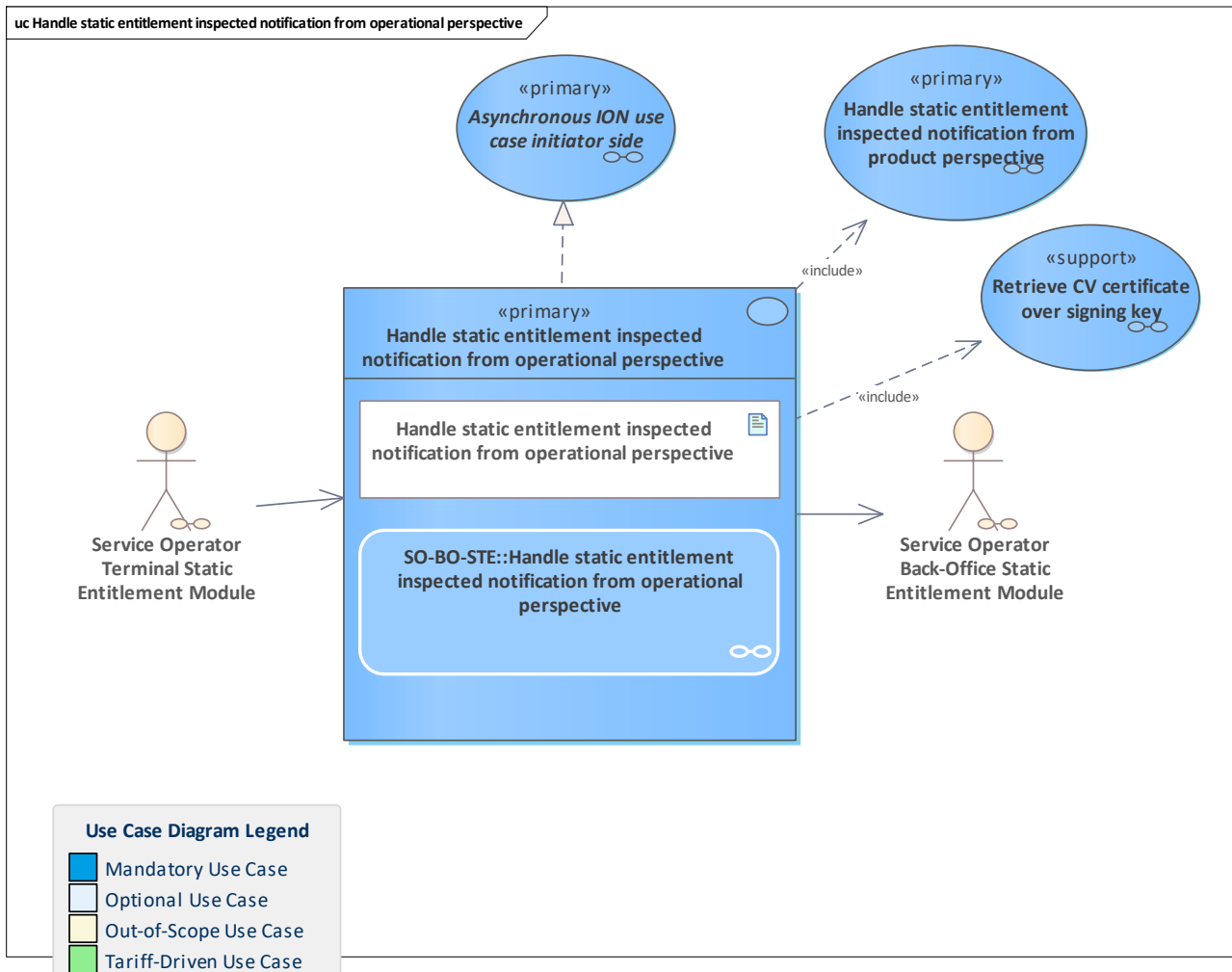
Functionality bundle that covers SO back-office system use cases for working with static entitlements.

### **16.1 Overview**

Handle static entitlement inspected notification from operational perspective

### **16.2 Use Cases**

#### **16.2.1 Handle static entitlement inspected notification from operational perspective**



<b>Use Case</b>	<a href="#">Handle static entitlement inspected notification from operational perspective</a>
<b>Description</b>	Use case for the SO back-office system with static entitlement extension. The notification from the terminal is received and checked from the operational perspective, including the SAM signature verification of the static entitlement. Finally, the notification is sent to the PO back-office system. This can be done either directly with a single message or in a scheduled process that sends a list of notifications.
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Static Entitlement Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Static Entitlement Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle static entitlement inspected notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	



<b>Inputs</b>	<a href="#">Notify static entitlement inspected from terminal :</a> <a href="#">tNotifyStaticEntitlementInspected</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-STE::Handle static entitlement inspected notification from operational perspective</a>

# 17 Miscellaneous Bundle SO-System

Functionality bundle that covers miscellaneous SO back-office system use cases.

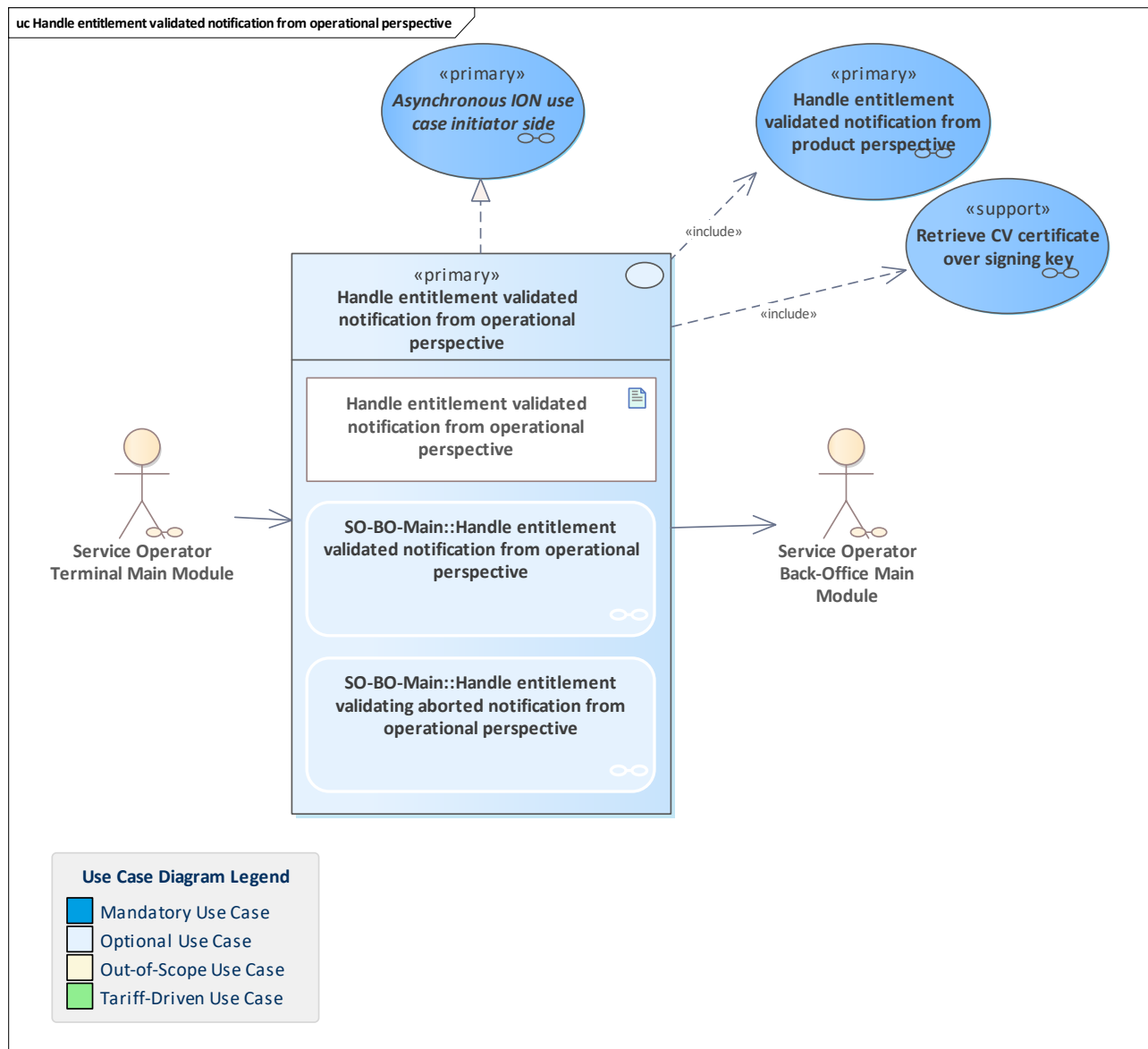
## 17.1 Overview

Optional: Handle entitlement validated notification from operational perspective

Optional: Retrieve valid entitlements for given app instance ID

## 17.2 Use Cases

### 17.2.1 Optional: Handle entitlement validated notification from operational perspective

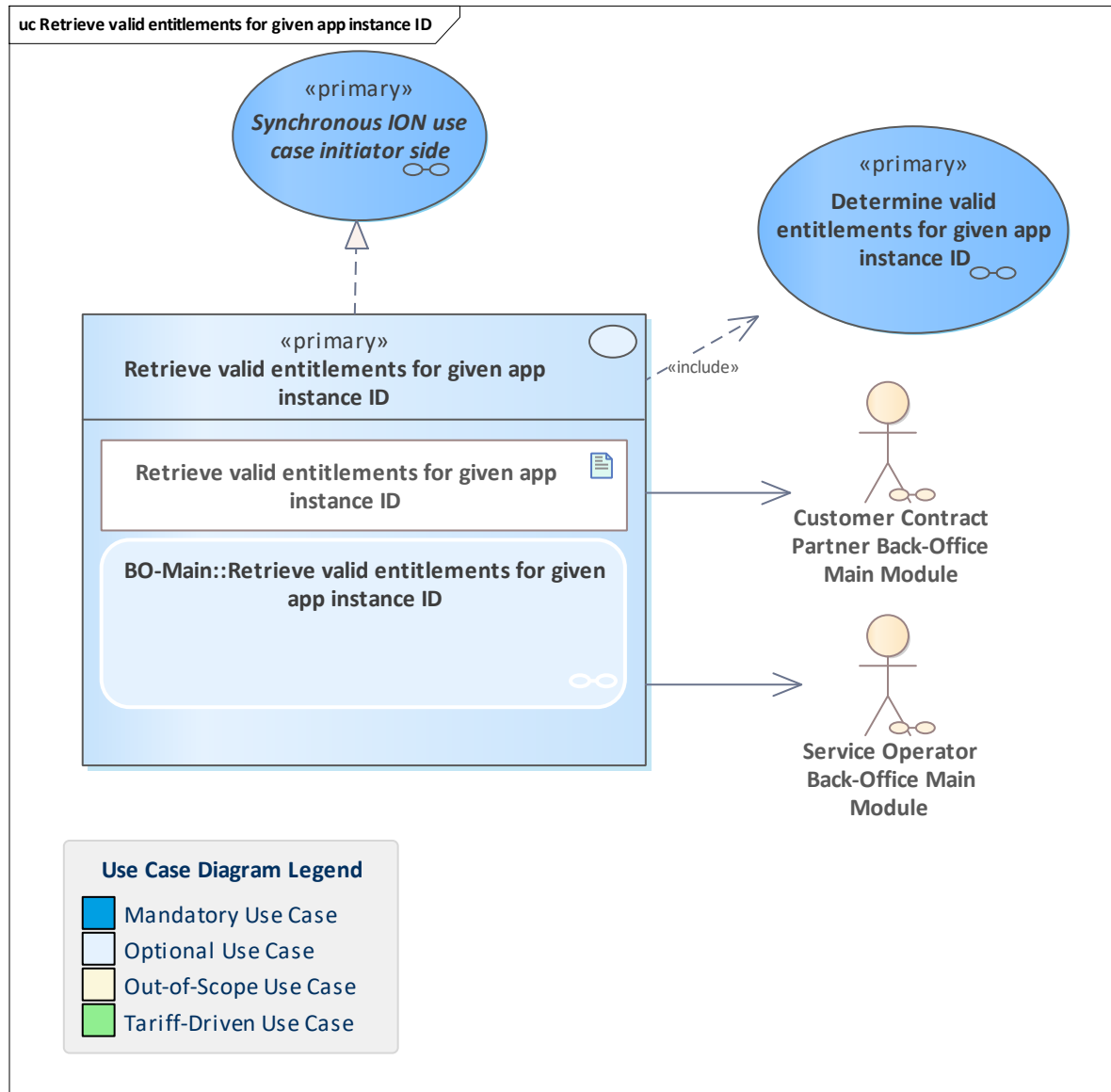






<b>Use Case</b>	<a href="#">Handle entitlement validated notification from operational perspective</a>
<b>Description</b>	<p>Handle an entitlement validated notification from the operational perspective.</p> <p>The entitlement validation notification is sent by the SO terminal to the SO back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p>Finally, the notification will be sent to the PO (and the PO will forward it later to the pCCP)</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.</p>
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement inspected notification from product perspective</a> / <a href="#">Handle entitlement validated notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify entitlement validated from terminal : tNotifyEntitlementValidated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle entitlement validated notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify validation aborted from terminal : tNotifyEntitlementValidatingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main::Handle entitlement validating aborted notification from operational perspective</a>

## 17.2.2 Optional: Retrieve valid entitlements for given app instance ID



Use Case	Retrieve valid entitlements for given app instance ID
Description	<p>The pCCP of the user medium with an application retrieves information about already issued and valid entitlements from each product owner that might come into question by sending the application instance ID.</p> <p>Please note that this is a snapshot of the entitlements valid at a given timestamp.</p> <p>Only the pCCP is informed about application hotlist entries or even an existing blocking of the application. If this use case is used as part of a user medium/entitlement replacement, the pCCP should be the only entity involved.</p> <p>However, this use case can be used by a SO to access entitlement information inside a penalty fare notice process.</p>



	It should be noted that neither the requesting SO nor the PO has information about blocked applications. Likewise, the PO has no information about the hotlist of applications. The requesting SO must check the hotlist of applications.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Determine valid entitlements for given app instance ID</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Valid on : ZonedDate</a> <a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">Complete list of valid entitlements : EntitlementOfGivenUserMedium</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Retrieve valid entitlements for given app instance ID</a>

