

SPEC-CCP-RT

Last changed:	20.11.2025 09:48:18
Version:	3.0.0-rc.6
Creator:	VDV ETS

Table of Contents

1	Einführung	8
2	Customer Contract Partner Terminal	8
2.1	Customer Contract Partner Terminal	10
2.2	Customer Contract Partner Terminal Main Module	10
2.3	Customer Contract Partner Terminal Order Execution Module	11
2.4	Customer Contract Partner Terminal Static Entitlement Module	11
3	General Specification Parts for Terminals	11
3.1	Basic Concepts	11
3.1.1	Session	11
3.1.2	Transaction	12
3.1.3	Shadow copy of SAM and UM state	12
3.1.4	Notification categorisation	12
3.2	Technical Requirements	15
3.2.1	Communication interfaces	15
3.2.2	Declarations of conformity	16
3.2.3	Use of SAMs	17
3.2.4	Contactless communication	18
3.2.5	Re-certification of terminals	19
3.2.6	Password entry	20
3.2.7	Behaviour of the terminal in the event of unforeseen circumstances	21
3.2.8	Additional requirements	22
3.2.9	Further notes	23
4	Labelling of terminal devices	23
4.1	Smart Card Reader according to ISO/IEC 14443	23
4.2	Smart Card Reader according to EMVco	24
5	Notification Process Patterns	25
5.1	Supporting activities	27
5.1.1	Perform transaction to XY	28
5.1.2	Prepare XY parameters	31
5.2	Supporting classes	32
5.2.1	Action parameters	32
5.2.2	XY parameters	32
5.2.3	SignedXYAttestation	32
5.2.4	Notification parameters	32
5.2.5	XYNotification	33
5.2.6	ForwardedXYNotification	33
5.2.7	tNotifyXY	33

5.2.8	tNotifyXYAborted	33
5.2.9	notifyXY	33
5.2.10	notifyXYResponse	33
5.2.11	notifyXYException	33
5.2.12	forwardXYNotification	33
5.2.13	forwardXYNotificationResponse	33
5.2.14	forwardXYNotificationException	33
5.3	Perform entitlement XY and notify	33
5.3.1	Perform entitlement XY and notify	34
5.3.2	T-Module::Perform entitlement XY and notify	35
5.3.3	Notify entitlement XY	36
5.3.4	Notify entitlement XY aborted	37
5.3.5	Notify entitlement XY aborted based on attestation	37
5.3.6	Notify XY (entitlement owned)	38
5.3.7	Notify XY (entitlement non-owned)	39
5.4	Perform application XY and notify	39
5.4.1	Perform application XY and notify	40
5.4.2	T-Module::Perform application XY and notify	40
5.4.3	Notify application XY	42
5.4.4	Notify application XY aborted	42
5.4.5	Notify XY (application owned)	43
5.4.6	Notify XY (application non-owned)	44
5.5	Handle entitlement XY notification from operational perspective	44
5.5.1	Handle entitlement XY notification from operational perspective	45
5.5.2	Role-BO-Module::Handle entitlement XY notification from operational perspective	45
5.5.3	Check and process entitlement XY notification from operational perspective	46
5.5.4	Role-BO-Module::Handle entitlement XY aborted notification from operational perspective	47
5.6	Handle application XY notification from operational perspective	48
5.6.1	Handle application XY notification from operational perspective	49
5.6.2	Role-BO-Module::Handle application XY notification from operational perspective	49
5.6.3	Check and process application XY notification from operational perspective	50
5.6.4	Role-BO-Module::Handle application XY aborted notification from operation perspective	52
5.7	Handle entitlement XY notification from product perspective	52
5.7.1	Handle entitlement XY notification from product perspective	53
5.7.2	PO-BO-Module::Handle entitlement XY notification from product perspective	53
5.7.3	Check and process entitlement XY notification from product perspective	55
5.7.4	Conditionally forward notification to pCCP	55
5.8	Handle entitlement XY notification from contractual perspective	56
5.8.1	Handle entitlement XY notification from contractual perspective	57
5.8.2	pCCP-BO-Module::Handle entitlement XY notification from contractual perspective	57
5.8.3	Check and process entitlement XY notification from contractual perspective	58

5.8.4	pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification	59
5.9	Handle application XY notification from contractual perspective	60
5.9.1	Handle application XY notification from contractual perspective	61
5.9.2	pCCP-BO-Module::Handle application XY notification from contractual perspective	61
5.9.3	Check and process application XY notification from contractual perspective	62
5.9.4	pCCP-BO-Module::Perform specific contractual tasks on application XY notification	63
6	Basic Bundle Terminal - Foundation	64
6.1	Overview	64
6.2	Use Cases	64
6.2.1	Terminal startup procedure	64
6.2.2	Update SAM configuration	67
6.2.3	Optional: Update SAM reset data	69
6.2.4	Update terminal hotlists	71
6.2.5	Update organisation list	73
6.2.6	Update tariff module	74
6.2.7	Update CA certificate repository	76
6.2.8	Update CV certificate revocation list	77
7	Basic Bundle Terminal - UM with Application	79
7.1	Overview	79
7.2	Use Cases	79
7.2.1	Get entitlement and check attestations	79
7.2.2	Establish session based on certificates	82
7.2.3	Establish session and get entitlement directory	84
7.2.4	Export derived key	86
7.2.5	Perform application blocking and notify	88
7.2.6	Perform entitlement blocking and notify	90
7.2.7	Optional: Log defective user medium with application	92
8	Basic Bundle Terminal - Extended Logging	94
8.1	Overview	94
8.2	Use Cases	94
8.2.1	Optional: Create extended logging for an application	94
8.2.2	Optional: Create extended logging for an entitlement	96
9	Basic Bundle CCP-Terminal - Foundation	98
9.1	Overview	98
9.2	Use Cases	98
9.2.1	Activate SAM	98
10	Basic Bundle CCP-Terminal - UM with Application	101
10.1	Overview	101
10.2	Use Cases	101
10.2.1	Check user medium with application as CCP	101

10.2.2	Delete entitlement	103
10.2.3	Display application data	104
10.2.4	Display entitlement	105
11	Basic Bundle CCP-Terminal - UM in Customer Center	107
11.1	Overview	107
11.2	Use Cases	107
11.2.1	Unblock application	107
11.2.2	Perform application unblocking and notify	110
11.2.3	Unblock entitlement	111
11.2.4	Perform entitlement unblocking and notify	113
11.2.5	Optional: Take back application	115
11.2.6	Optional: Perform application termination and notify	117
11.2.7	Optional: Configure user medium application	119
11.2.8	Optional: Exchange user medium with application	120
12	Basic Bundle CCP-Terminal - UM with Customer Data	122
12.1	Overview	122
12.2	Use Cases	122
12.2.1	Initialise User Medium with application for customer	122
12.2.2	Read information from user medium with application	125
12.2.3	Write customer	127
12.2.4	Delete customer	128
12.2.5	Display customer	129
12.2.6	Write discounts	131
12.2.7	Delete discounts	133
12.2.8	Read customer and discounts	134
12.2.9	Display discounts	136
12.2.10	Change customer and discounts	137
12.2.11	De-personalise application	139
13	Basic Bundle CCP-Terminal - UM with Password	141
13.1	Overview	141
13.2	Use Cases	141
13.2.1	Initialise password	141
13.2.2	Write password configuration	143
13.2.3	Change password	144
13.2.4	Verify password	146
14	Electronic Ticket Bundle CCP-Terminal	147
14.1	Overview	147
14.2	Use Cases	147
14.2.1	Issue entitlement	147
14.2.2	Perform entitlement issuance and notify	150
14.2.3	Take back entitlement	152
14.2.4	Perform entitlement termination and notify	154

14.2.5	Reimburse and terminate electronic ticket	156
14.2.6	Change entitlement	158
14.2.7	Display entitlement	160
14.2.8	Optional: Save electronic ticket as favourite	162
14.2.9	Optional: Write favourites	164
14.2.10	Optional: Change favourites	165
14.2.11	Optional: Delete favourites	167
14.2.12	Optional: Display favourites	168
14.2.13	Optional: Print customer receipt	170
15	Account-Based Payment Bundle CCP-Terminal	171
15.1	Overview	171
15.2	Use Cases	171
15.2.1	Issue entitlement	171
15.2.2	Perform entitlement issuance and notify	174
15.2.3	Take back entitlement	176
15.2.4	Perform entitlement termination and notify	178
15.2.5	Change entitlement	180
15.2.6	Display entitlement	182
15.2.7	Optional: Reimburse and terminate account-based payment method	184
16	Stored-Value Payment Bundle CCP-Terminal	186
16.1	Overview	186
16.2	Use Cases	186
16.2.1	Recharge stored-value payment method	186
16.2.2	Perform stored-value payment method recharging and notify	188
16.2.3	Optional: Autoload stored-value payment method	190
16.2.4	Reimburse stored-value payment method	192
16.2.5	Perform stored-value payment method reimbursing and notify	194
16.2.6	Reimburse and terminate stored-value payment method	196
16.2.7	Change entitlement	198
16.2.8	Display entitlement	200
17	Sale Electronic Ticket via Account-Based Payment Bundle CCP-Terminal	202
17.1	Overview	202
17.2	Use Cases	202
17.2.1	Sell electronic ticket using account-based payment method	202
17.2.2	Optional: Sell static entitlement using account-based payment method	205
17.2.3	Debit account-based payment method	207
17.2.4	Perform account-based payment method debiting and notify	209
17.2.5	Credit account-based payment method	211
17.2.6	Perform account-based payment method crediting and notify	213
18	Sale Electronic Ticket via Stored-Value Payment Bundle CCP-Terminal	215
18.1	Overview	215
18.2	Use Cases	215

18.2.1	Sell electronic ticket using stored-value payment method	215
18.2.2	Optional: Sell static entitlement using stored-value payment method	218
18.2.3	Debit stored-value payment method	220
18.2.4	Perform stored-value payment method debiting and notify	222
18.2.5	Credit stored-value payment method	224
18.2.6	Perform stored-value payment method crediting and notify	226
19	IN-OUT Bundle CCP-Terminal	228
19.1	Overview	228
19.2	Use Cases	228
19.2.1	Optional: Change user tariff parameters	228
19.2.2	Optional: Perform user tariff parameters change and notify	230
20	Ordered Action Management Bundle Executing CCP-Terminal	232
20.1	Overview	232
20.2	Use Cases	232
20.2.1	Execute action list entries	232
20.2.2	Issue entitlement triggered by action order	235
20.2.3	Perform ordered entitlement issuance and notify	237
20.2.4	Unblock entitlement triggered by action order	239
20.2.5	Perform ordered entitlement unblocking and notify	241
20.2.6	Terminate entitlement triggered by action order	243
20.2.7	Perform ordered entitlement termination and notify	245
20.2.8	Block entitlement triggered by action order	247
20.2.9	Perform ordered entitlement blocking and notify	249
20.2.10	Update terminal action list	251
21	Static Entitlements Bundle CCP-Terminal	252
21.1	Overview	252
21.2	Use Cases	252
21.2.1	Check user medium without application	252
21.2.2	Issue static entitlement	255
21.2.3	Authorise static entitlement	257
21.2.4	Take back static entitlement	259
21.2.5	Reimburse and terminate static entitlement	261
21.2.6	Notify static entitlement terminated	263
21.2.7	Change static entitlement	264
21.2.8	Display static entitlement	266
21.2.9	Optional: Print customer receipt	268
22	Miscellaneous Bundle CCP-Terminal	269
22.1	Overview	269
22.2	Use Cases	269
22.2.1	Optional: Reissue entitlements	269

1 Einführung

The customer contract partner terminal handles on-site tasks relating to customer interaction. For example, it is responsible for issuing or taking back entitlements and for interacting with payment methods. All interactions with a user medium are reported to the relevant back-office system for verification and monitoring purposes.

This reference specification contains all functionality bundles and use cases for a customer contract partner terminal.

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

For a fully functional customer contract partner terminal, all basic packages must be implemented. There are separate specifications for customer contract partner terminals with limited functionality.

2 Customer Contract Partner Terminal

This chapter contains all components and interfaces that are involved with the [Customer Contract Partner Terminal](#). Depending on the deployment variant, not all components and interfaces are required.

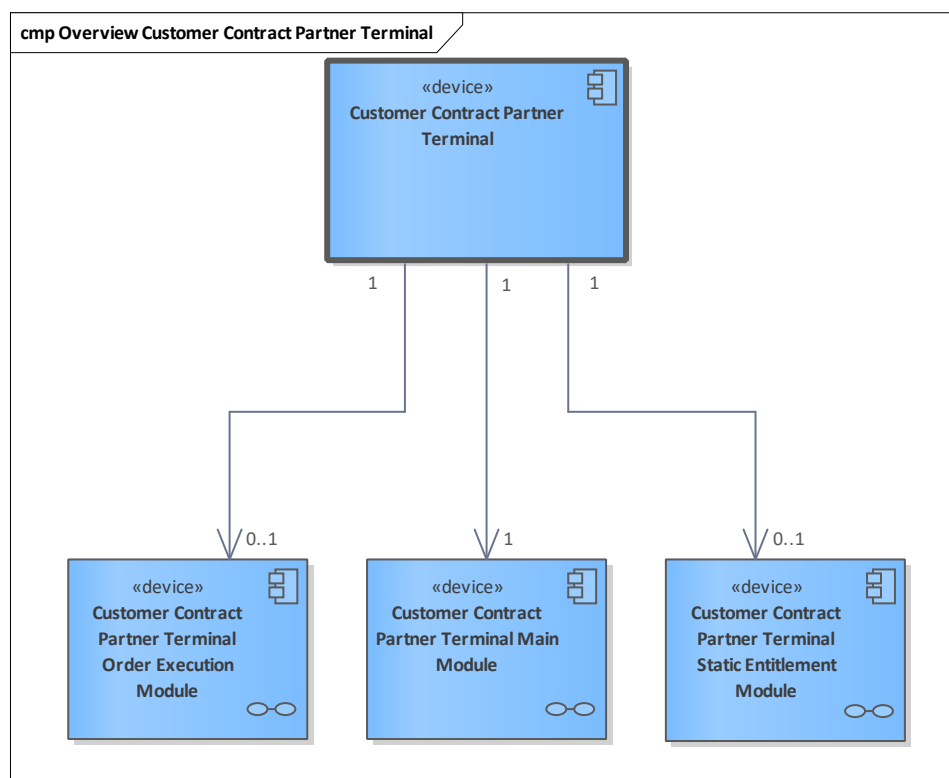


Figure 1: Overview Customer Contract Partner Terminal
Shows the composition of a [Customer Contract Partner Terminal](#).

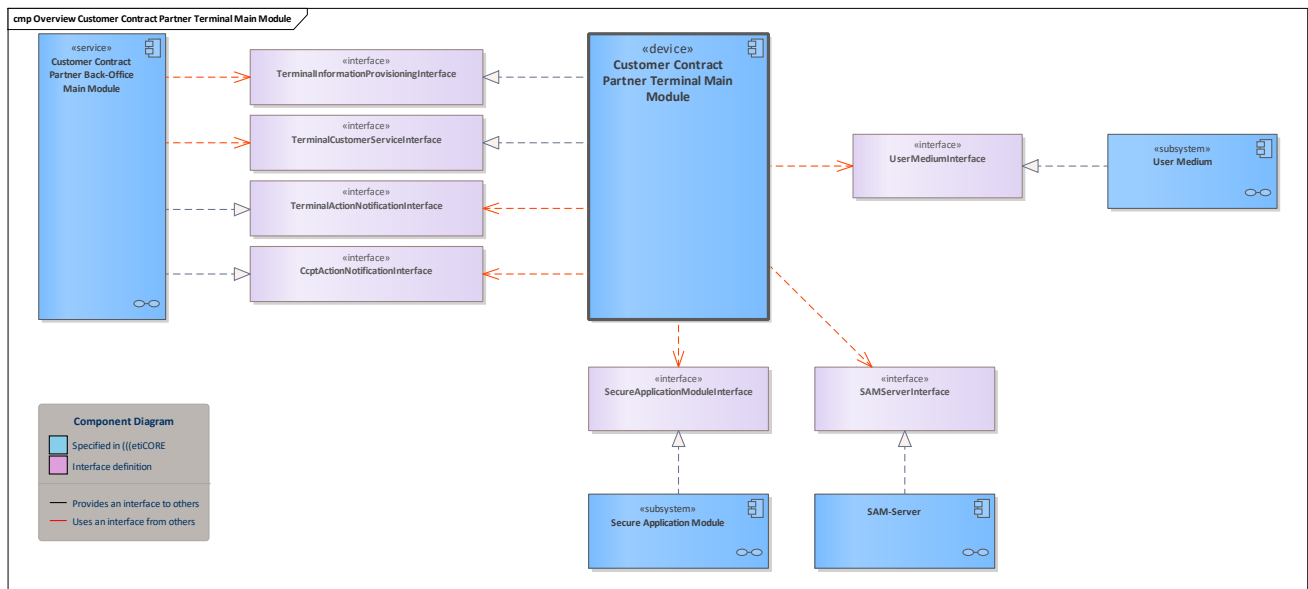


Figure 2: Overview Customer Contract Partner Terminal Main Module

Shows the interaction of a [Customer Contract Partner Terminal Main Module](#) via interfaces with other components.

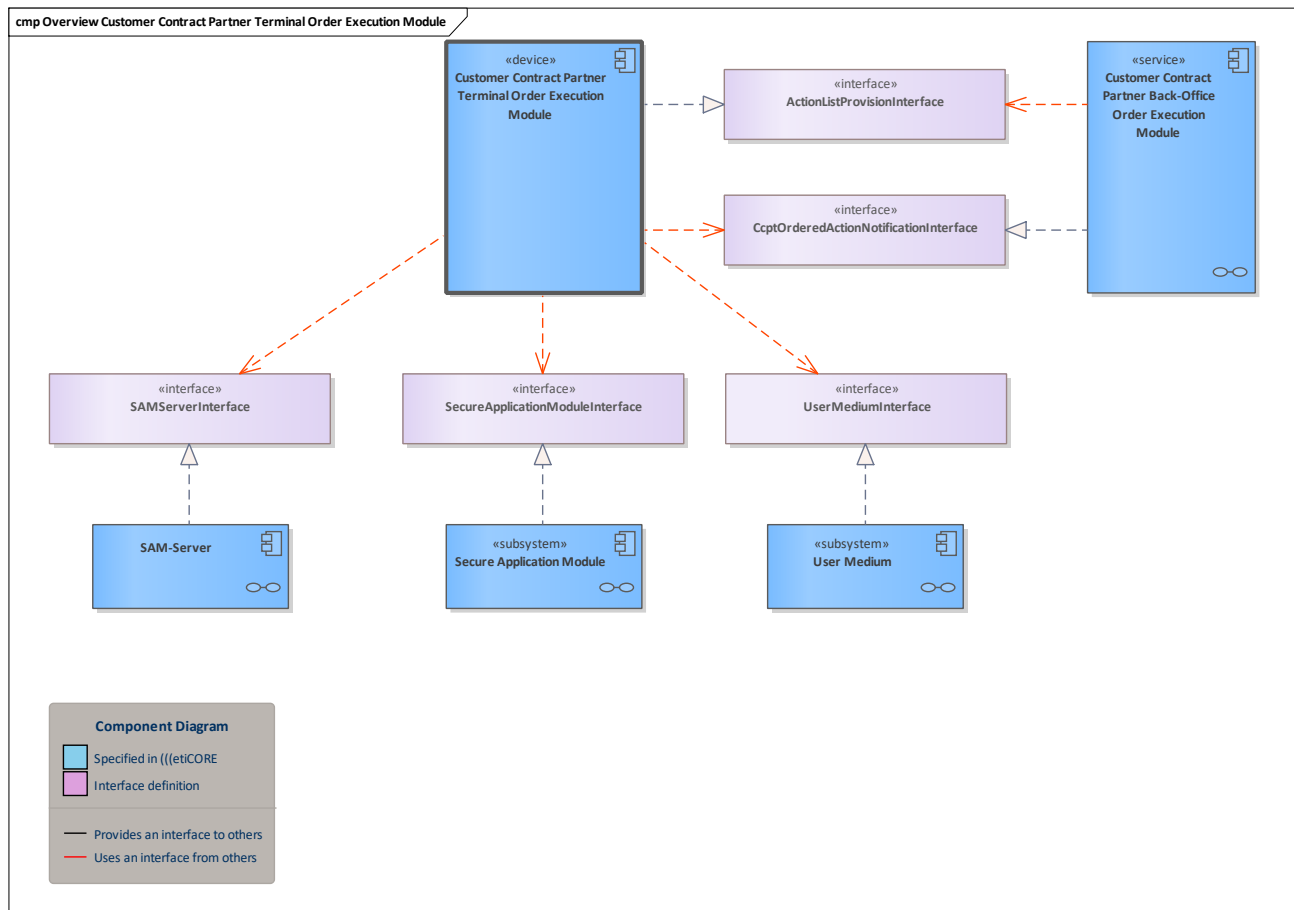


Figure 3: Overview Customer Contract Partner Terminal Order Execution Module

Shows the interaction of a [Customer Contract Partner Terminal Order Execution Module](#) via interfaces with other components.

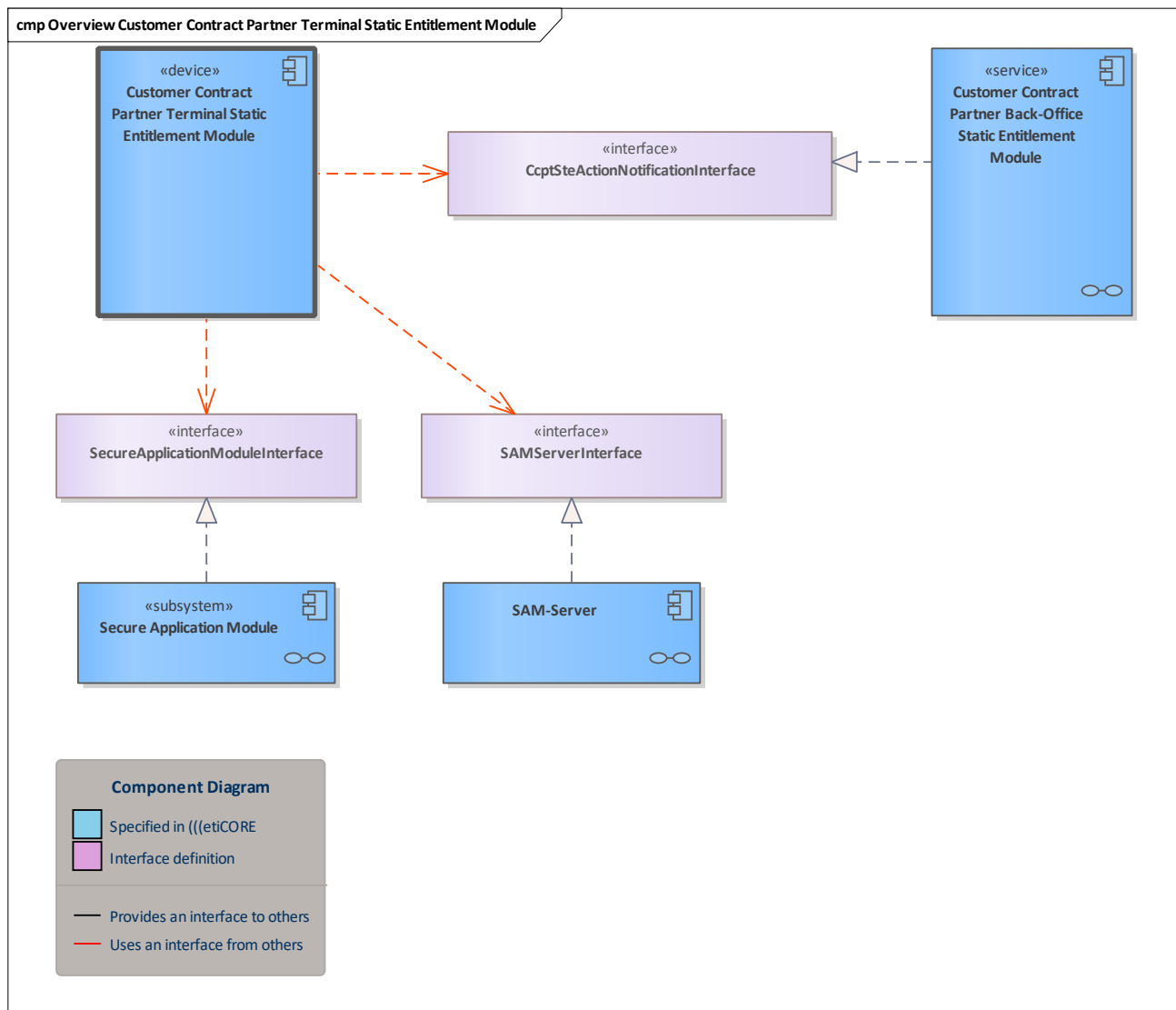


Figure 4: Overview Customer Contract Partner Terminal Static Entitlement Module

Shows the interaction of a [Customer Contract Partner Terminal Static Entitlement Module](#) via interfaces with other components.

2.1 Customer Contract Partner Terminal

Component for a customer contract partner terminal. Depending on the implemented functionality bundles, the customer contract partner terminal consists of

- [Customer Contract Partner Terminal Main Module](#) (always)
- [Customer Contract Partner Terminal Order Execution Module](#) (if the terminal can execute actions on user media via action lists)
- [Customer Contract Partner Terminal Static Entitlement Module](#) (if the terminal can work with static entitlements)

2.2 Customer Contract Partner Terminal Main Module



The Customer Contract Partner Terminal Main Module offers the basic functionality and is operated by the [Customer Contract Partner](#). It provides the basic functionality of sales and information terminals.

2.3 Customer Contract Partner Terminal Order Execution Module

Component which adds the necessary functionality for the [Customer Contract Partner Terminal](#) to execute actions on user media via action lists.

2.4 Customer Contract Partner Terminal Static Entitlement Module

Component which adds the necessary functionality for the [Customer Contract Partner Terminal](#) to check, pay, issue, terminate, etc. static entitlements.

3 General Specification Parts for Terminals

Terminals in etiCORE are employed for purchasing electronic tickets on one hand and for recording and inspection purposes on the other hand.

A terminal usually employs a SAM for secure messaging if data from a user medium has to be read or data has to be written to it. The terminal controls these processes.

3.1 Basic Concepts

This chapter describes the basic concepts of a terminal and describes sessions, transactions and data caching.

3.1.1 Session

A secure messaging session (or session for short) is a secure communications channel between two instances of the etiCORE security concept, e.g. between the SAM and the user medium or between the SAM and the SAM management system.

A session is a prerequisite of many functions of SAM and User Medium, e.g. for entitlement issuance. A session can be set up based on master keys (symmetric encryption) or based on certificates (asymmetric encryption).

The core of the master key-based session establishment is shown in Railroad diagram: "master key-based session establishment".

Certificate-based session establishment is explained and shown in [Establish session based on certificates](#).

Sessions are ended by resetting one of the securely communicating devices (i.e. de-energise or call SELECT) or when one of the devices responds with code 0x69 88 ('*SM data objects incorrect*').



Railroad diagram: master key-based session establishment

3.1.2 Transaction

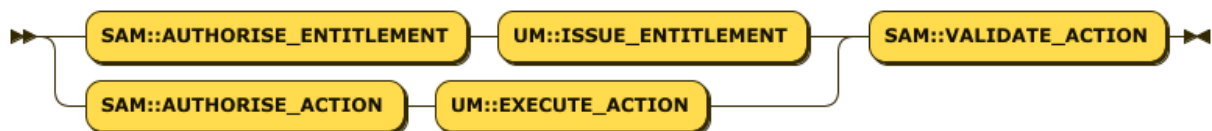
A transaction is a sequence of operations. Embedding operations into a transaction guarantees that either all operations are performed or no operation is performed (atomicity). In etiCORE, the user medium operations [UM::ISSUE_ENTITLEMENT](#) and [UM::EXECUTE_ACTION](#) are always performed within a transaction that is finalised by the operation [UM::COMMIT_TRANSACTION](#). Successfully returning from a [UM::COMMIT_TRANSACTION](#) call indicates that all operations part of the transaction were successfully performed. However, to ensure the authenticity of that success, the response from the user medium needs to be validated by the SAM via [SAM::UNWRAP_RAPDU](#).

In the diagrams Railroad diagram: operation execution and Railroad diagram: transaction, an overview of the allowed sequence of operations during a transaction is shown. The interweaving of the SAM operations shown is necessary from a security standpoint but has nothing to do with the transaction itself. There are no transactional mechanisms in the SAM.

In etiCORE, the transaction concept is important primarily for three situations:

- Issuing an electronic ticket and paying for it with an ePaymentMethod
- Terminating an electronic ticket and granting a refund by crediting an ePaymentMethod
- Terminating an entitlement and issuing a replacement entitlement

For all other actions specified so far, it is recommended to isolate them in separate transactions and commit them one by one to avoid unnecessary memory pressure for the user medium.



Railroad diagram: operation-execution



Railroad diagram: transaction

3.1.3 Shadow copy of SAM and UM state

The terminal keeps a shadow copy of the state of the SAM and the user medium. This is helpful for notifications about aborted actions, which contain values of counters that the terminal would otherwise need to read post-mortem. It additionally enables the terminal to assess the feasibility of use cases.

This shadow copy is kept in different data stored in the package [Terminal](#).

These are read in many situations as needed and written to whenever the corresponding value in the SAM / user medium changes as a result of a command (and initially during [Check user medium](#) and [Terminal startup procedure](#)).

3.1.4 Notification categorisation

Before diving into the details of the notification categories given below, consider familiarising yourself with the [Notification process patterns](#).



Notification	Category	Terminal
EntitlementIssuedNotification	Entitlement owned	CCP-T
EntitlementTerminatedNotification	Entitlement owned and non-owned	CCP-T
EntitlementIssuingAbortedNotification	Entitlement owned	CCP-T
EntitlementBlockedNotification	Entitlement owned and non-owned	T
EntitlementUnblockedNotification	Entitlement owned	CCP-T
EntitlementInspectedNotification	Entitlement non-owned	SO-T
EntitlementValidatedNotification	Entitlement non-owned	SO-T
CheckinNotification	Entitlement non-owned	SO-T
CheckoutNotification	Entitlement non-owned	SO-T
UserTariffParametersChangedNotification	Entitlement owned and non-owned	CCP-T
StoredValuePaymentMethodRechargedNotification	Entitlement owned and non-owned	T
StoredValuePaymentMethodReimbursedNotification	Entitlement owned	CCP-T
StoredValuePaymentMethodDebitedNotification	Entitlement owned and non-owned	CCP-T
StoredValuePaymentMethodCreditedNotification	Entitlement owned and non-owned	CCP-T
AccountBasedPaymentMethodDebitedNotification	Entitlement owned and non-owned	CCP-T
AccountBasedPaymentMethodCreditedNotification	Entitlement owned and non-owned	CCP-T
OrderedEntitlementIssuedNotification	Entitlement owned and non-owned	CCP-T
OrderedEntitlementTerminatedNotification	Entitlement owned and non-owned	CCP-T
OrderedEntitlementUnblockedNotification	Entitlement owned and non-owned	CCP-T
OrderedEntitlementBlockedNotification	Entitlement owned and non-owned	CCP-T
ApplicationTerminatedNotification	Application owned	CCP-T
ApplicationBlockedNotification	Application owned and non-owned	T
ApplicationUnblockedNotification	Application owned	CCP-T
StaticEntitlementInspectedNotification	Entitlement non-owned	SO-T
StaticEntitlementIssuedNotification	Entitlement owned	CCP-T
StaticEntitlementIssuingAbortedNotification	Entitlement owned	CCP-T
StaticEntitlementTerminatedNotification	Entitlement owned and non-owned	CCP-T



Legend for Terminal:

- T is any terminal
- SO-T is an SO terminal
- CCP-T is a CCP terminal



3.2 Technical Requirements

The aim of this section is to describe the consolidated requirements for all terminals involved in etiCORE.

Parts marked with "Note" are not relevant for certification or are not part of the specification. All terminals require a certificate from the VDV-ETS accredited test laboratory to prove their etiCORE functionality.

3.2.1 Communication interfaces

The terminal manufacturer must ensure that data exchange with the corresponding background system can take place securely, in accordance with the etiCORE interface specification.

The terminals must have suitable interfaces for the provisioning and removal of data to/from their back-office system.

If the execution unit is operated separately from the control unit, the etiCORE SPEC-EUI additionally applies to this interface.



3.2.2 Declarations of conformity

The terminal must have a certificate of conformity for type testing in accordance with the Law on the Electromagnetic Compatibility of Equipment (EMVG) and the resulting valid standards.

Note: The contactless technology in the user medium, together with the corresponding terminals, forms an inductive radio system that must comply with both national and international approval regulations.

The manufacturer must provide an EU Declaration of Conformity according to Article 18 of the RE Directive (Directive 2014/53/EU) as per the valid Official Journal, which for RFID components includes at least the following points:

- ETSI EN 300 330 (RF)
- ETSI EN 301 489 -1 and -3 (EMC)
- EN 60950 (until 2019) or 62368 (safety requirements).



3.2.3 Use of SAMs

The terminal must provide the Installation of hardware SAMs, unless a Light-SAM is used exclusively.

The terminal requires at least 2 slots to plug in hardware SAMs. The terminal software must be able to control the SAMs appropriately.

Tamper protection according to ISO 13491 (requirements for tamper-resistant devices) must be guaranteed

- particularly: voltage and, thus, data loss when removing the SAM



3.2.4 Contactless communication

For contactless communication with a proximity technology user medium in accordance with ISO/IEC 14443, the following also applies:

Presence of a reading device that meets all requirements according to ISO/IEC 14443, specifically:

- communication interfaces must be provided for both type A and type B media.
- **Note:** any medium must be supported that uses any (default) parameters (especially time parameters) within the range permitted by the standard.
- the minimum reading range is 1cm
- Problems with several media in the reading field should be avoided via collision detection

Note: The user medium may send WTX commands during the start phase (for example, to carry out hardware tests for safety checks). The terminal must respect these. The values to be expected can be taken from the data sheet accompanying the etiCORE SPEC UM.

Note: Regarding the evaluation, if communication errors occur, the behaviour of the ISO-compliant card reader is to be ensured as follows:

- ◆ After successfully establishing a connection (anti-collision), the card reader automatically tries to read the card. Due to the increased card energy consumption (e.g. due to card-side hardware checks when booting the user medium), connections may be interrupted in the outermost field range. An ISO-compliant card reader must not attempt to reconnect during the WTX time window. A WTX timeout ensures that the terminal can make a new connection request in a timely manner.
- ◆ In this case, the card reader should try to re-establish communication and run the application.
- ◆ Only then should the processing by the terminal/reader be aborted with an error message if no connection could be established.
- ◆
- ◆

The reader must have an authenticated certificate from an accredited certification body proving standard-compliant implementation in accordance with ISO/IEC 14443.



3.2.5 Re-certification of terminals

Terminals that have already been certified require re-certification if any of the following product characteristics change:

- the terminal identification
- the NM reader type (if available)
- the NM reader's installation position
- the SAM interface (if available)
- the etiCORE version and functional enhancements following a release update (use cases)

If an error occurs in the operation of previously certified terminals, a retest is required. During the retest, the basic etiCORE functionality is always tested again.

For software updates that do not affect the etiCORE use cases (e. g., operation system version, java version, framework version, driver version), a self-declaration from the component manufacturer is required, stating that the changes made do not affect any etiCORE use cases tested in the certification test laboratory and do not affect the etiCORE functionality in any way. This self-declaration must be submitted to VDV eTicket Service GmbH & Co. KG and automatically presented to the customer (transport companies) following a purchase or update.



3.2.6 Password entry

If it is necessary to enter a personal password to authorise the reading of personal data, the following requirements apply:

Whether as a separate component or part of the terminal, the keyboard must

- contain the numeric digits from 0 to 9

An attacker must be prevented from manipulating the keyboard and scanning the password while it is being entered.



3.2.7 Behaviour of the terminal in the event of unforeseen circumstances

In the event of a sudden power failure, the terminal will interrupt all actions and save its current status. When service is resumed, it restarts with the saved data. This procedure must be recorded.

Defects in the terminal must not lead to losses of read data and counter values that are managed in connection with the etiCORE use cases. If errors or damage to the terminal occur that make it impossible to obtain or read data and counter values, it must be possible to use the last data transfer to reconstruct the counter values for each terminal.

The provider must explain and implement a suitable concept for this.

Note: This behaviour is similar to the one specified for aborted transaction triggered by SAM or User Medium.



3.2.8 Additional requirements

- Design of the operator/customer interface in structure, appearance and use according to the [etiCORE Terminal Customer Interface Specification](#).
These must also be ensured for the hearing and visually impaired.
The icon for the terminal antenna surface (as per the ; circle with 4 arrows pointing to the ((e logo) must be applied to the reader surface of the terminal in such a way that it indicates the optimum reading position for the user medium.
- Presence of a real-time clock, including date function, with a deviation of <2 min/year.
- Regular updating (usually at least once a day) of the hotlists must be ensured.



3.2.9 Further notes

{Pkg.Notes}

3.2.9.1 Historical characters

Note: Since no historical characters for the ATR were specified in the etiCORE SPEC-SAM for the interface to the SAM, the terminal must not expect or require any historical characters. The etiCORE SPEC-UM also does not specify any historical characters/bytes for the interface to the user medium, so the terminal must not expect any historical characters/bytes here either.

3.2.9.2 Note for readers that have to be able to process cards with DESFire emulation

Note: Due to NXP's specifications for DESFire emulation, DESFire cards are delivered with a maximum frame size of 64 bytes. In tests with different frame sizes, this frame size gave the best results in tests with the etiCORE User Medium Application and the DESFire emulation. Omnikey readers seem to ignore the maximum frame size by default and send larger commands to the card. For this, there is a "DESFire Native Mode", in which the frame sizes are reduced (to 64 bytes), allowing for successful readouts.

4 Labelling of terminal devices

This chapter describes the labelling of the terminal devices for the application areas.

4.1 Smart Card Reader according to ISO/IEC 14443

The antenna area of the reader is to be labelled with the following symbol for self-service or user-operated terminals.

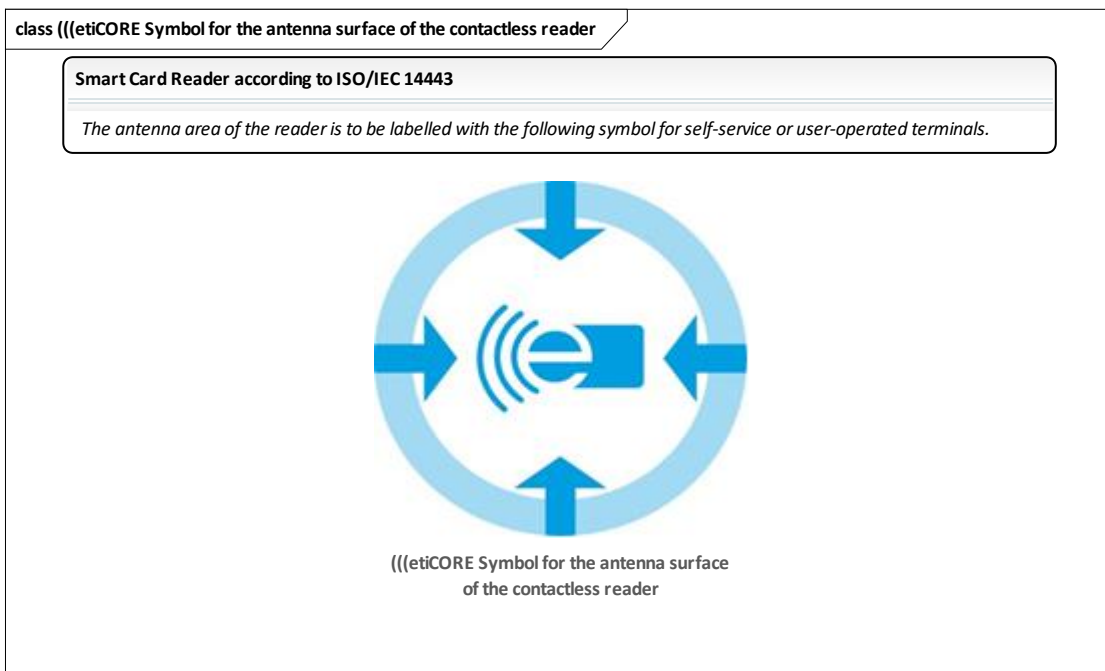


Figure 5: ((etiCORE Symbol for the antenna surface of the contactless reader

4.2 Smart Card Reader according to EMVco

The EMVCo symbol is permitted as an antenna label if the EMVCo payment function has also been implemented.

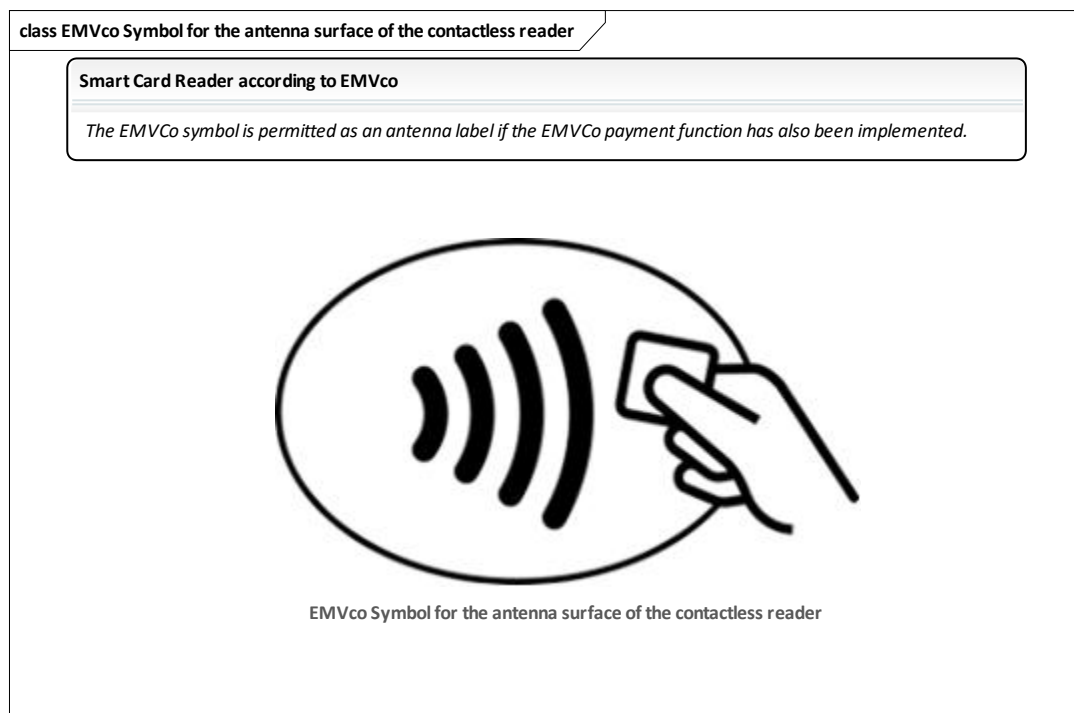


Figure 6: EMVco Symbol for the antenna surface of the contactless reader

5 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.

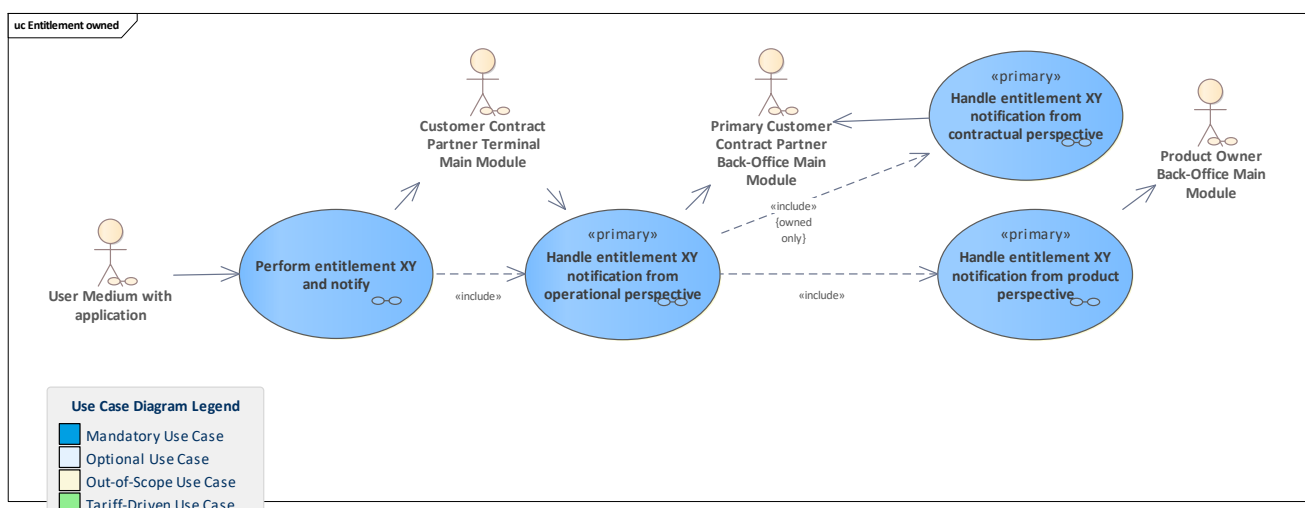


Figure 7: Entitlement owned

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.

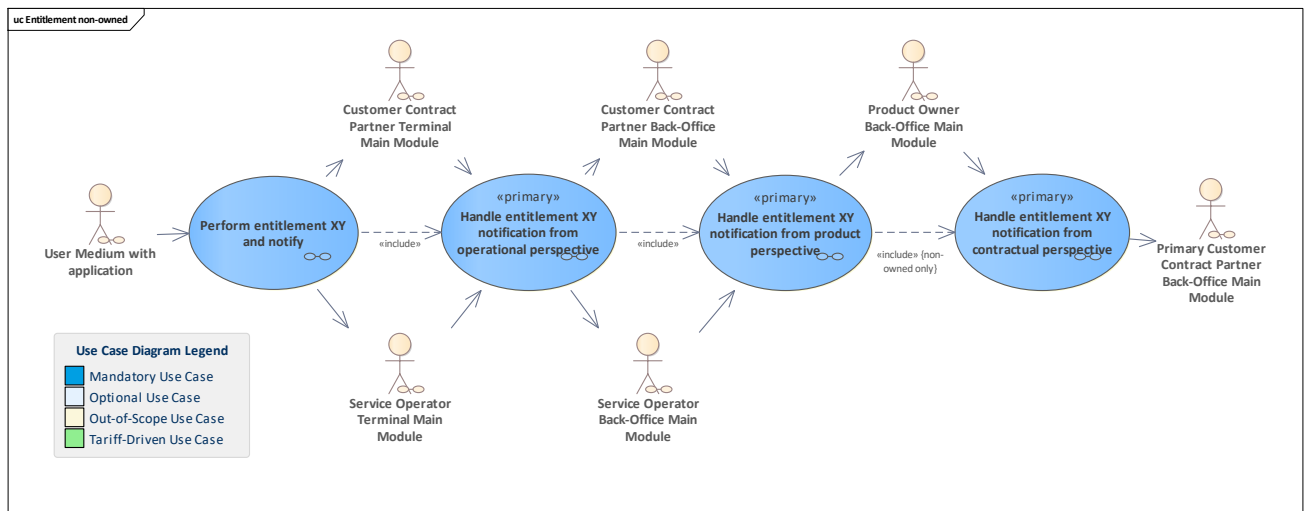


Figure 8: Entitlement non-owned

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.

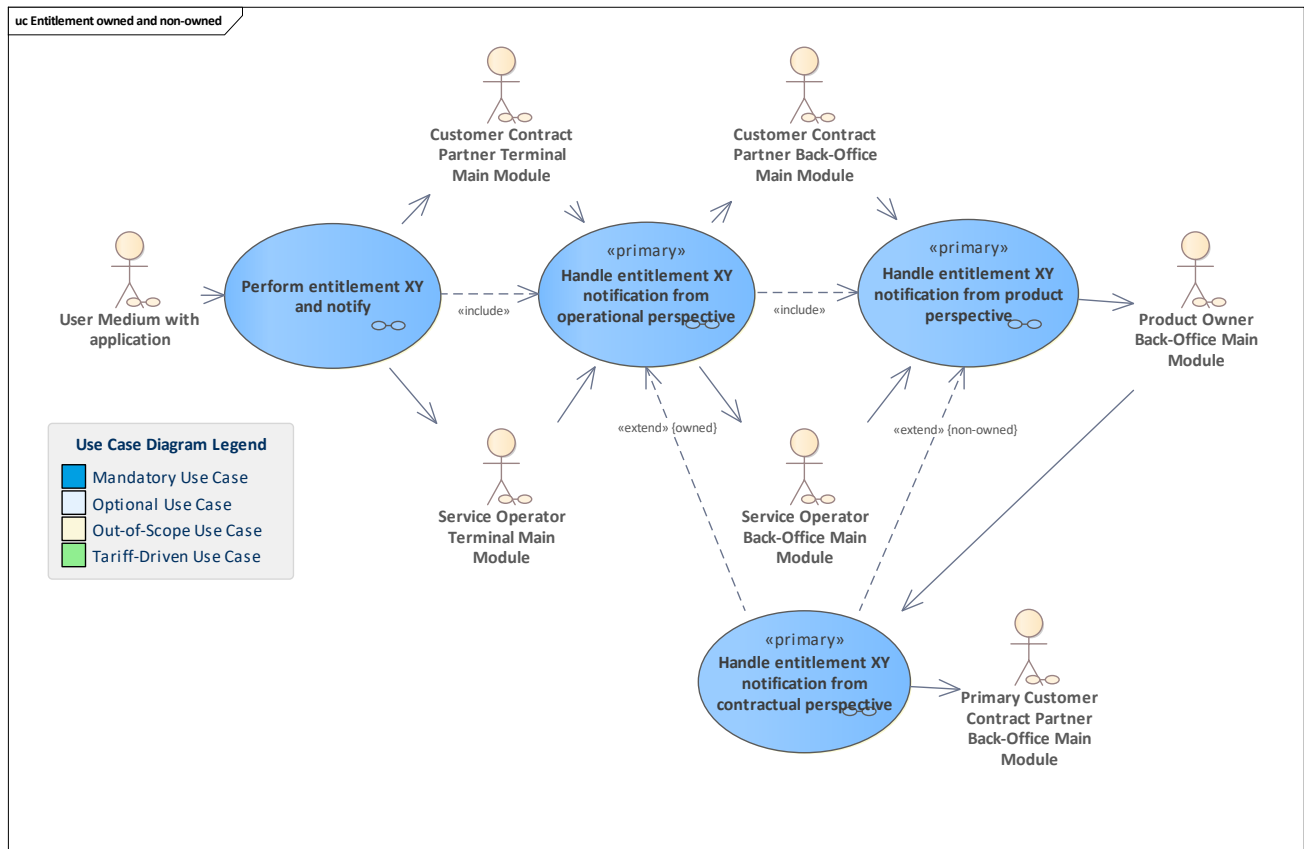


Figure 9: Entitlement owned and non-owned

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.

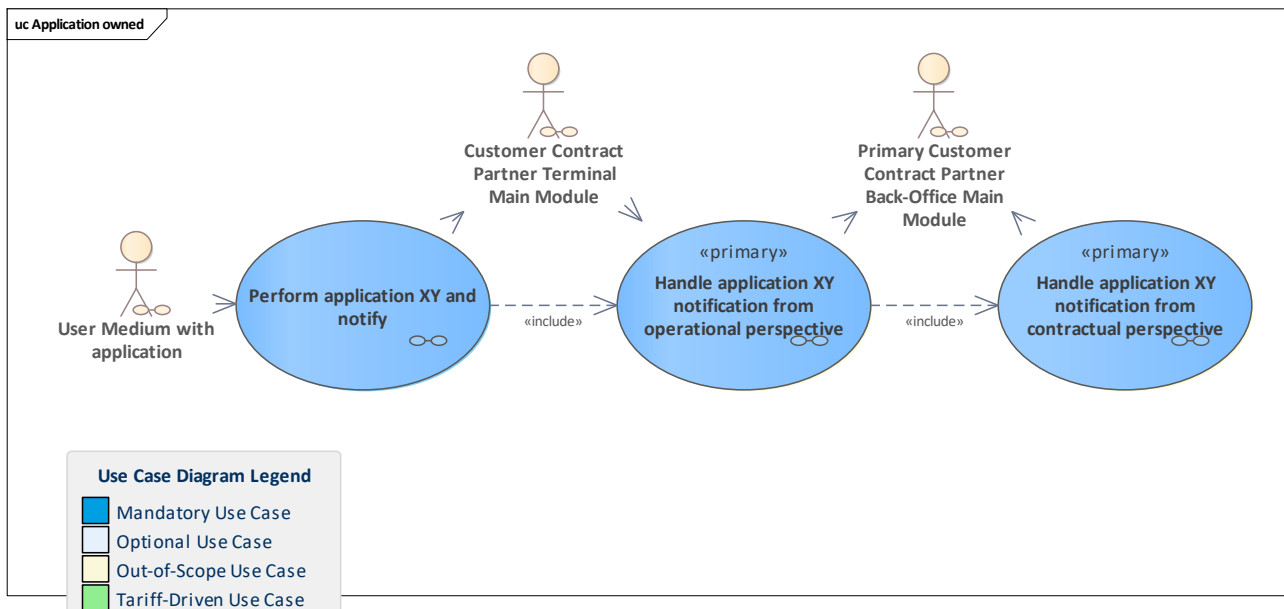


Figure 10: Application owned

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

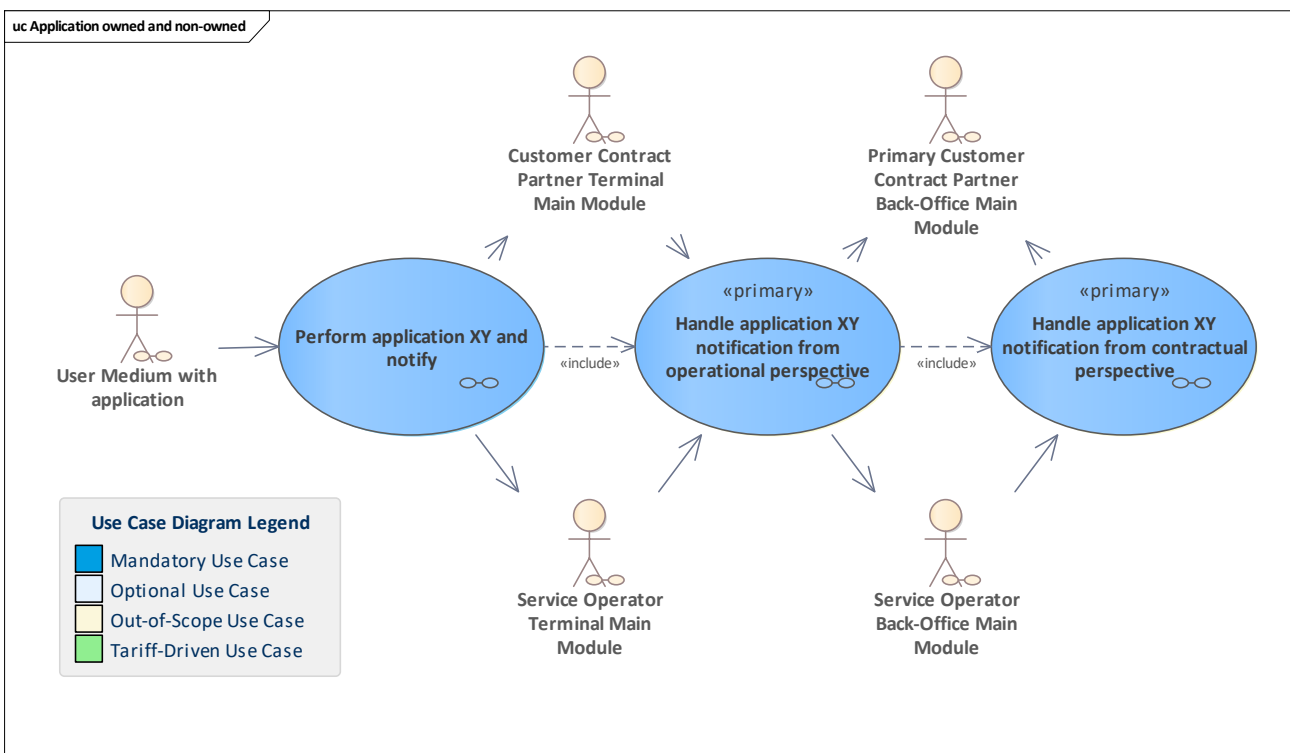


Figure 11: Application owned and non-owned

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.

5.1 Supporting activities

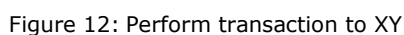
This chapter contains activities that are used in more than one use case.



5.1.1 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.



See [Prepare XY parameters](#)



5.1.1.2 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.

5.1.1.3 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.

5.1.1.4 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.

5.1.1.5 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.

5.1.1.6 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.

5.1.2 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.

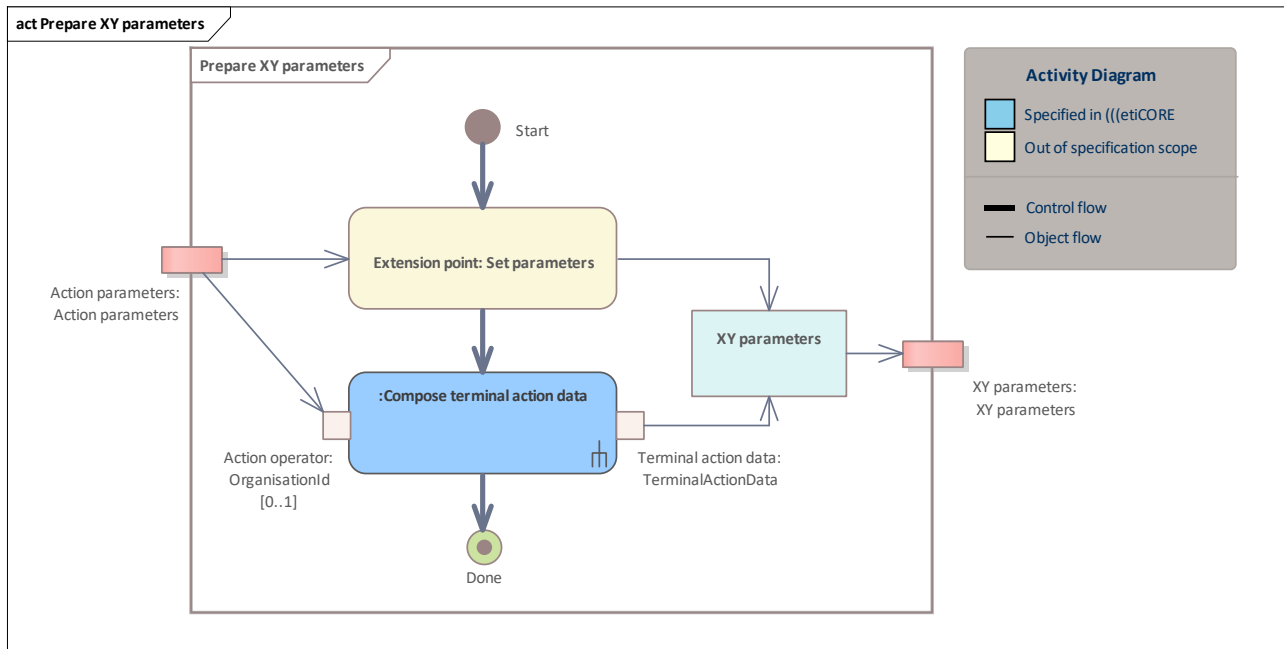


Figure 13: Prepare XY parameters

5.1.2.1 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.

5.2 Supporting classes

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

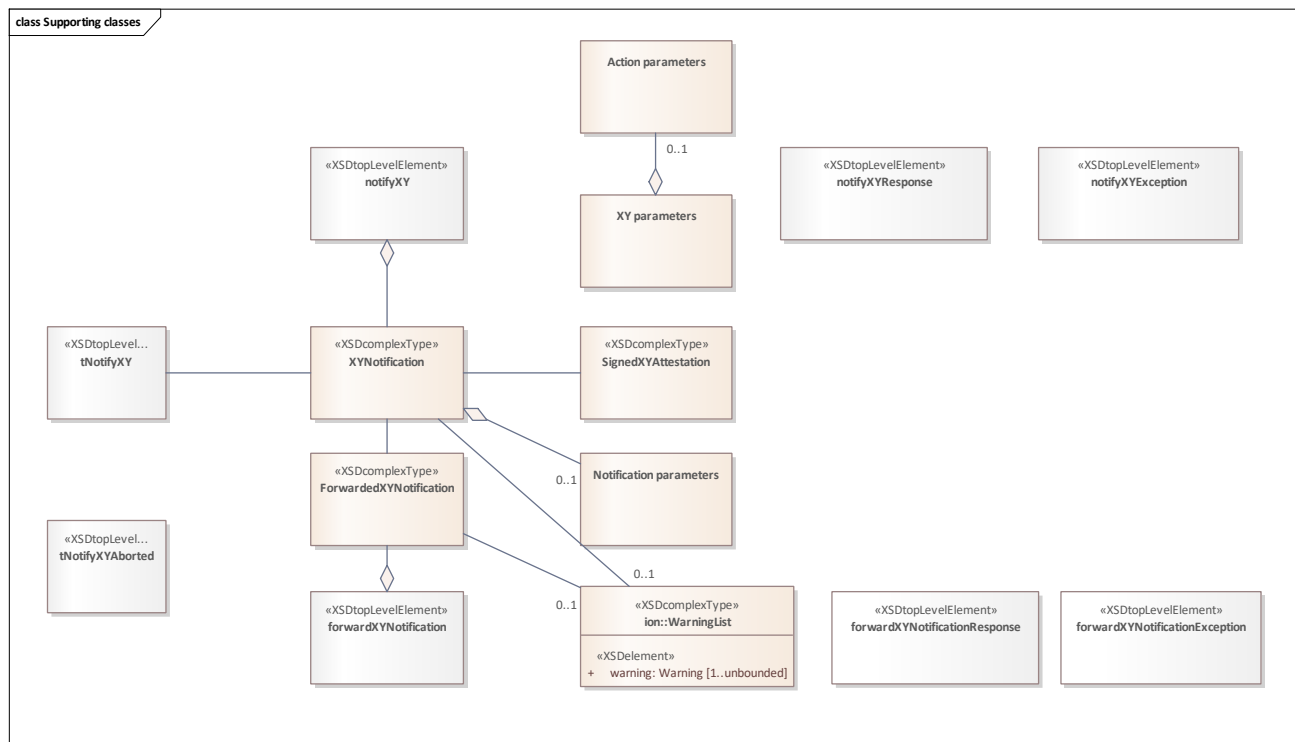


Figure 14: Supporting classes

5.2.1 Action parameters

Only applicable to entitlement actions.

Always contains the entry ID and the product ID.

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

5.2.2 XY parameters

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

5.2.3 SignedXYAttestation

An attestation of successful action execution signed by the UM involved.

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

5.2.4 Notification parameters

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



5.2.5 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

5.2.6 ForwardedXYNotification

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

5.2.7 tNotifyXY

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

5.2.8 tNotifyXYAborted

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

5.2.9 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.

5.2.10 notifyXYResponse

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

5.2.11 notifyXYException

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

5.2.12 forwardXYNotification

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

5.2.13 forwardXYNotificationResponse

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

5.2.14 forwardXYNotificationException

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

5.3 Perform entitlement XY and notify

See [Perform entitlement XY and notify](#)

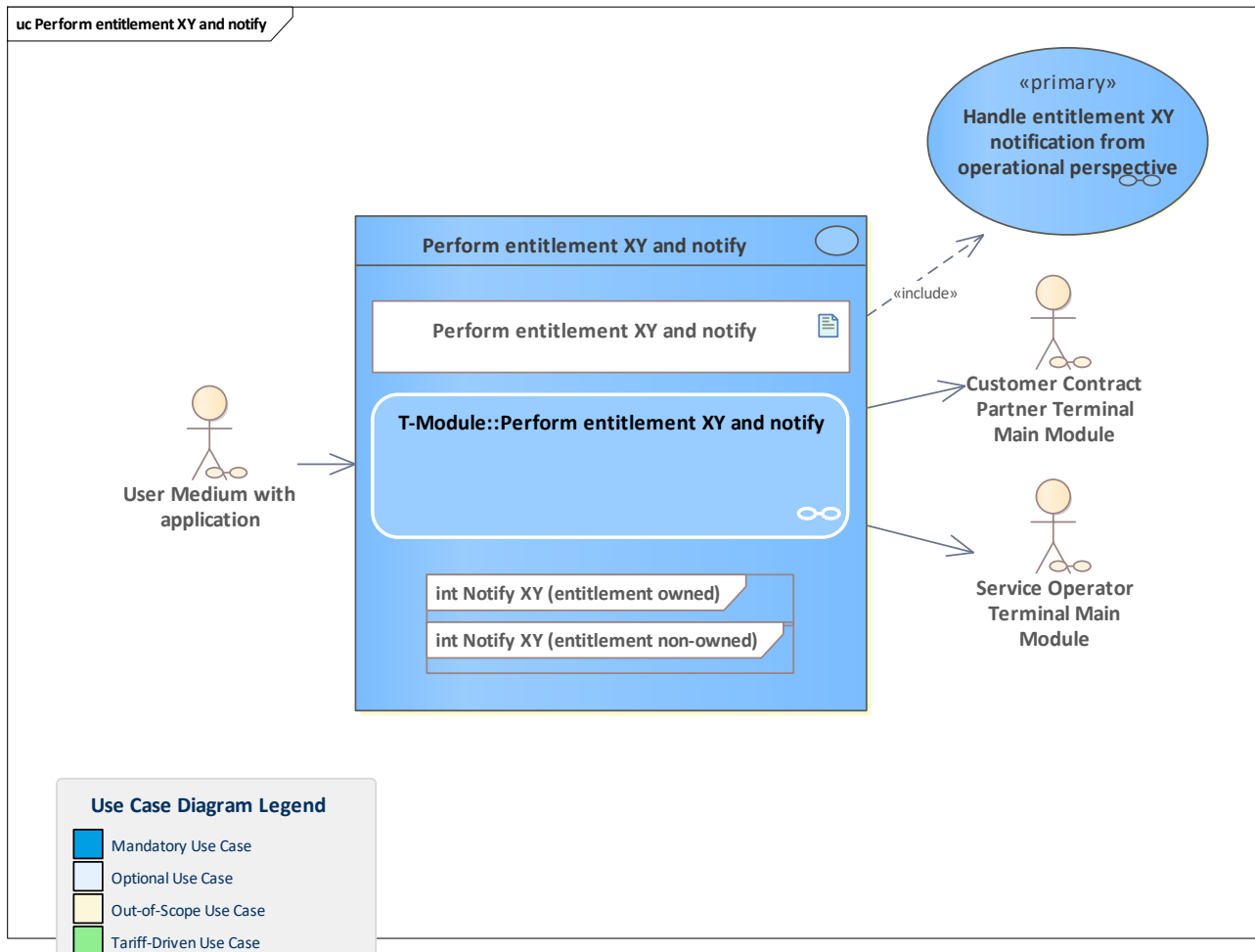


Figure 15: Perform entitlement XY and notify

5.3.1 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).

5.3.2 T-Module::Perform entitlement XY and notify

See [Perform entitlement XY and notify](#)

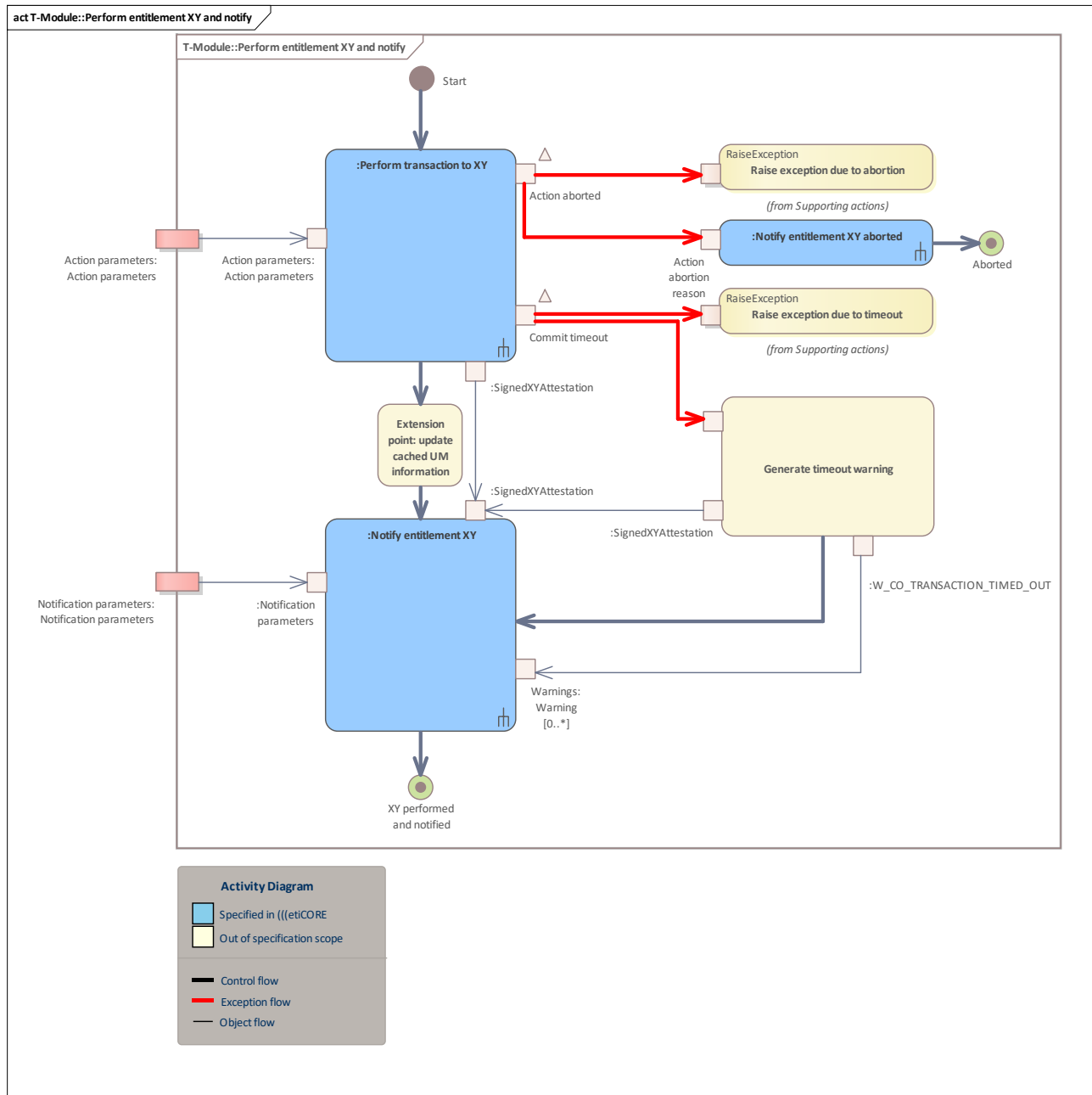


Figure 16: T-Module::Perform entitlement XY and notify

5.3.2.1

See [Perform transaction to XY](#)

5.3.2.2

See [Notify entitlement XY](#)

5.3.2.3

See [Notify entitlement XY aborted](#)

5.3.2.4 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.

5.3.2.5 Generate timeout warning

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

5.3.3 Notify entitlement XY

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

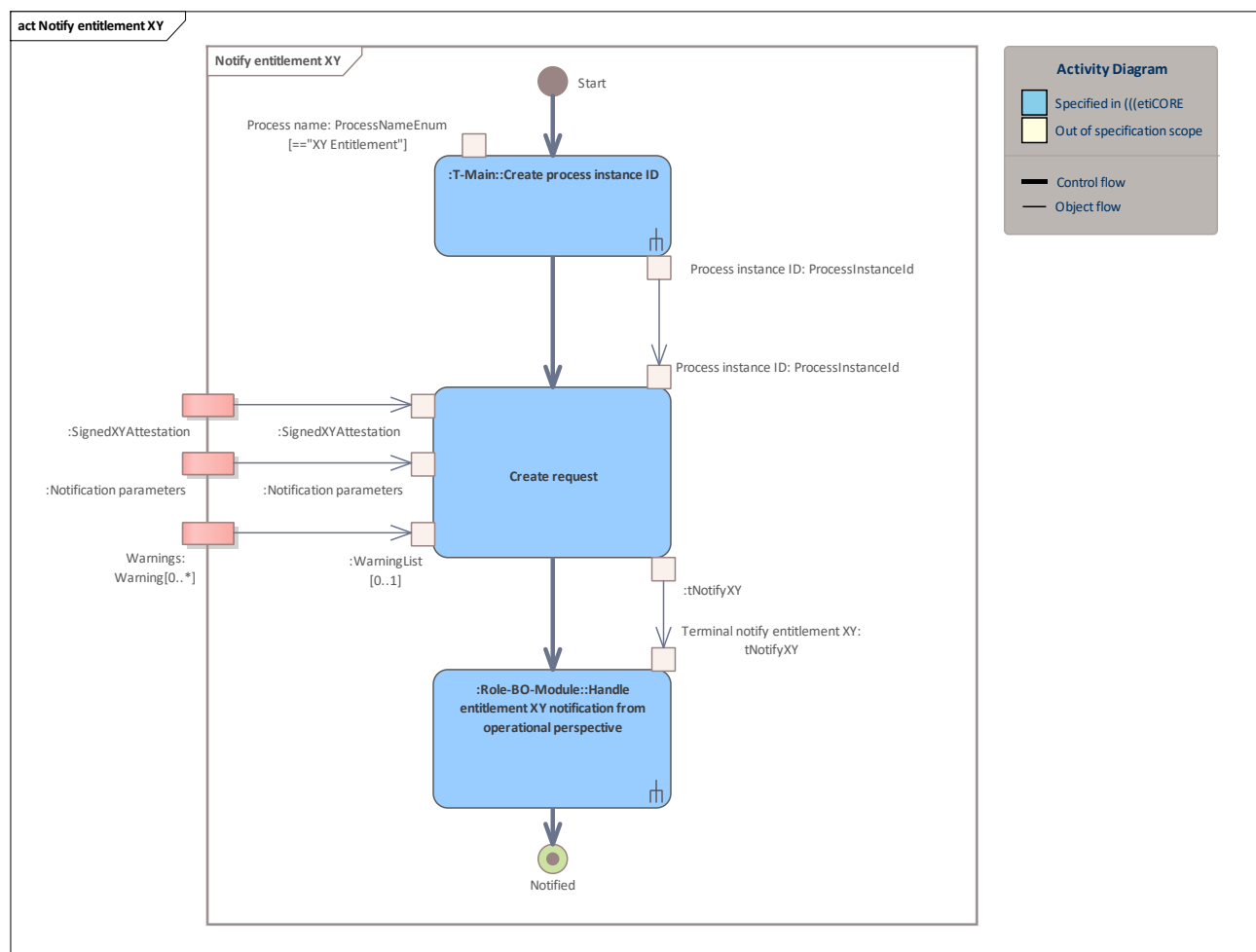


Figure 17: Notify entitlement XY

5.3.4 Notify entitlement XY aborted

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

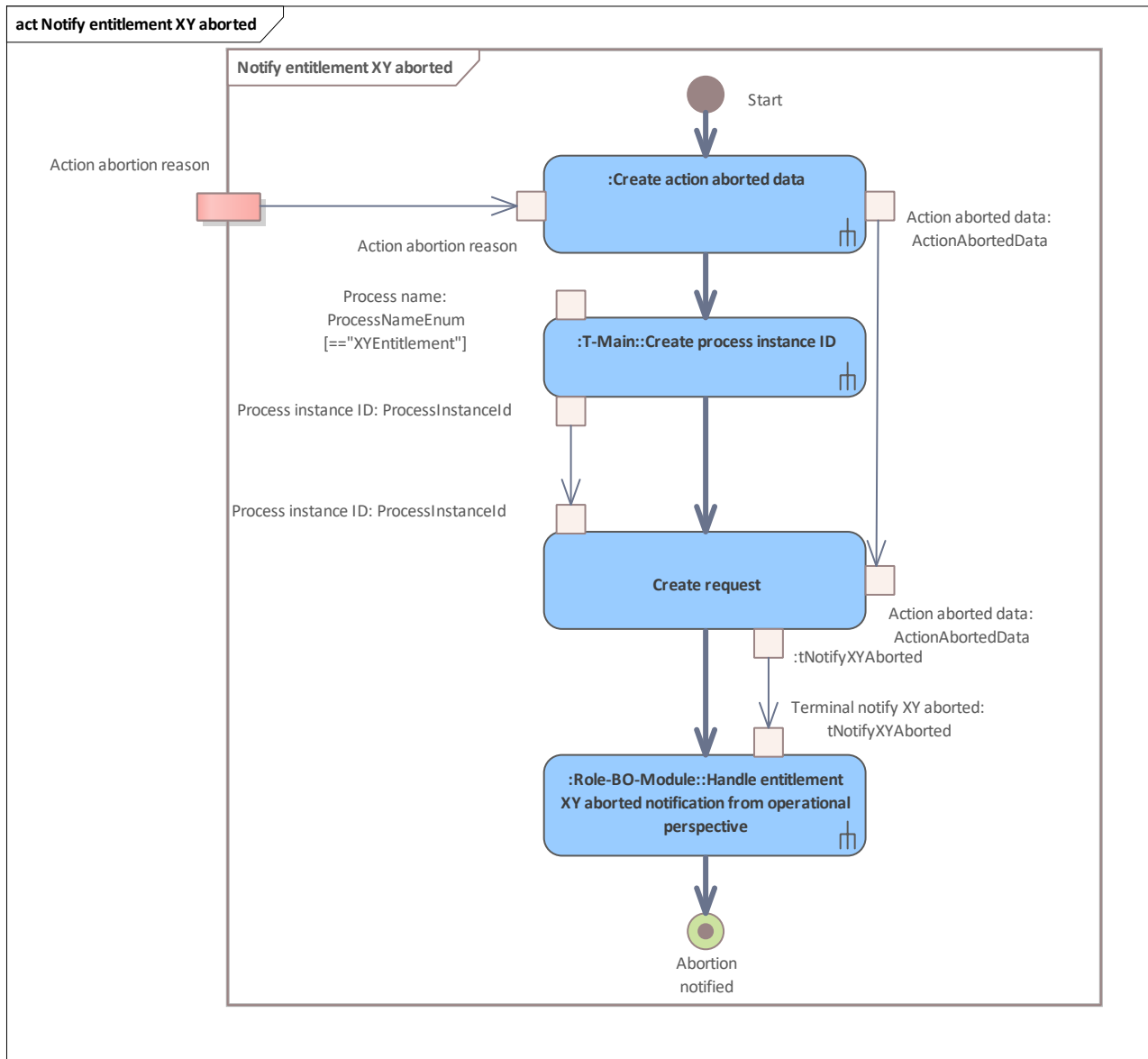


Figure 18: Notify entitlement XY aborted

5.3.5 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.

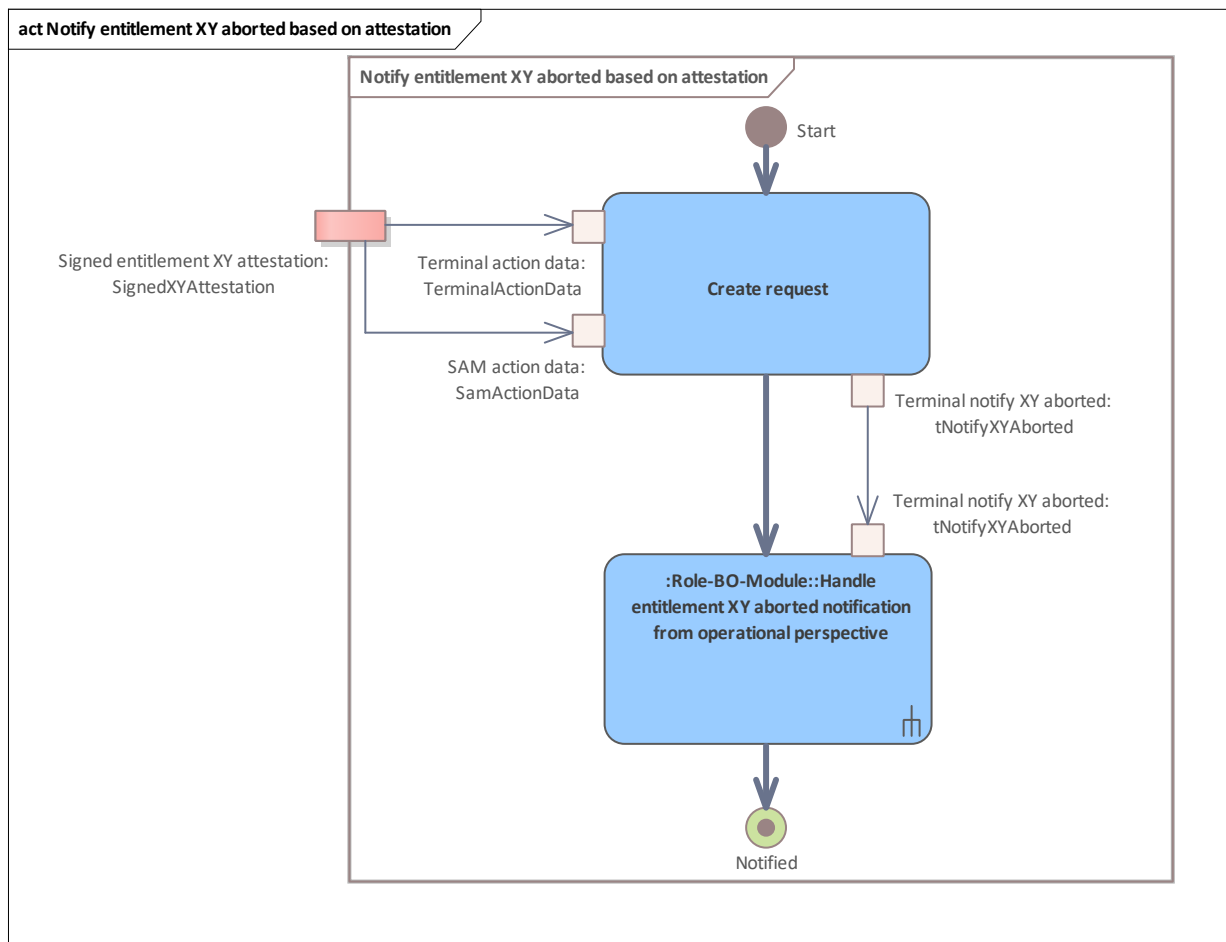


Figure 19: Notify entitlement XY aborted based on attestation

5.3.6 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.

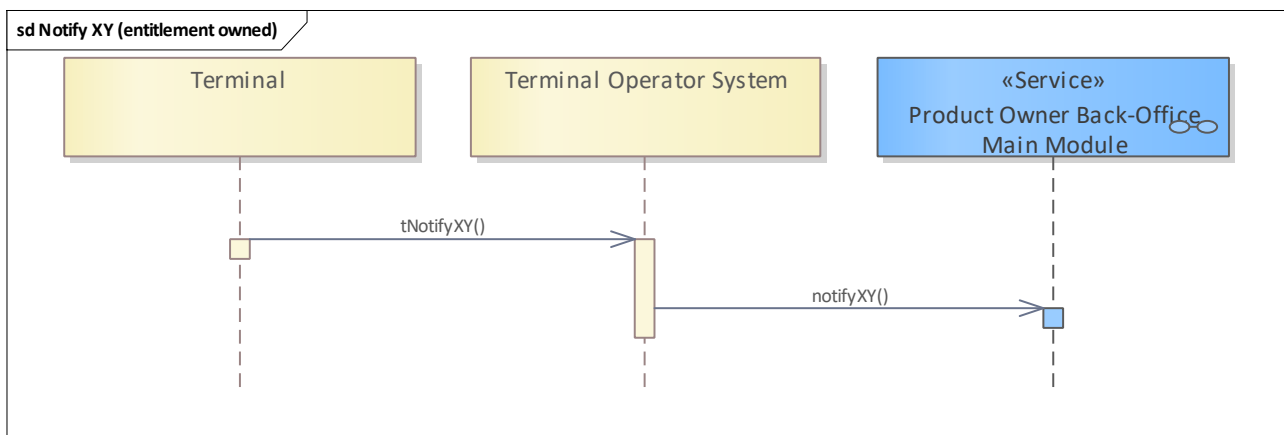


Figure 20: Notify XY (entitlement owned)

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

5.3.7 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.

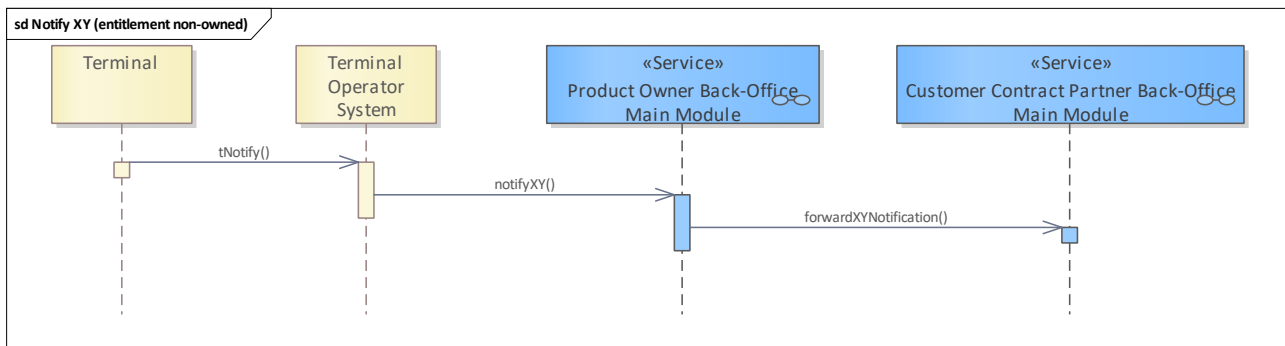


Figure 21: Notify XY (entitlement non-owned)

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

5.4 Perform application XY and notify

See [Perform application XY and notify](#)

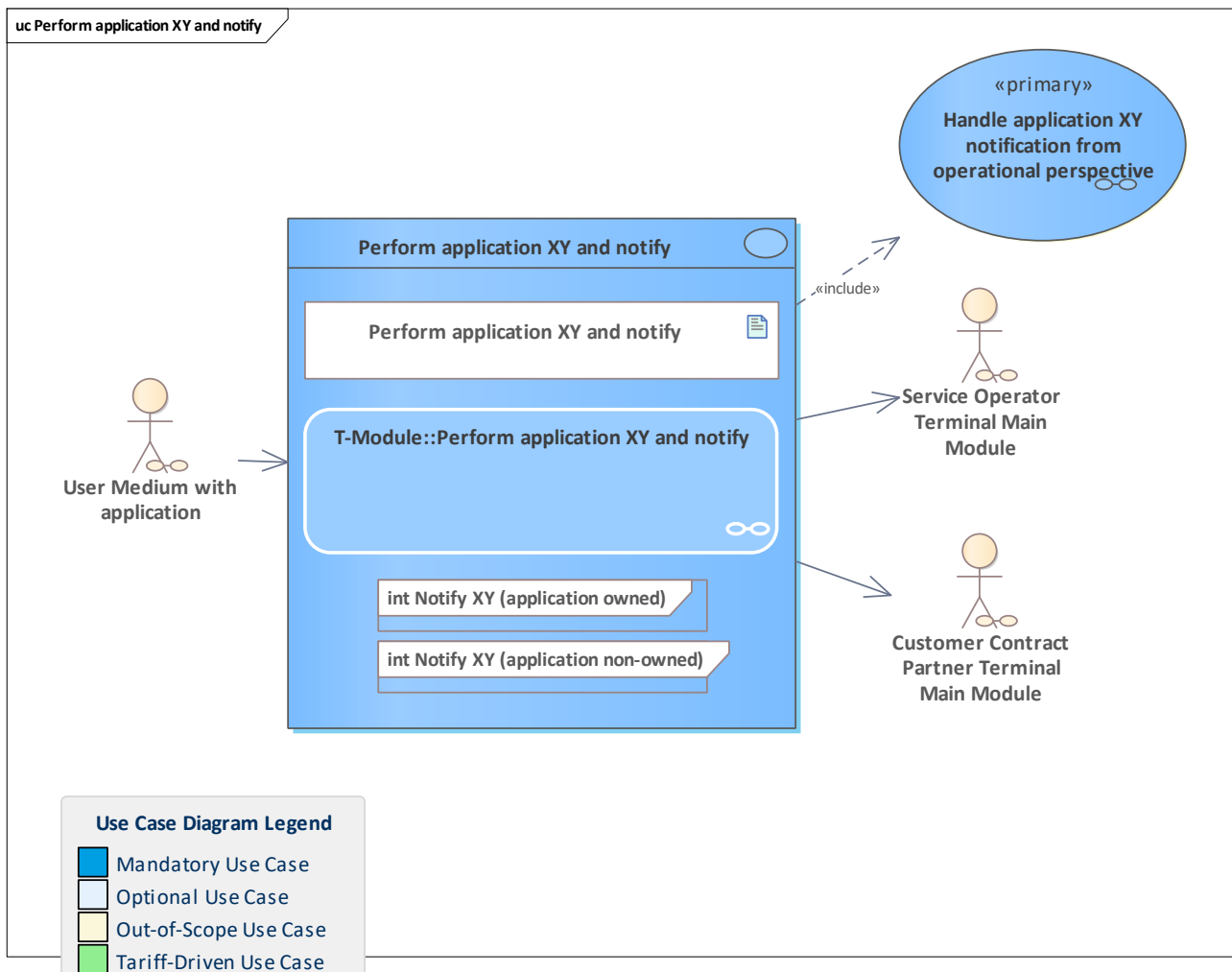


Figure 22: Perform application XY and notify

5.4.1 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.

5.4.2 T-Module::Perform application XY and notify

See [Perform application XY and notify](#)

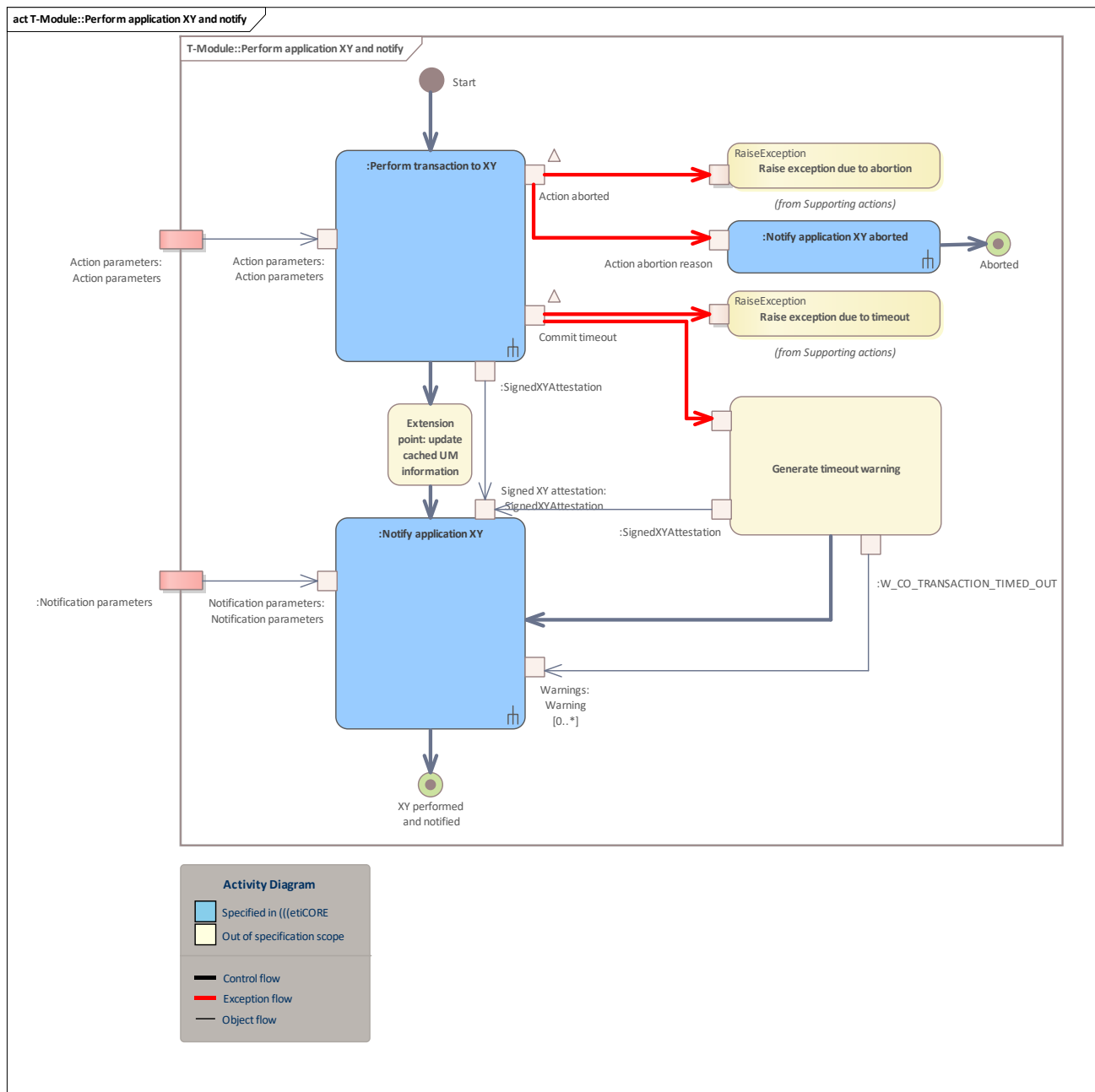


Figure 23: T-Module::Perform application XY and notify

5.4.2.1

See [Perform transaction to XY](#)

5.4.2.2

See [Notify entitlement XY aborted](#)

5.4.2.3

See [Notify entitlement XY](#)

5.4.2.4 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.

5.4.2.5 Generate timeout warning

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

5.4.3 Notify application XY

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

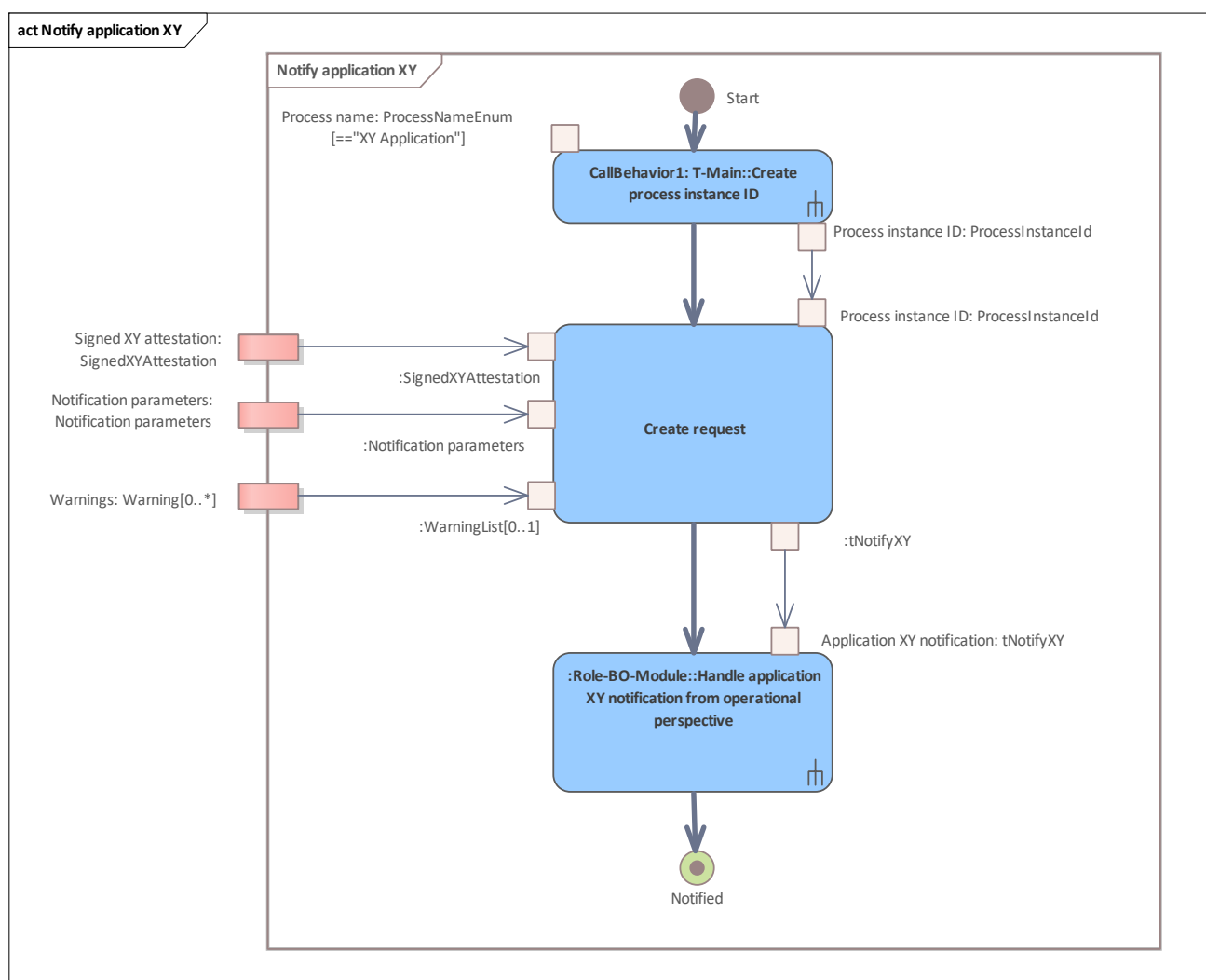


Figure 24: Notify application XY

5.4.4 Notify application XY aborted

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

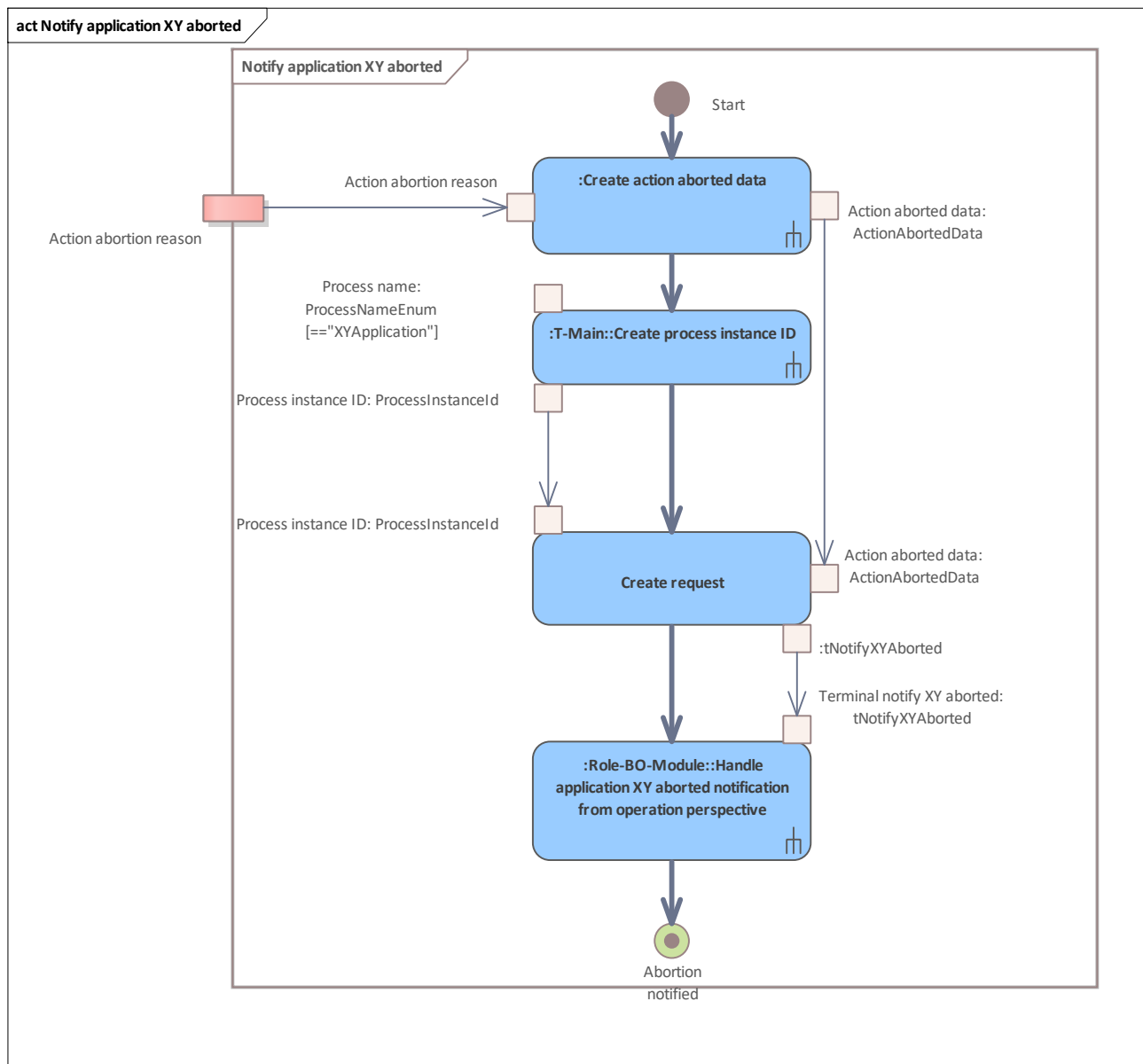


Figure 25: Notify application XY aborted

5.4.5 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).

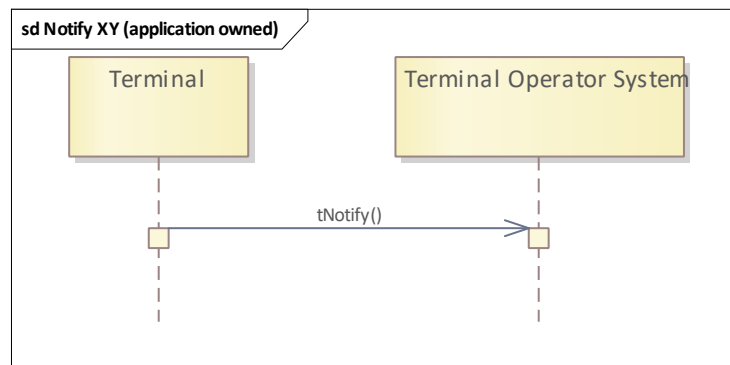


Figure 26: Notify XY (application owned)

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

5.4.6 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.

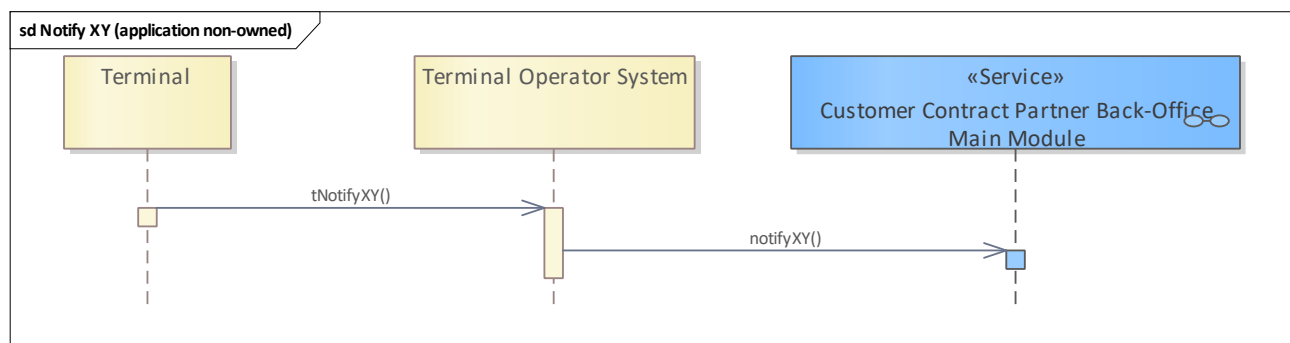


Figure 27: Notify XY (application non-owned)

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

5.5 Handle entitlement XY notification from operational perspective

See [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.

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

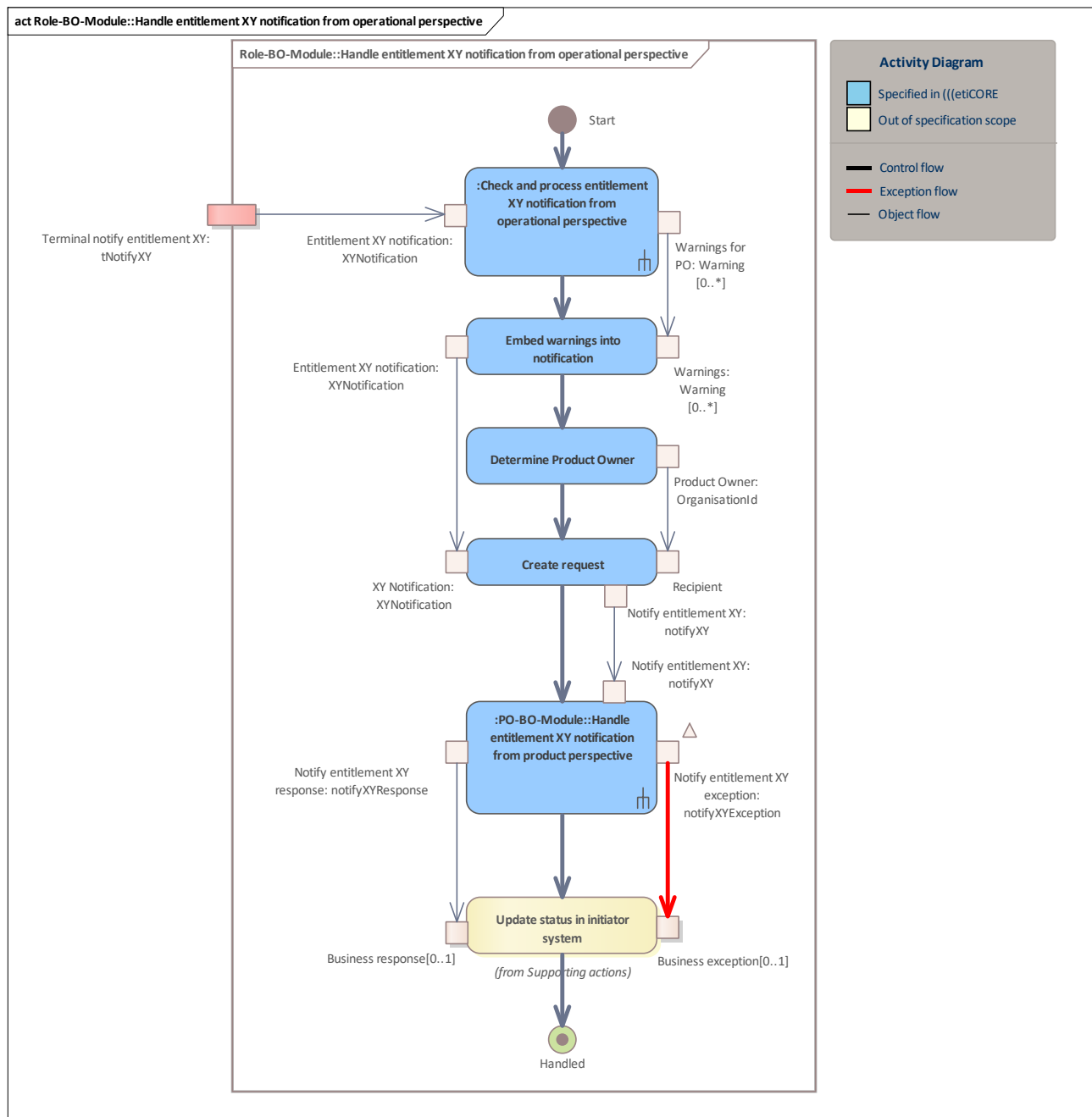


Figure 29: Role-BO-Module::Handle entitlement XY notification from operational perspective

5.5.3 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.

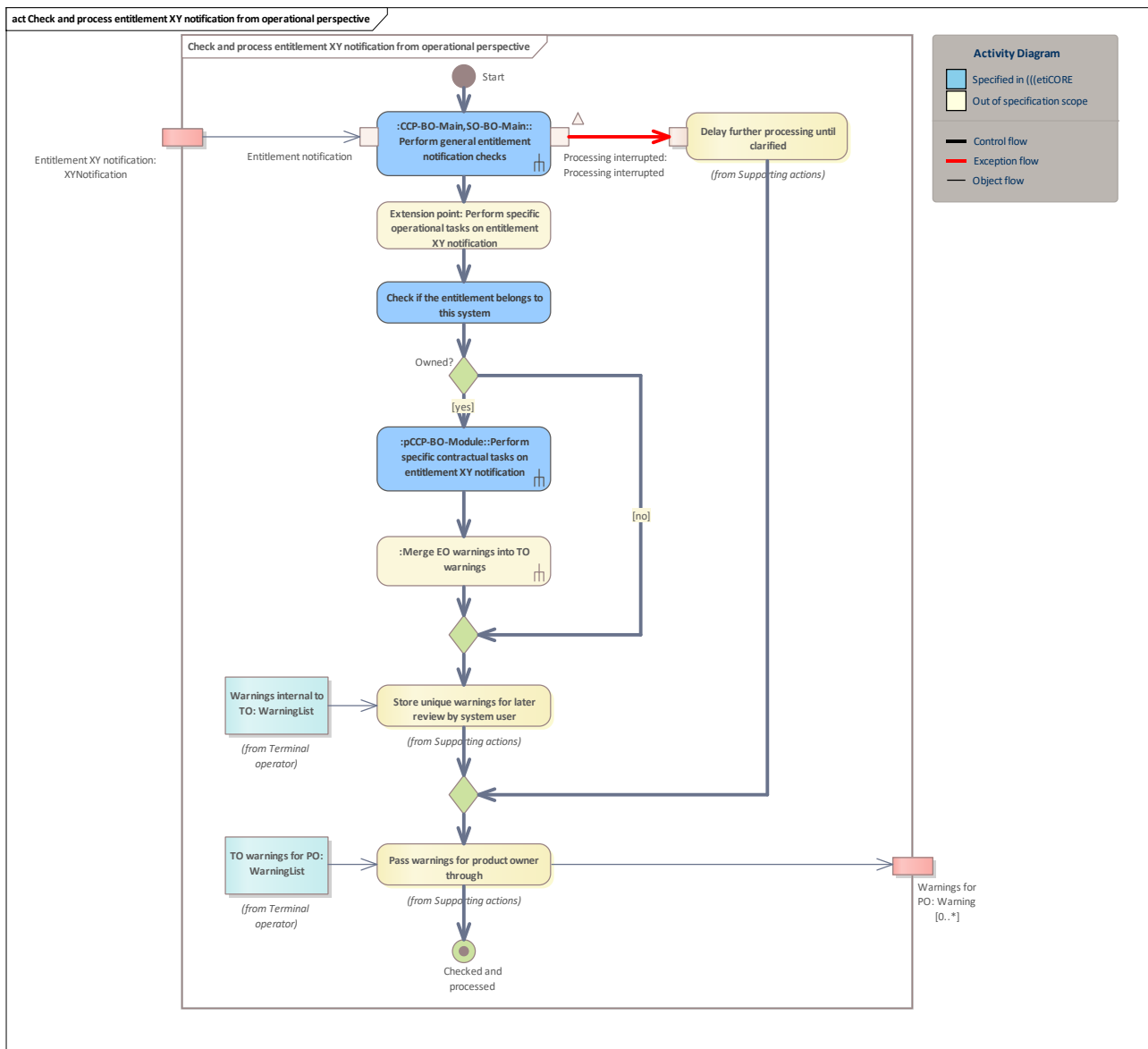


Figure 30: Check and process entitlement XY notification from operational perspective

5.5.3.1 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.

5.5.4 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.

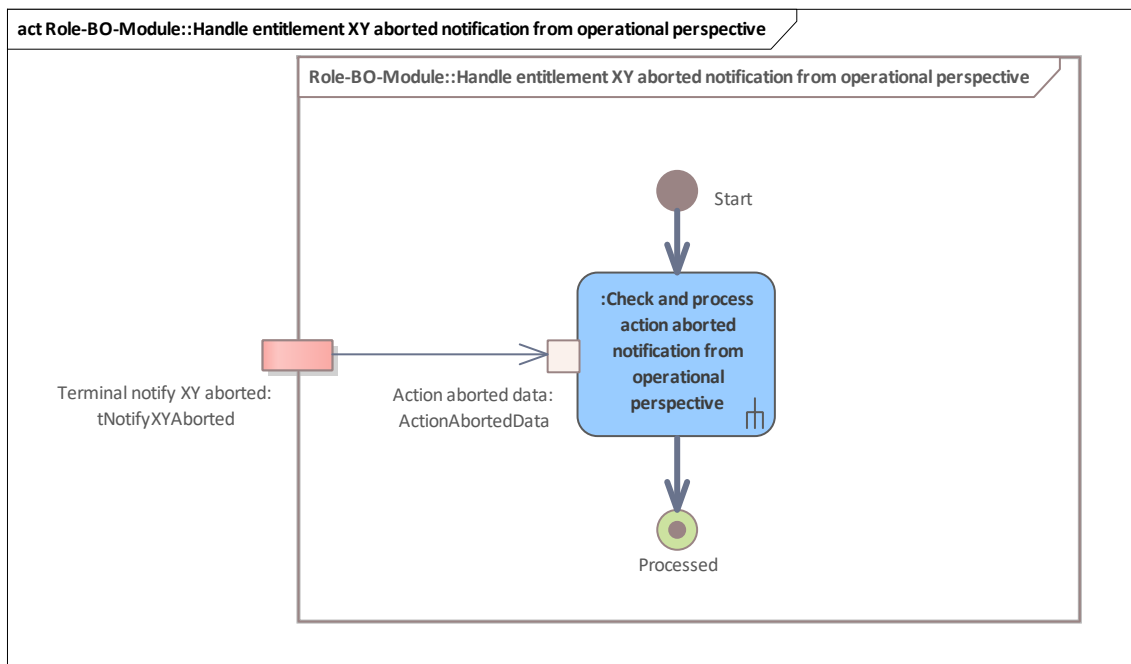


Figure 31: Role-BO-Module::Handle entitlement XY aborted notification from operational perspective

5.6 Handle application XY notification from operational perspective

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

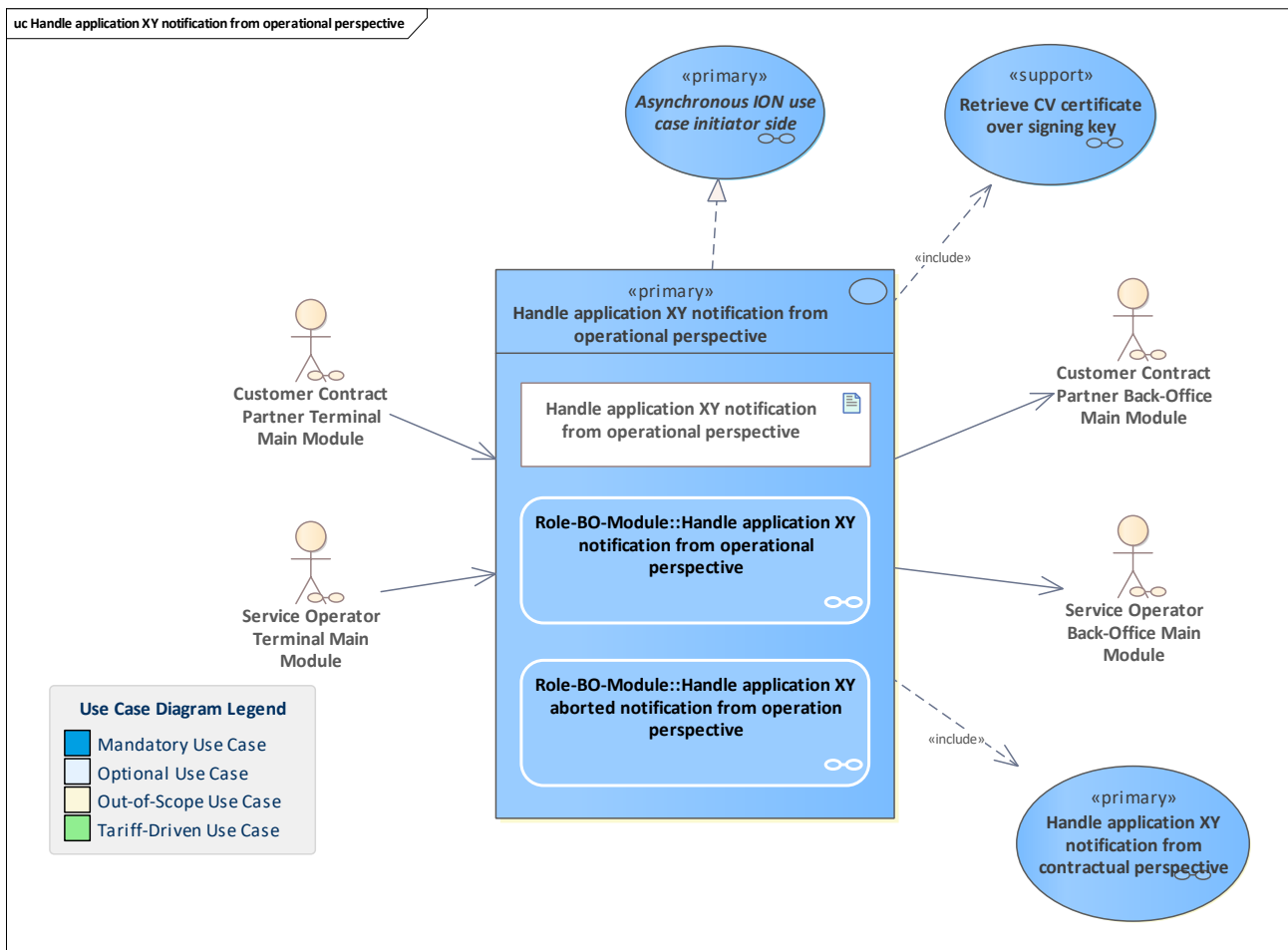


Figure 32: Handle application XY notification from operational perspective

5.6.1 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.

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

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

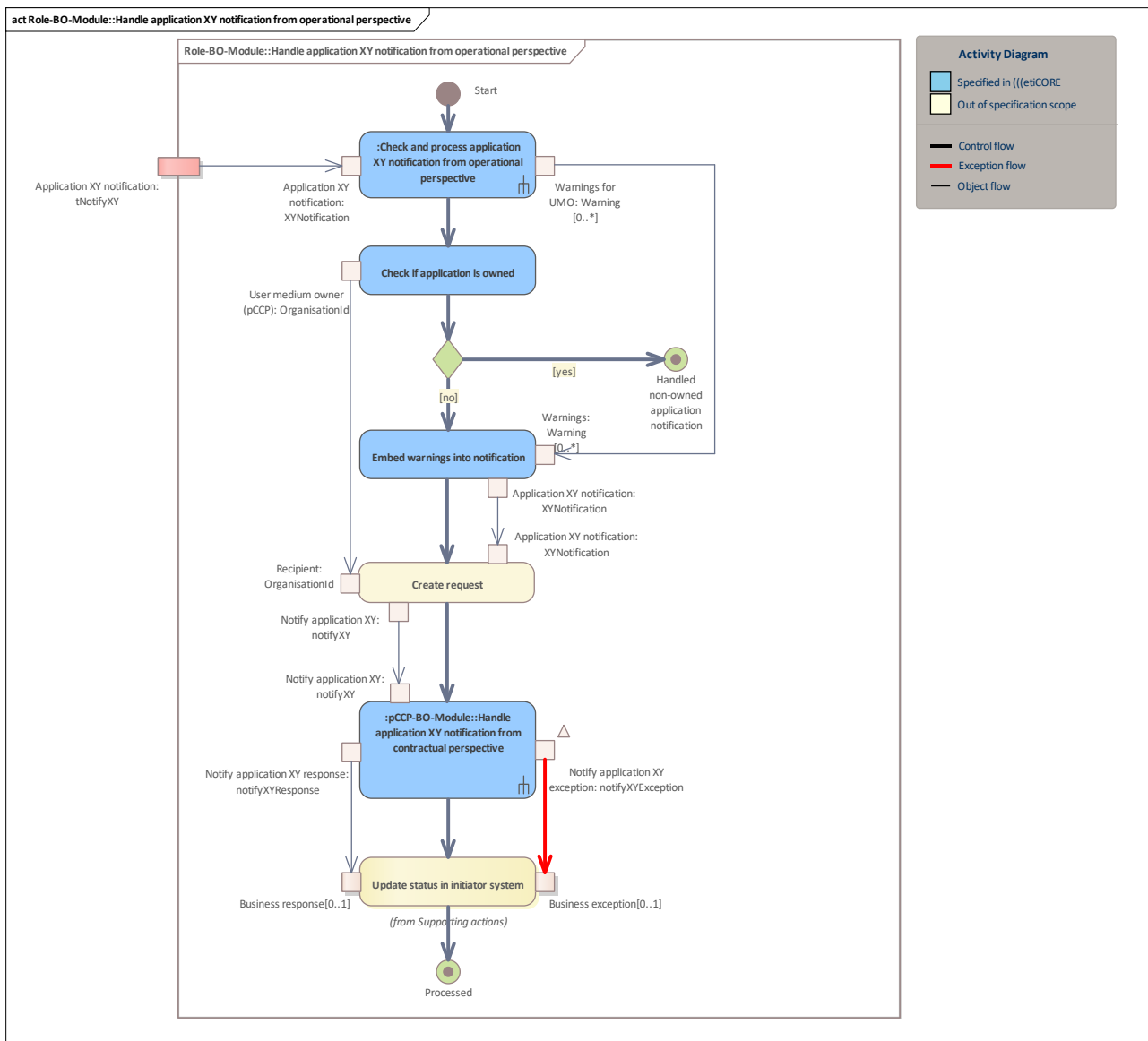


Figure 33: Role-BO-Module::Handle application XY notification from operational perspective

5.6.3 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.

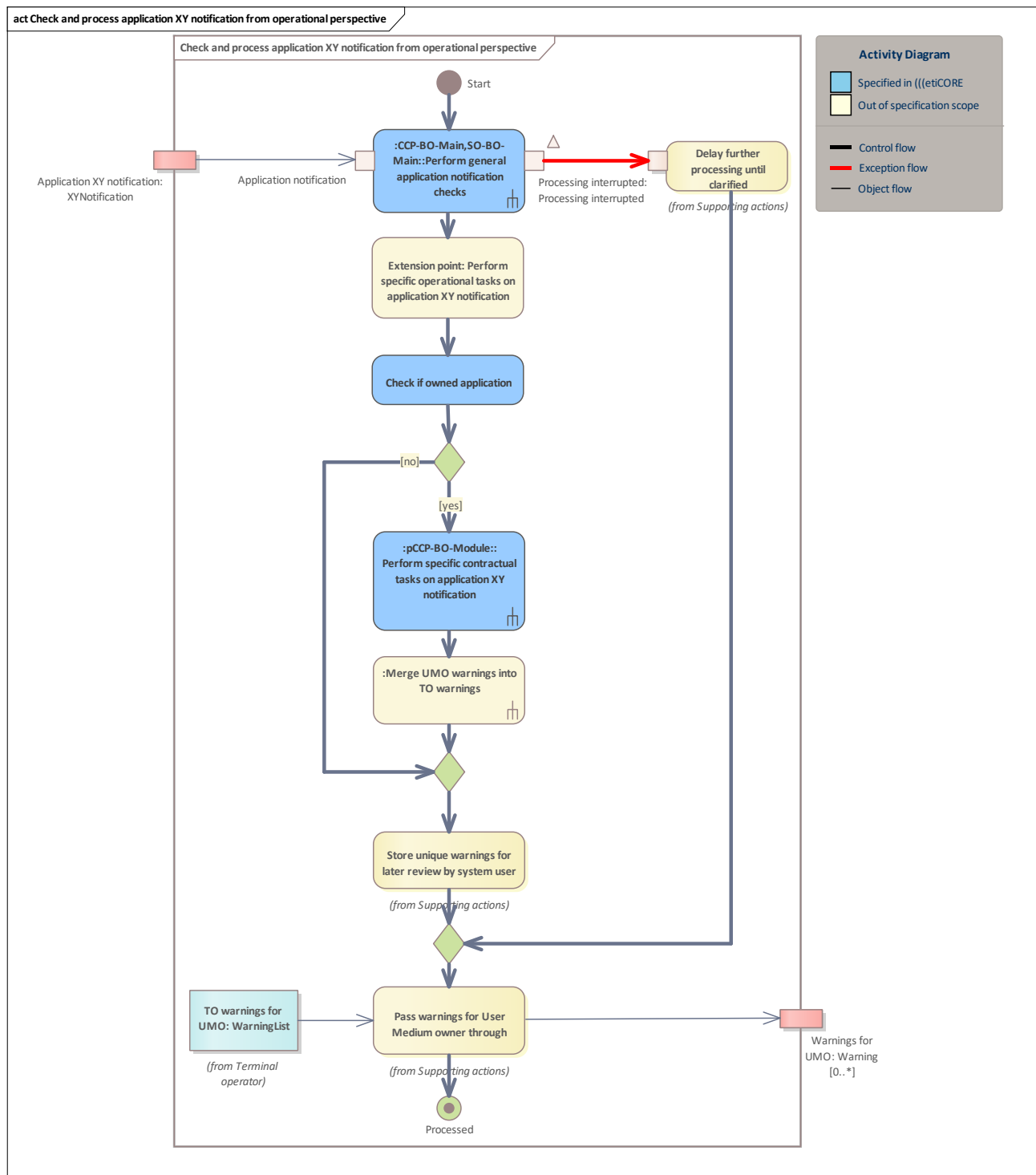


Figure 34: Check and process application XY notification from operational perspective

5.6.3.1 Extension point: Perform specific operational tasks on application XY notification

Log SAM action counter value, etc.

5.6.4 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.

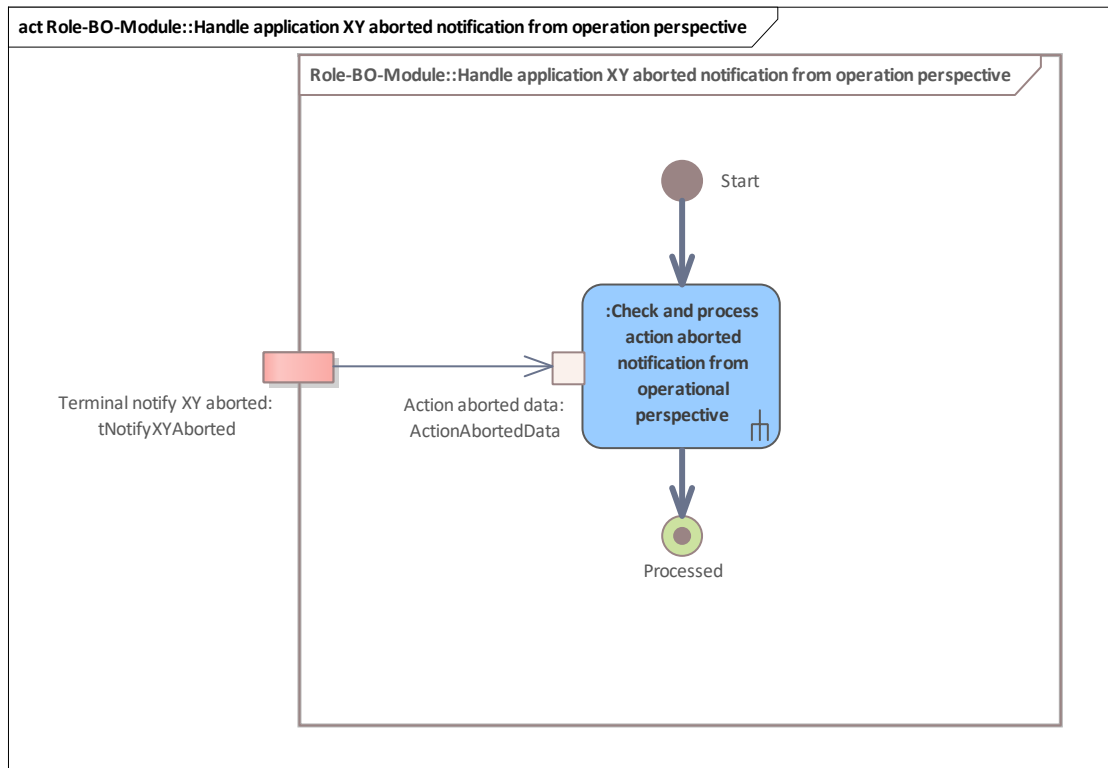


Figure 35: Role-BO-Module::Handle application XY aborted notification from operation perspective

5.7 Handle entitlement XY notification from product perspective

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

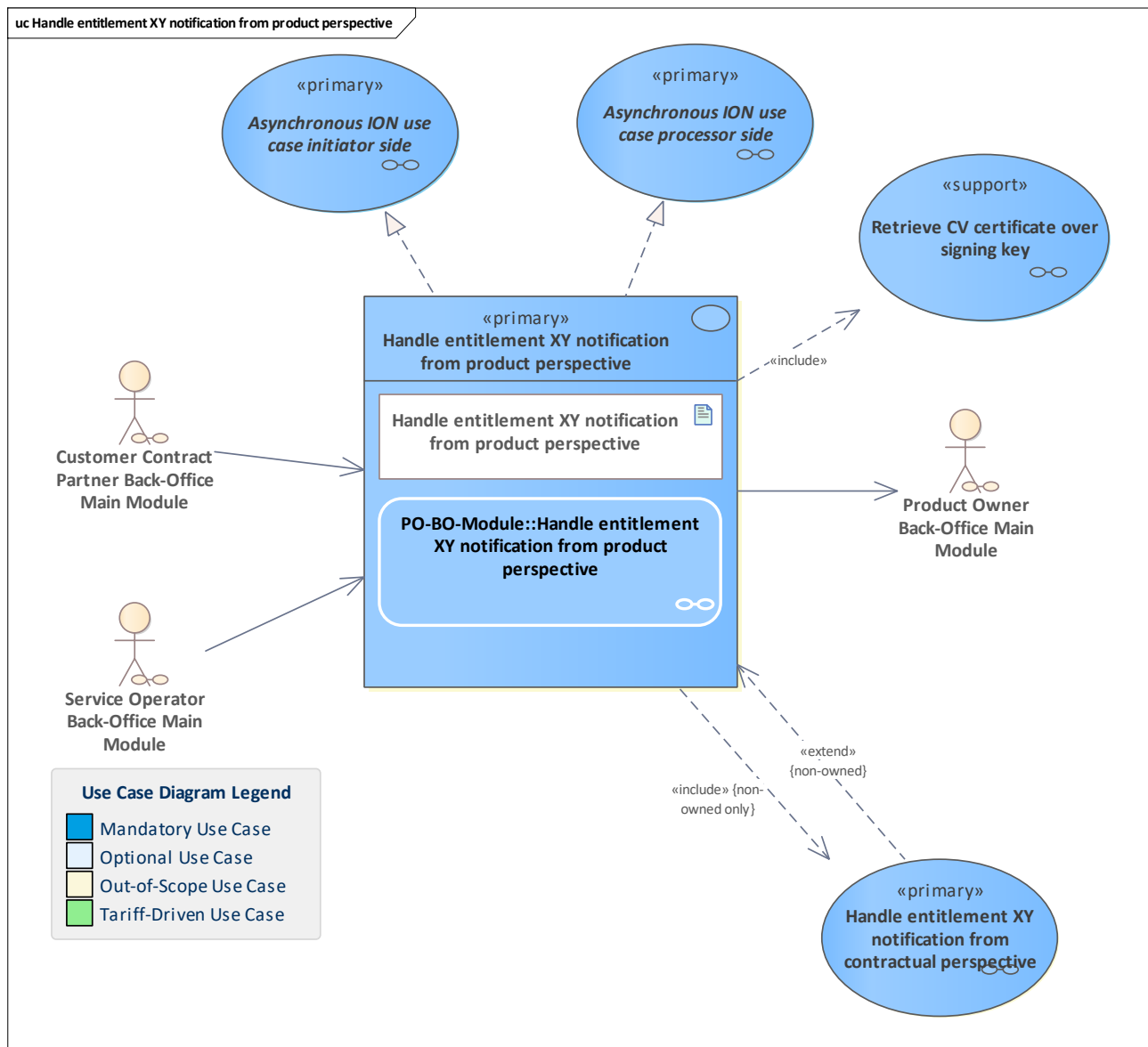


Figure 36: Handle entitlement XY notification from product perspective

5.7.1 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.

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

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

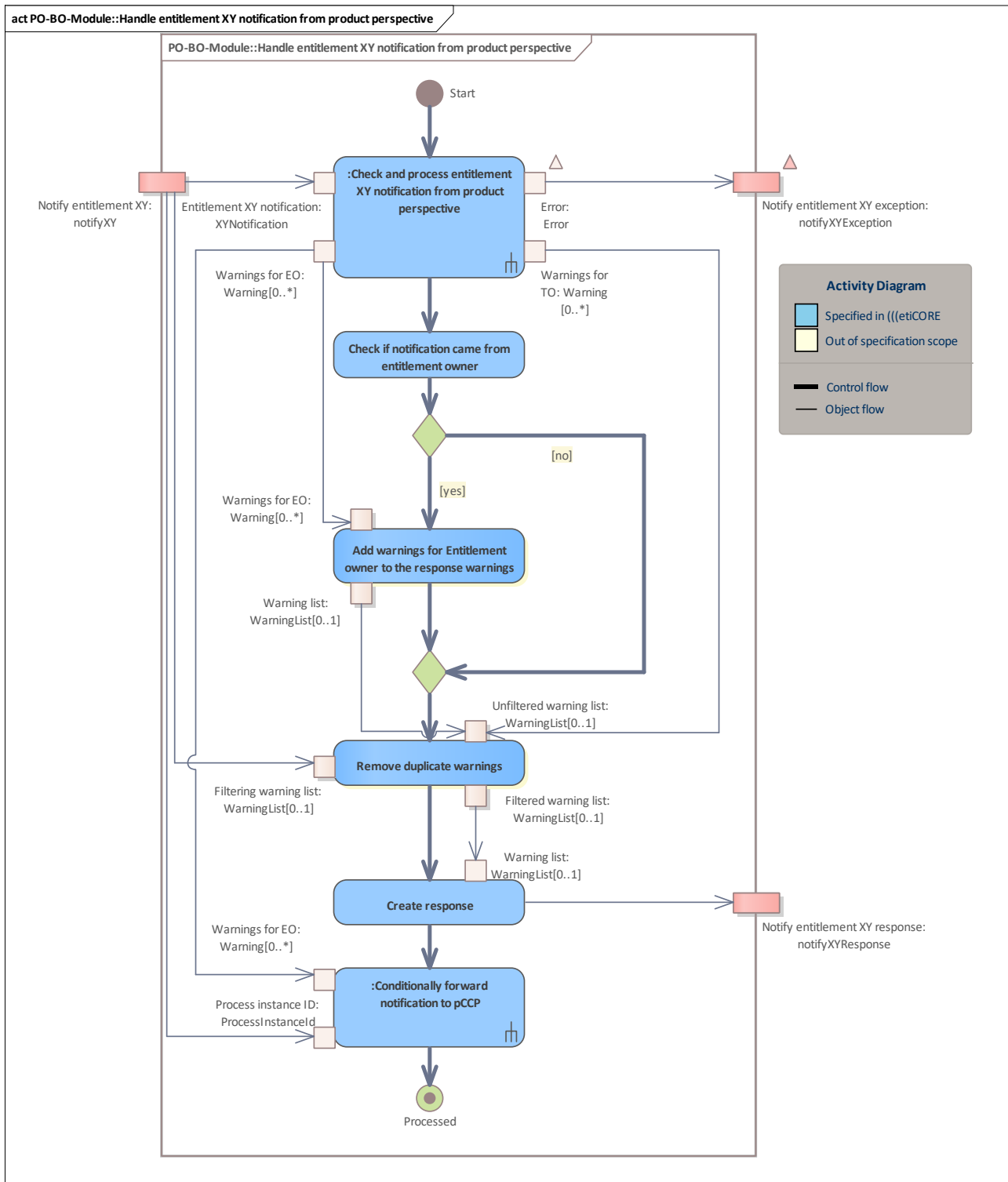


Figure 37: PO-BO-Module::Handle entitlement XY notification from product perspective

5.7.3 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.

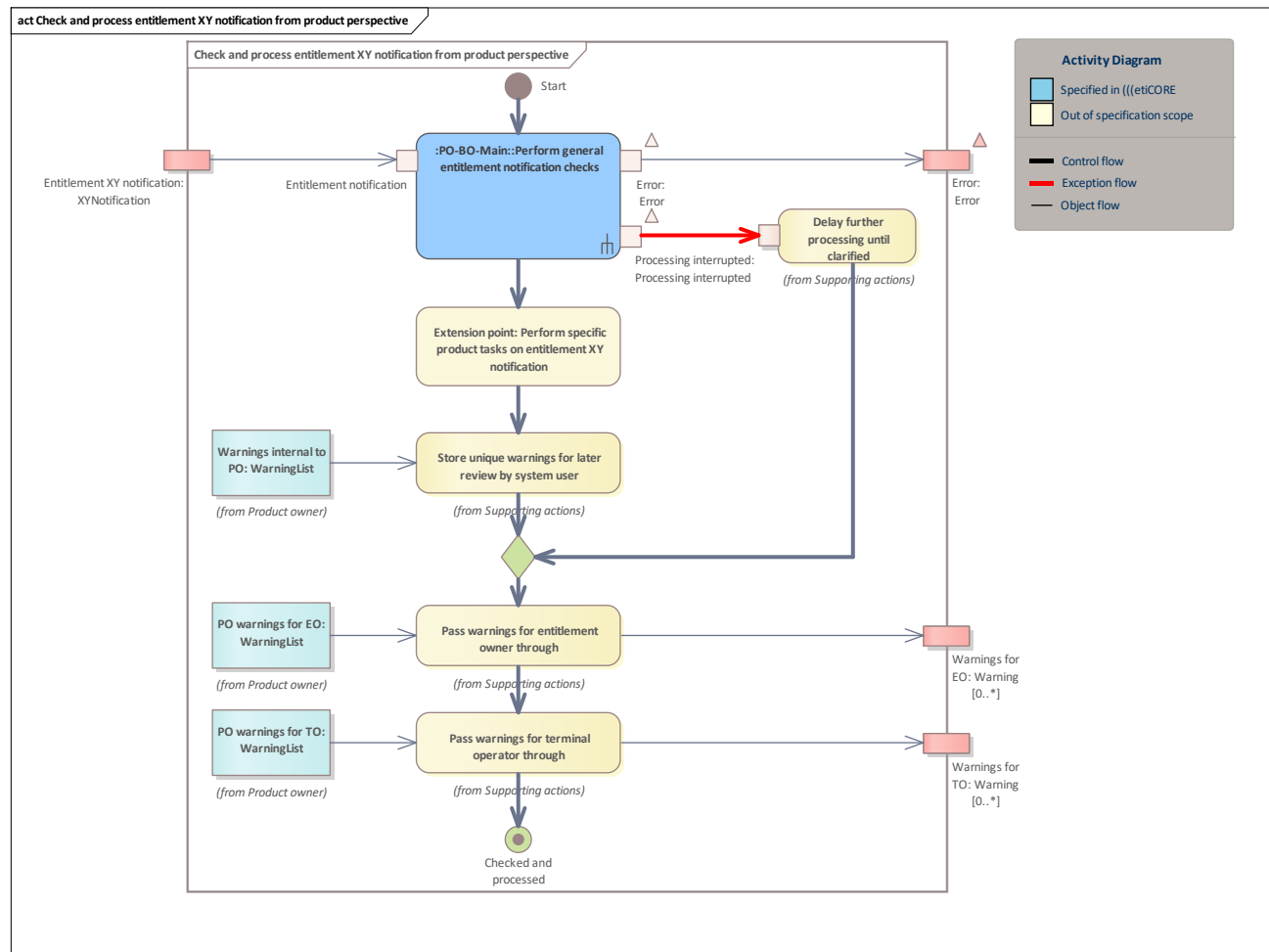


Figure 38: Check and process entitlement XY notification from product perspective

5.7.4 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.

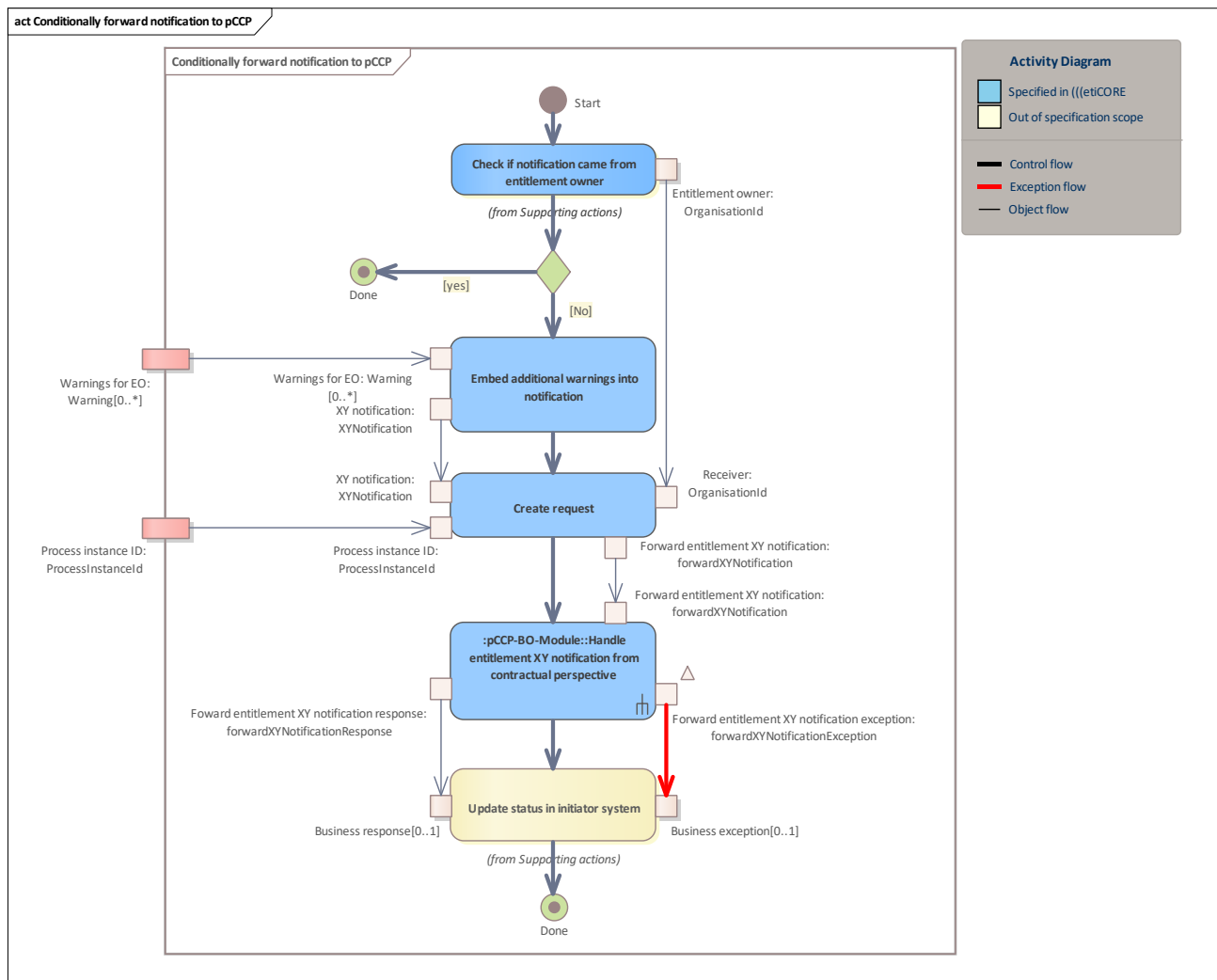


Figure 39: Conditionally forward notification to pCCP

5.7.4.1 Embed additional warnings into notification

Embed the given warnings into the notification currently being handled.

5.8 Handle entitlement XY notification from contractual perspective

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

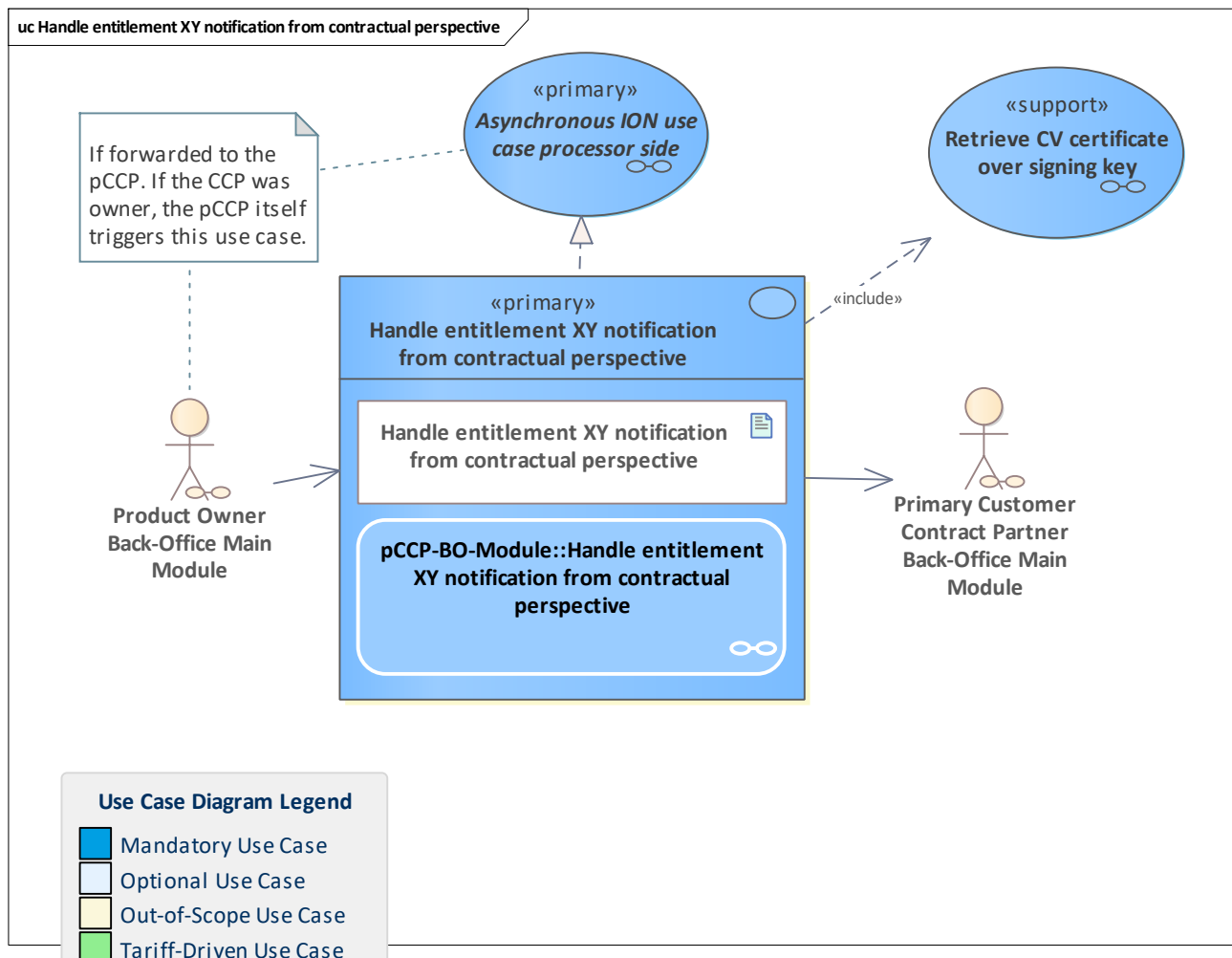


Figure 40: Handle entitlement XY notification from contractual perspective

5.8.1 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.

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

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

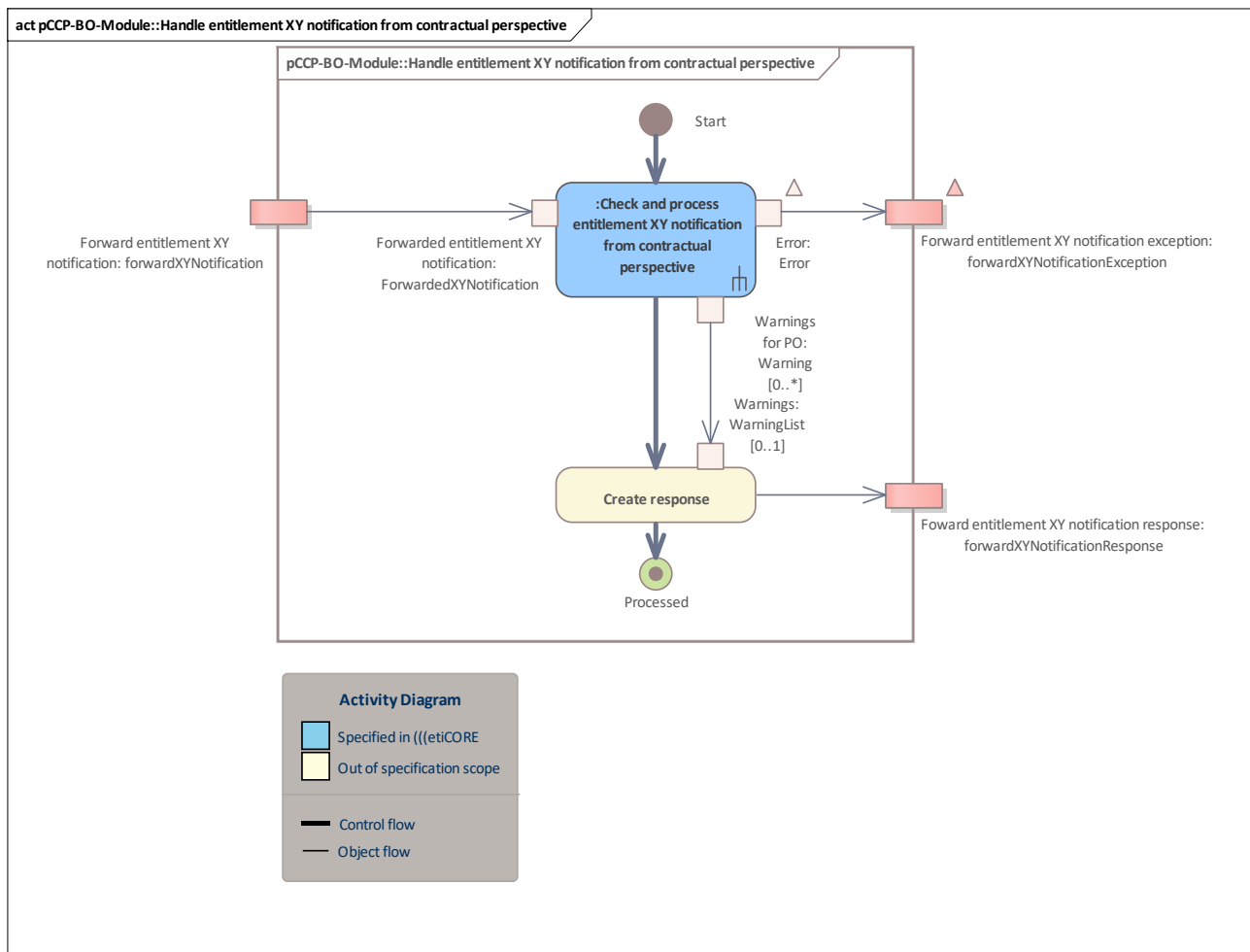


Figure 41: pCCP-BO-Module::Handle entitlement XY notification from contractual perspective

5.8.3 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.

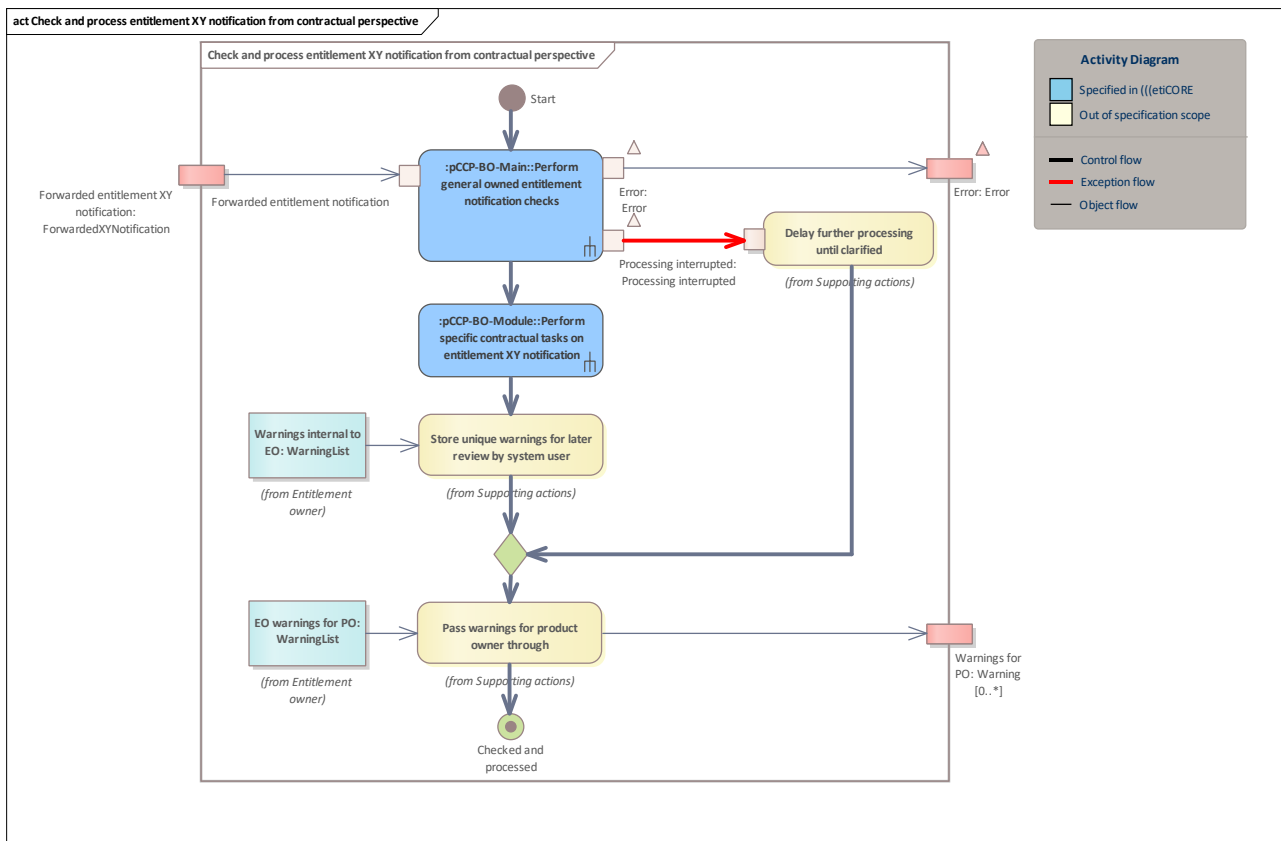


Figure 42: Check and process entitlement XY notification from contractual perspective

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

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

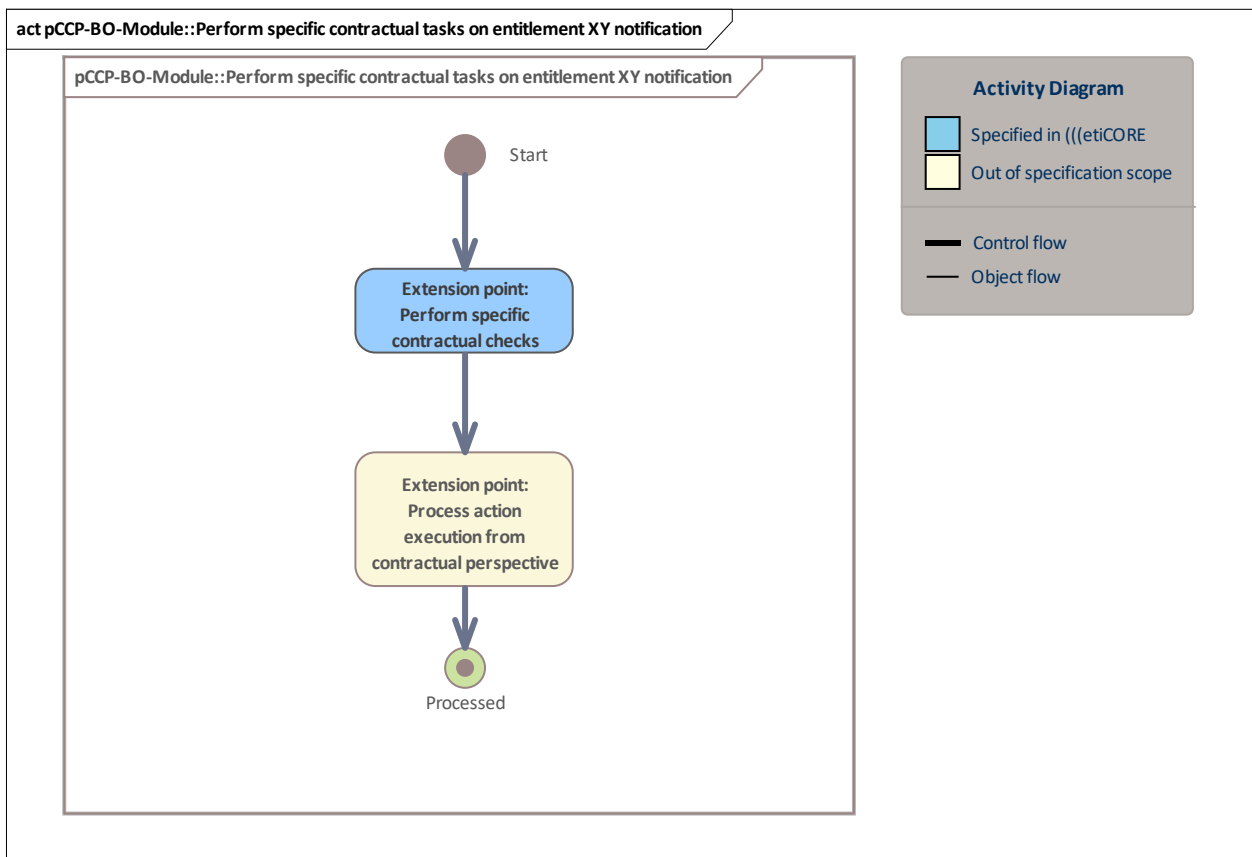


Figure 43: pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification

5.8.4.1 Extension point: Process action execution from contractual perspective

Update sales register, update entitlement state, etc.

5.9 Handle application XY notification from contractual perspective

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

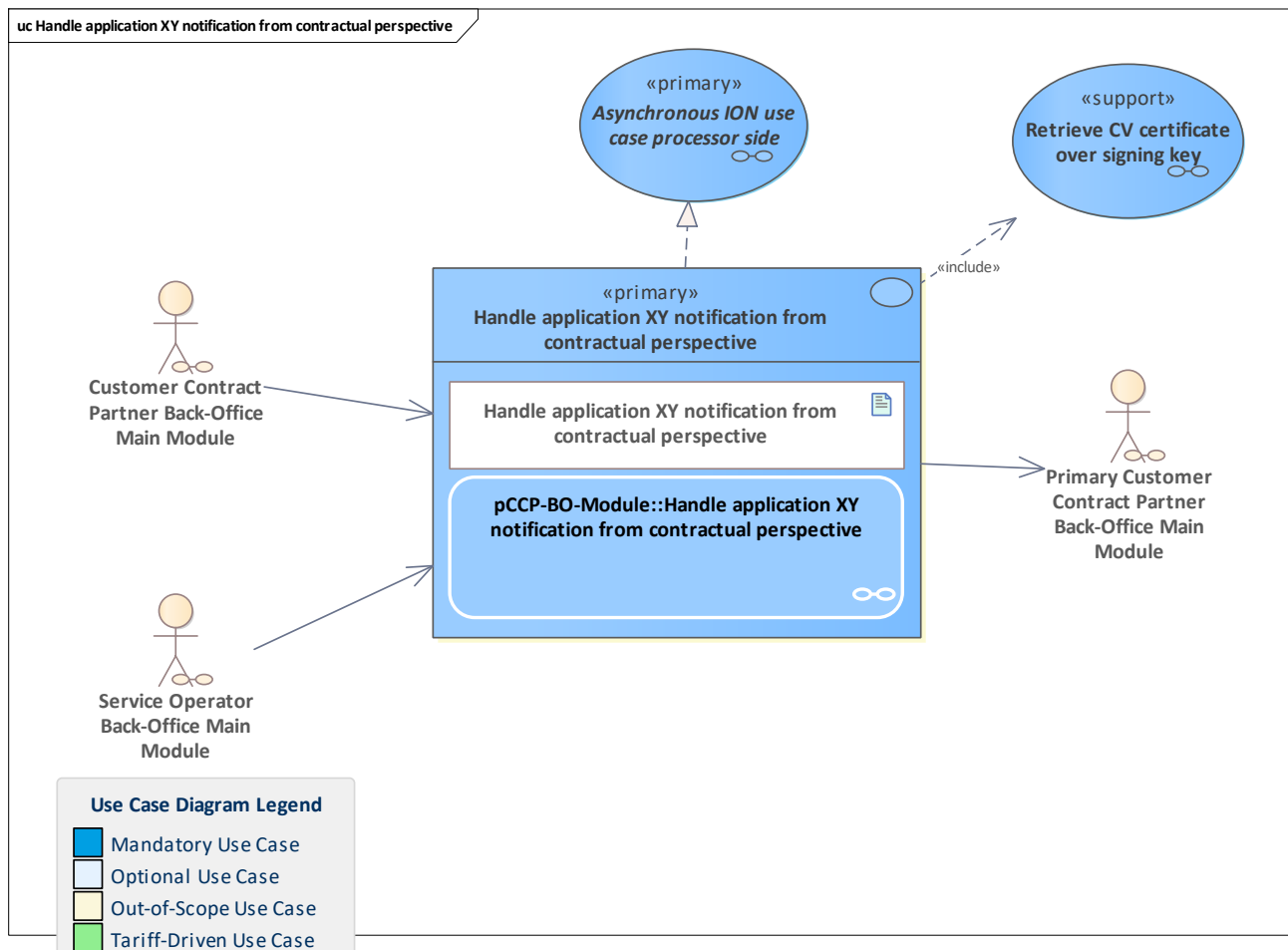


Figure 44: Handle application XY notification from contractual perspective

5.9.1 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.

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

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

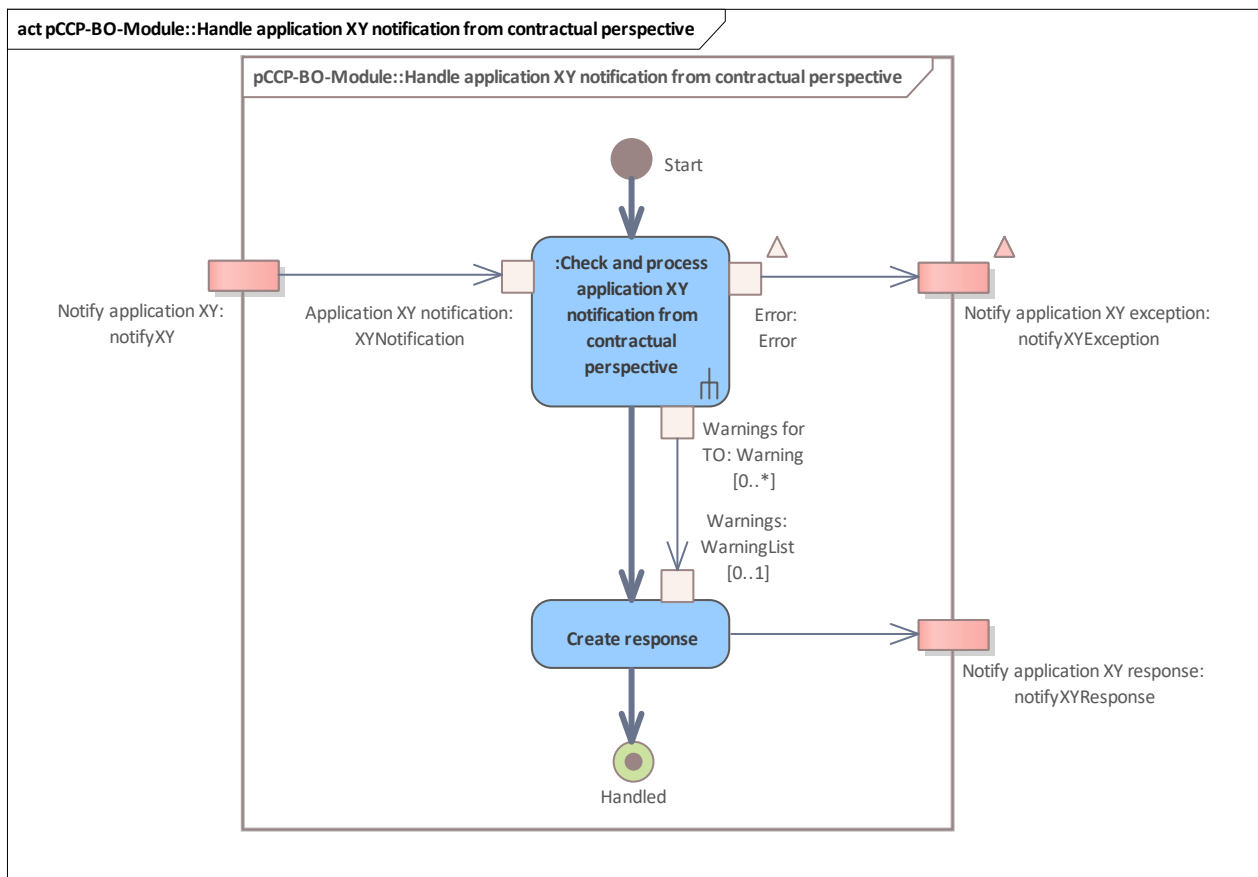


Figure 45: pCCP-BO-Module::Handle application XY notification from contractual perspective

5.9.3 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.

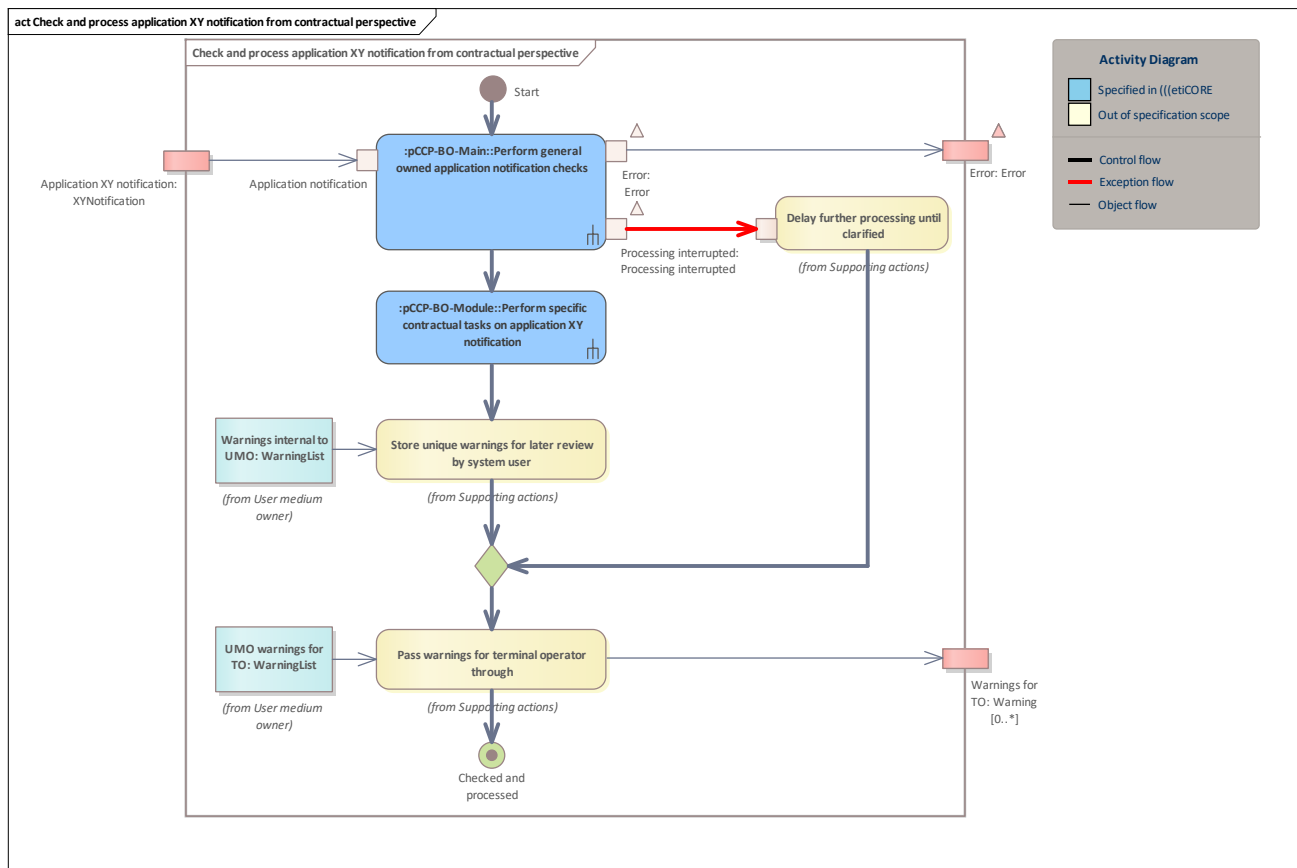


Figure 46: Check and process application XY notification from contractual perspective

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

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

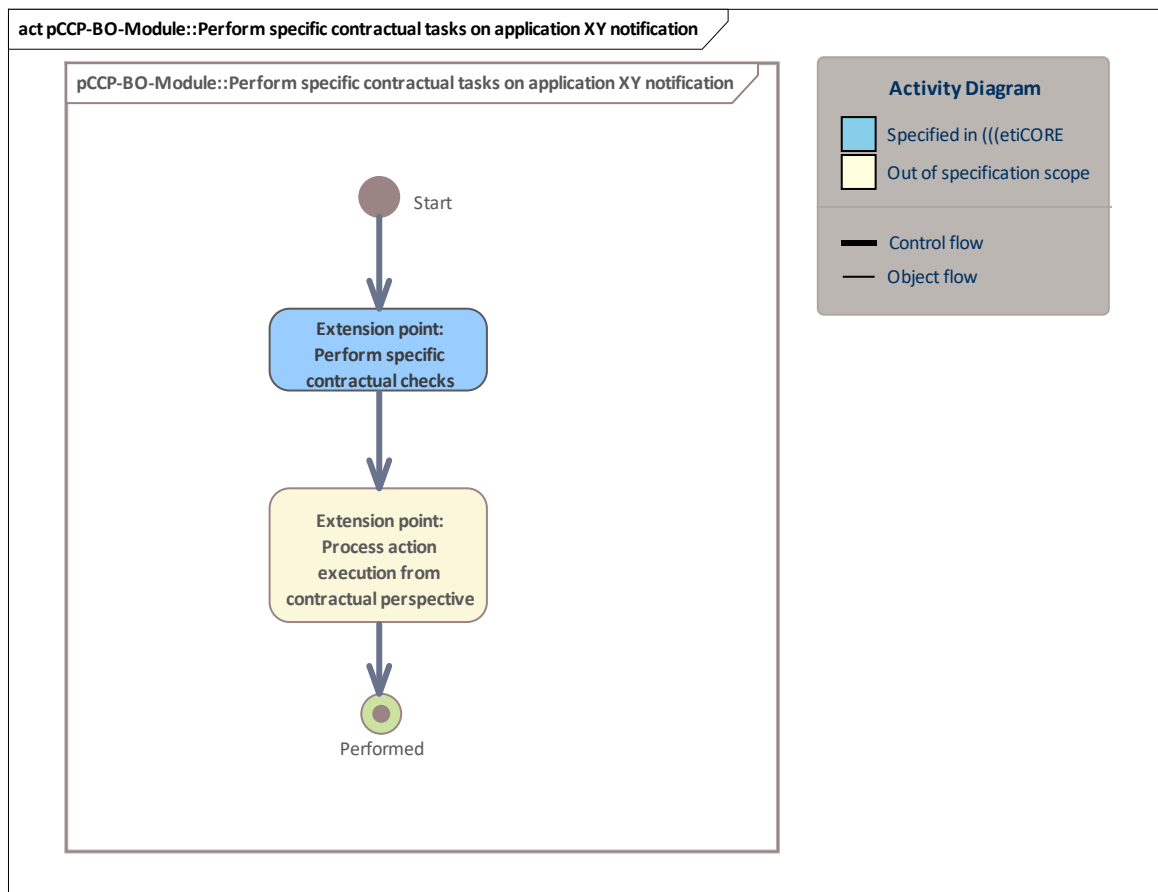


Figure 47: pCCP-BO-Module::Perform specific contractual tasks on application XY notification

6 Basic Bundle Terminal - Foundation

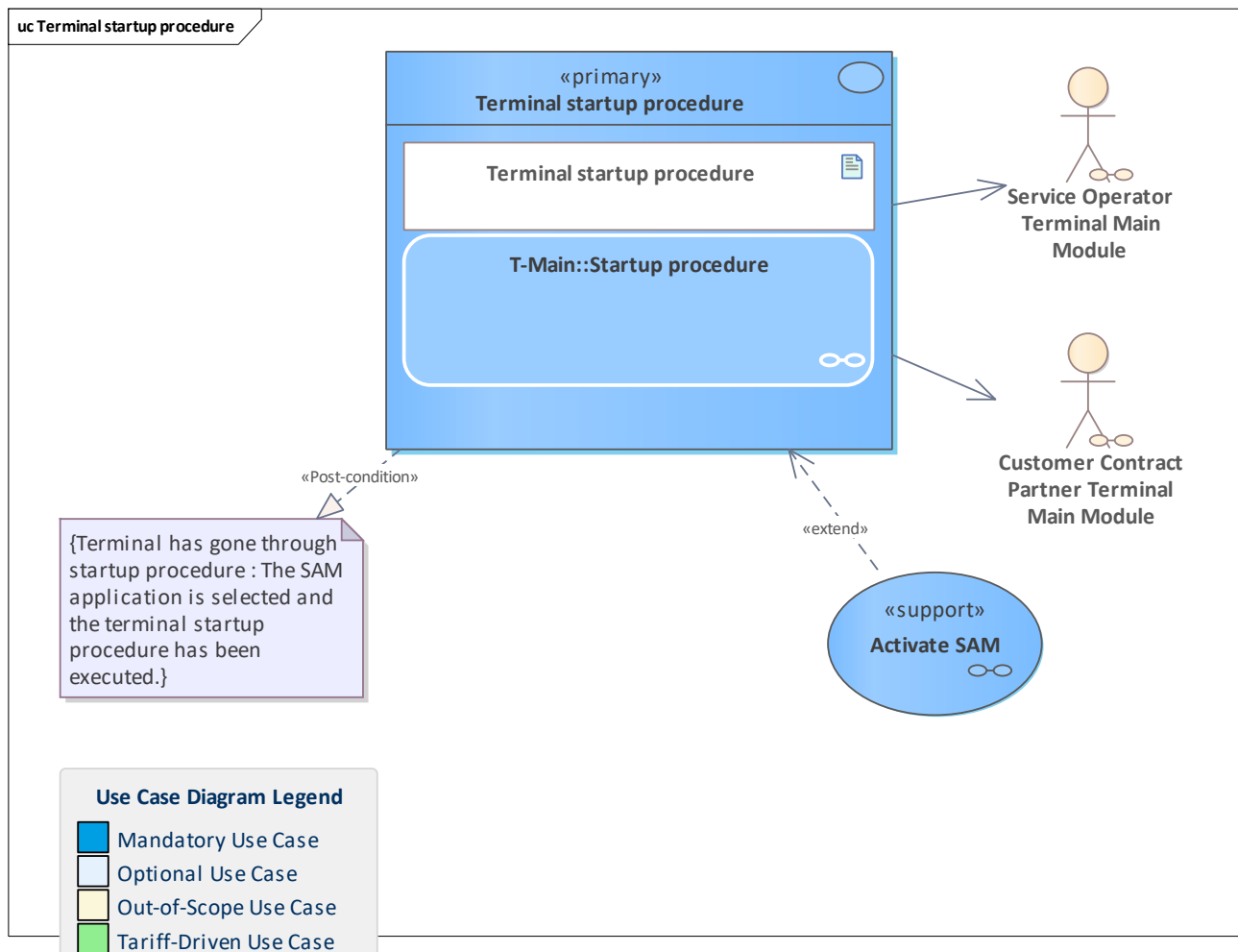
Basic functionality bundle that covers the use cases which have to be implemented for all terminals.

6.1 Overview

[Terminal startup procedure](#)
[Update SAM configuration](#)
[Optional: Update SAM reset data](#)
[Update terminal hotlists](#)
[Update organisation list](#)
[Update tariff module](#)
[Update CA certificate repository](#)
[Update CV certificate revocation list](#)

6.2 Use Cases

6.2.1 Terminal startup procedure

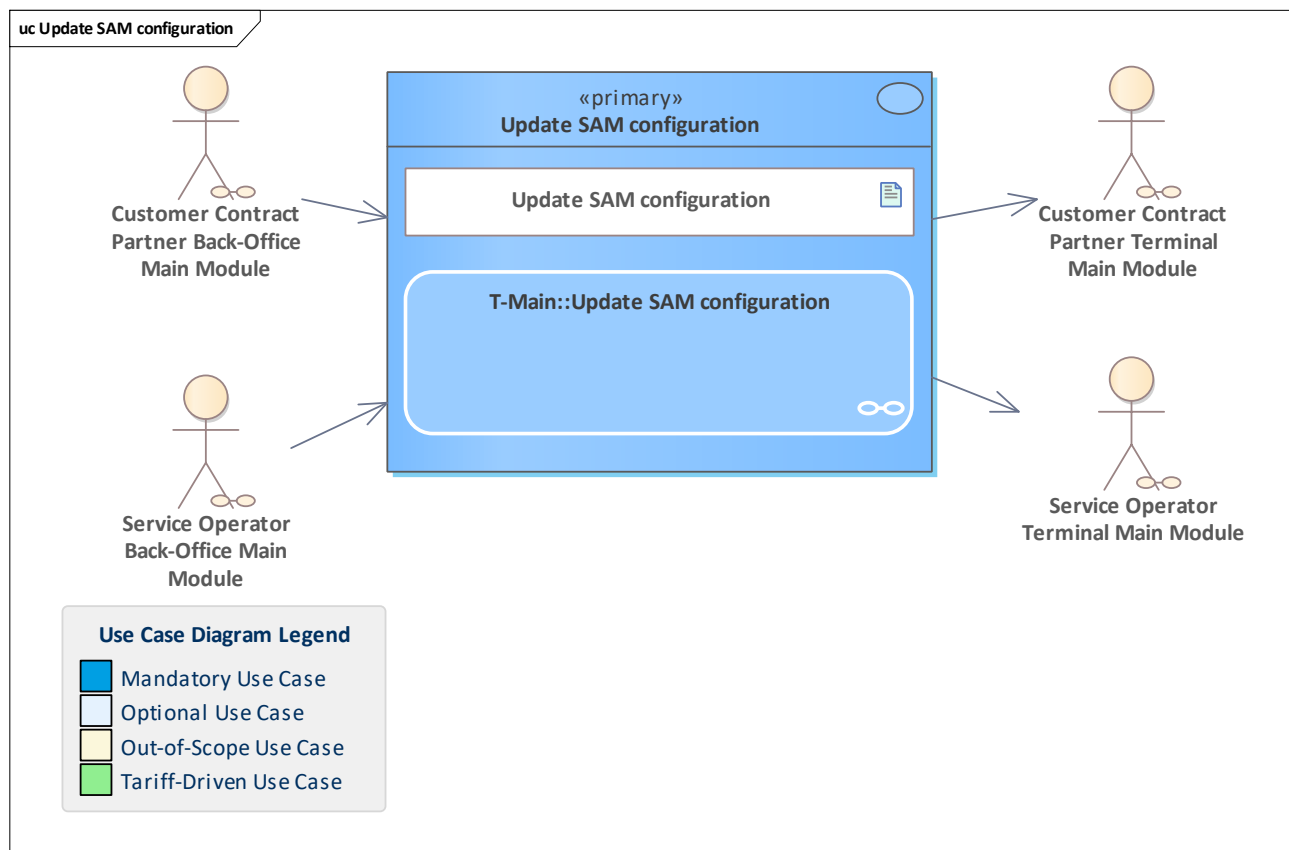


Use Case	Terminal startup procedure
Description	<p>This startup procedure brings the terminal into an operational state, initialising the SAM and caching all relevant information about it for use in UM interactions.</p> <p>It is used</p> <ul style="list-style-type: none"> to initially start the terminal, when the SAM has been replaced, when the Action operator ID has been re-configured, when the hotlists have been updated, when new SAM configuration scripts are available, and periodically (e.g. every 24 hours) to recheck validity periods
Initiating Actor	
Reacting Actor	Service Operator Terminal Main Module Customer Contract Partner Terminal Main Module
Preconditions	Terminal has gone through startup procedure
Postconditions	Terminal has gone through startup procedure
Linked Use Cases (Extended By)	Activate SAM
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Terminal has gone through startup procedure
Base Activity	



Inputs	
Outputs	
Error Cases	
Activity Diagram	T-Main::Startup_procedure

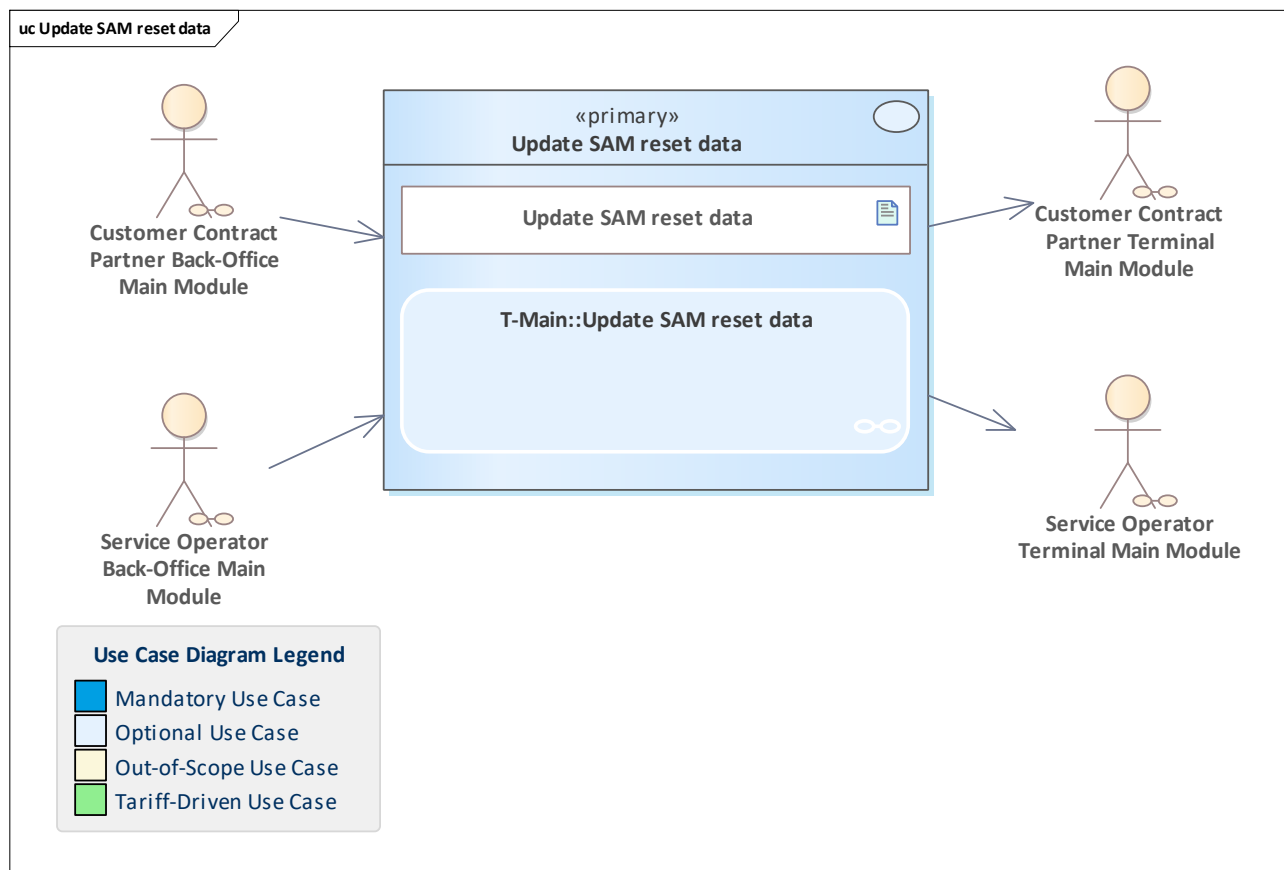
6.2.2 Update SAM configuration



Use Case	Update SAM configuration
Description	The SAM configuration data is updated in the terminal.
Initiating Actor	Customer Contract Partner Back-Office Main Module Service Operator Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Configure SAM for terminal : tConfigureSam
Outputs	Inform about configured SAM from terminal : tConfigureSamResponse
Error Cases	Configure SAM exception : tConfigureSamException
Activity Diagram	T-Main::Update SAM configuration



6.2.3 Optional: Update SAM reset data

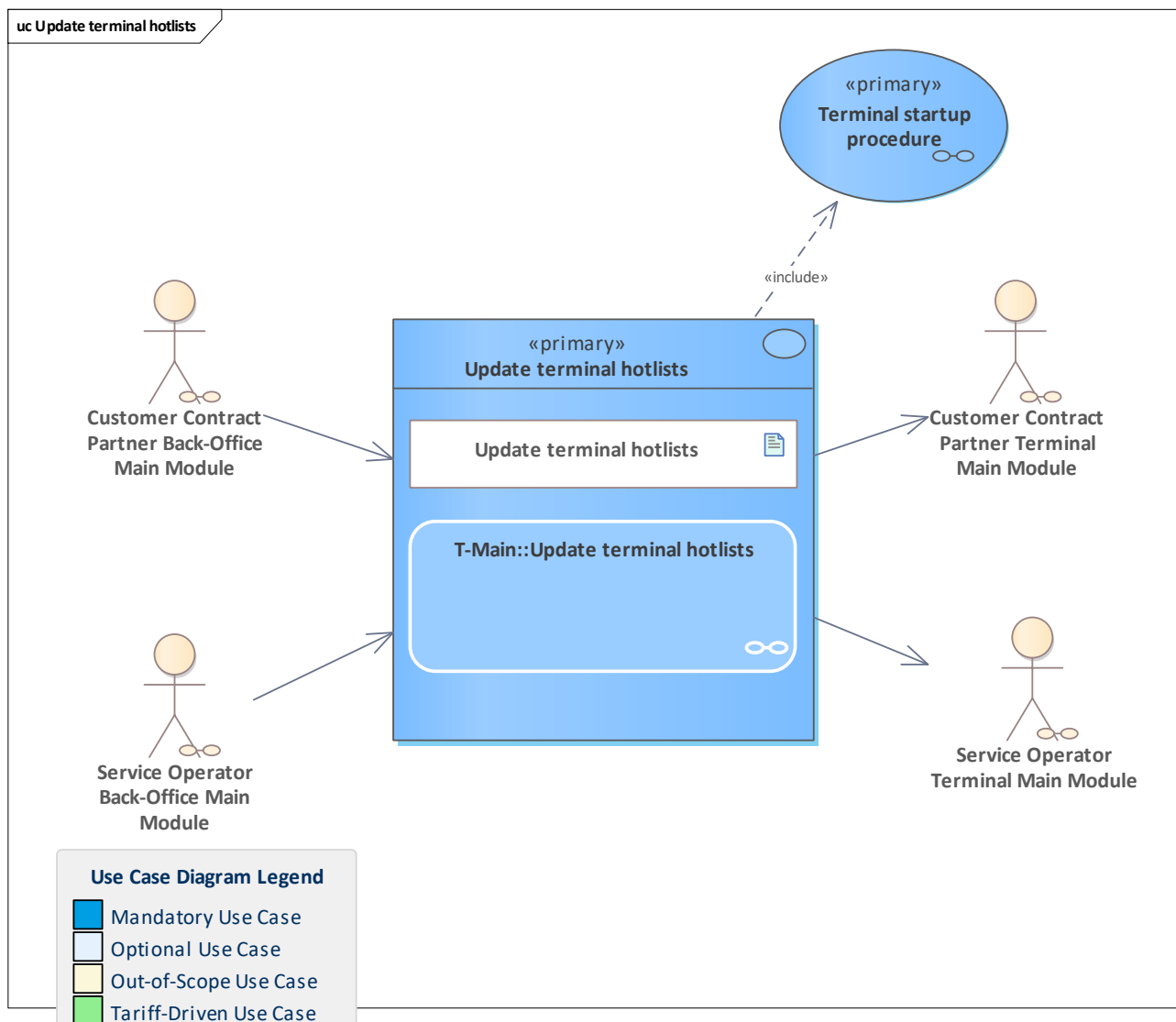


Use Case	Update SAM reset data
Description	The SAM reset data is updated in the terminal. Please note that after the SAM is reset, it cannot be used any more in ((etiCORE processes without being re-configured.
Initiating Actor	Customer Contract Partner Back-Office Main Module Service Operator Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Configure SAM for terminal : tResetSam
Outputs	Inform about reset SAM from terminal : tResetSamResponse
Error Cases	Reset SAM exception : tResetSamException



Activity Diagram	T-Main::Update SAM reset data
-------------------------	---

6.2.4 Update terminal hotlists

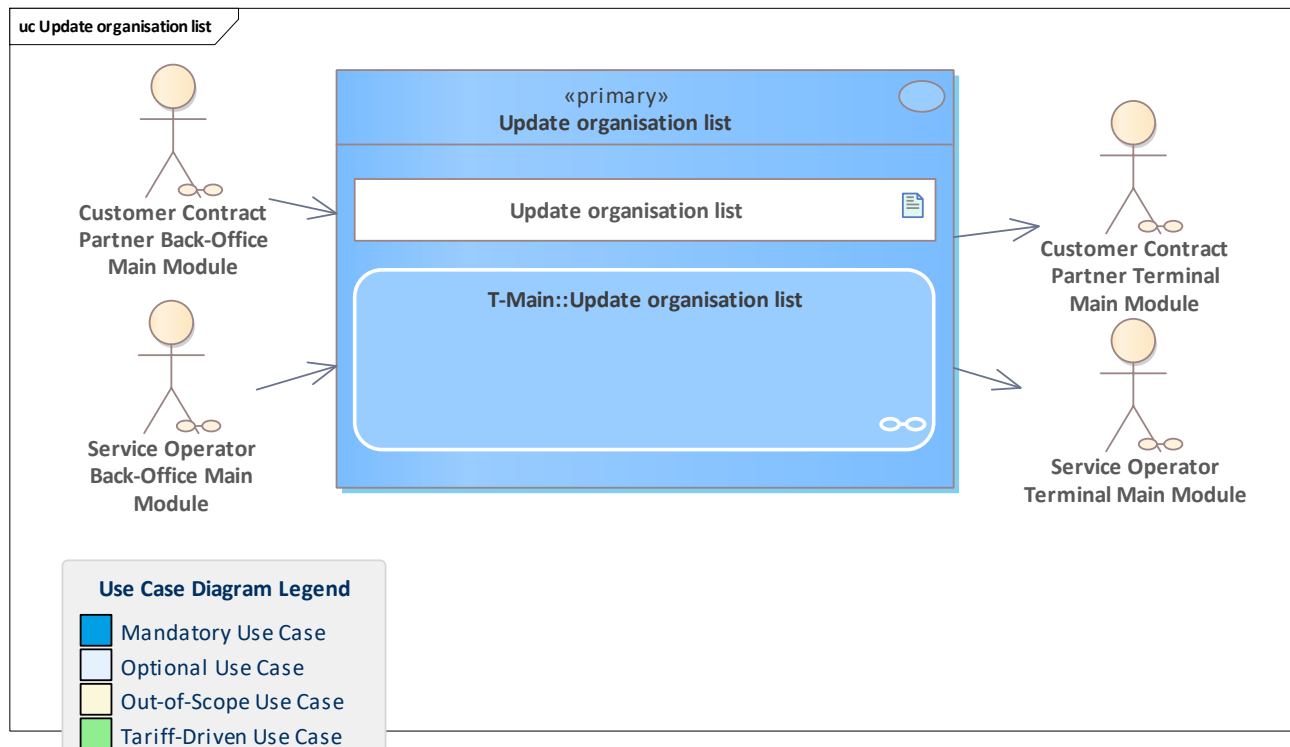


Use Case	<u>Update terminal hotlists</u>
Description	<p>Use case for terminal operators (CCP or SO). Once per cycle, the hotlists on the terminals have to be updated or replaced with new ones.</p> <p>Depending on the terminal architecture, after the update, the terminal startup process has to be performed.</p> <p>Please note that it is assumed that all hotlists (entitlement, application, SAM, authentication key and organisation hotlist) are gathered together and pushed in one operation to a terminal. Retrieval of each hotlist contains its process instance ID. Pushing all hotlists into the terminal takes a new process instance ID. Furthermore, it is assumed that the incremental hotlists are integrated into the hotlist inventory of the back-office system and the updated status is transferred to the terminal(s). Applying the incremental hotlists would also be possible directly in the terminal. This is not described here.</p>
Initiating Actor	<u>Customer Contract Partner Back-Office Main Module</u>



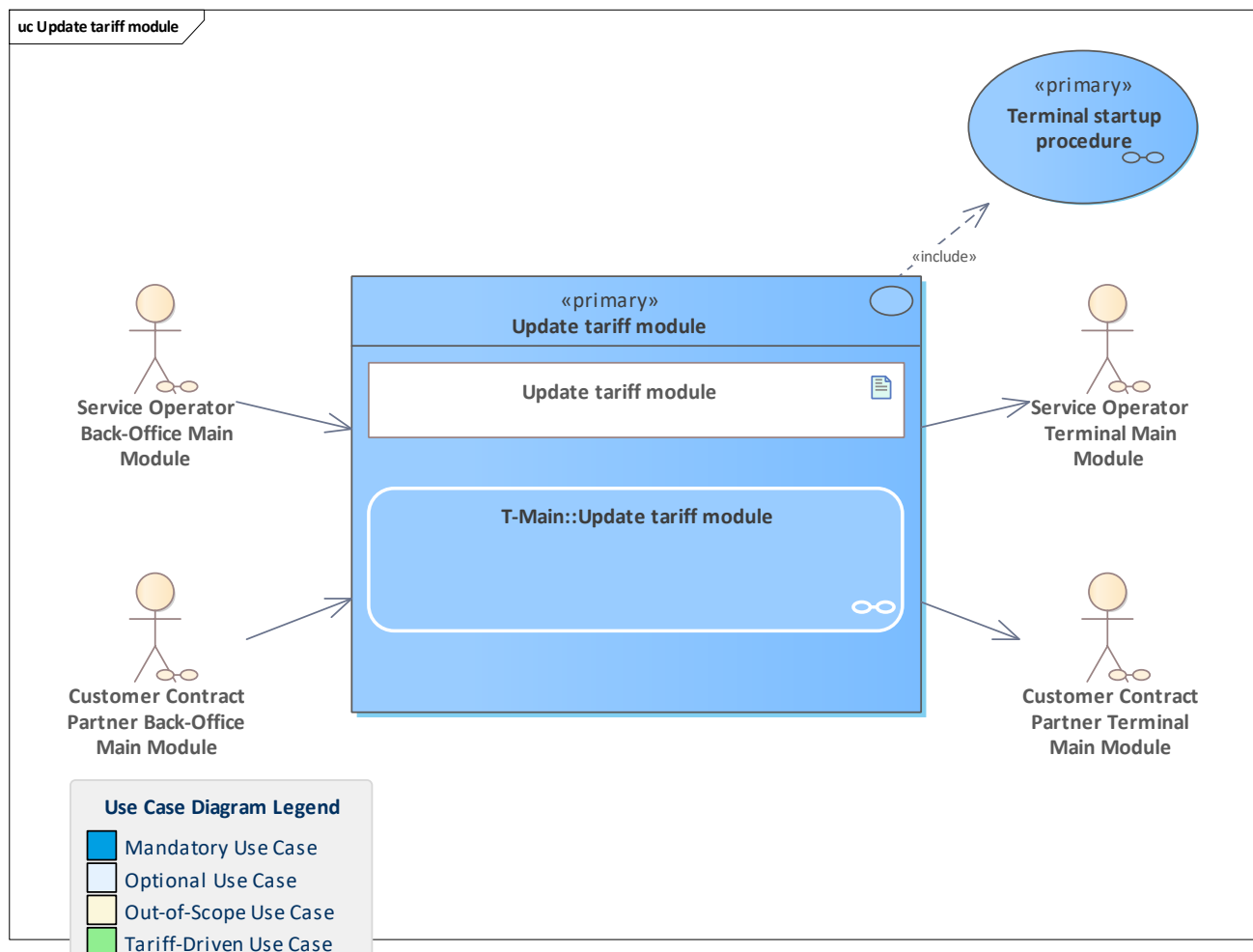
	Service Operator Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Terminal startup procedure
Linked Use Cases (Realises)	
Base Activity	
Inputs	tHotlists : tHotlists
Outputs	Terminal hotlists response : tHotlistsResponse
Error Cases	Terminal hotlist exception : tHotlistsException
Activity Diagram	T-Main::Update terminal hotlists

6.2.5 Update organisation list



Use Case	Update organisation list
Description	The organisation information list is updated in the terminal by replacing the old one with new one.
Initiating Actor	Customer Contract Partner Back-Office Main Module Service Operator Back-Office Main Module
Reacting Actor	Service Operator Terminal Main Module Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Organisation information list : tOrganisationList
Outputs	Terminal organisation list response : tOrganisationListResponse
Error Cases	Terminal organisation list exception : tOrganisationListException
Activity Diagram	T-Main::Update organisation list

6.2.6 Update tariff module

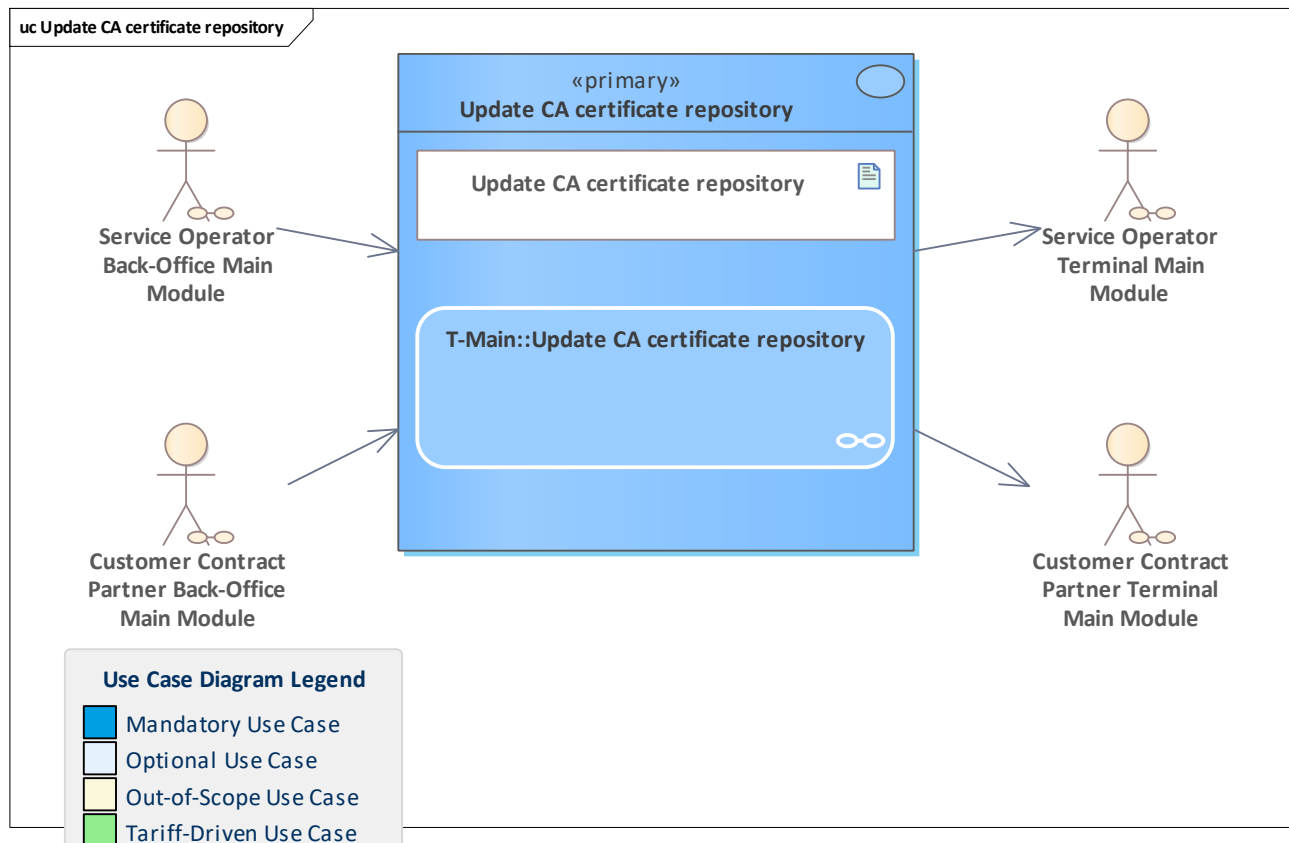


Use Case	Update tariff module
Description	Updates the tariff module in the terminal.
Initiating Actor	Service Operator Back-Office Main Module Customer Contract Partner Back-Office Main Module
Reacting Actor	Service Operator Terminal Main Module Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Terminal startup procedure
Linked Use Cases (Realises)	
Base Activity	
Inputs	Terminal tariff module : tTariffModule
Outputs	Terminal tariff module response : tTariffModuleResponse
Error Cases	Terminal tariff module exception : tTariffModuleException



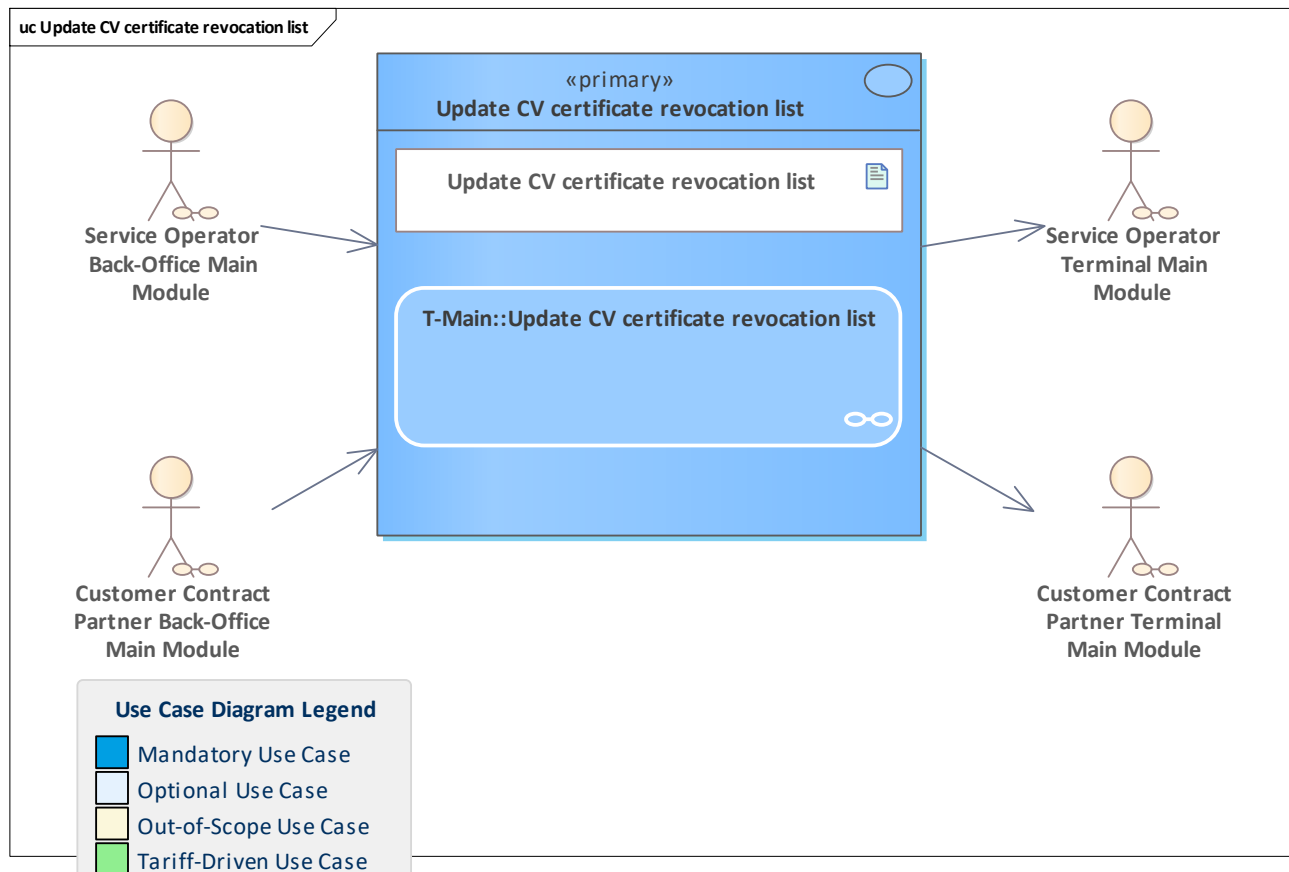
Activity Diagram	T-Main::Update tariff module
-------------------------	--

6.2.7 Update CA certificate repository



Use Case	Update CA certificate repository
Description	Updates the CA certificate repository in the terminal.
Initiating Actor	Customer Contract Partner Back-Office Main Module Service Operator Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Terminal CA certificate repository : tCaCertificateRepository
Outputs	Terminal CA certificate repository response : tCaCertificateRepositoryResponse
Error Cases	Terminal CA certificate repository exception : tCaCertificateRepositoryException
Activity Diagram	T-Main::Update CA certificate repository

6.2.8 Update CV certificate revocation list



Use Case	Update CV certificate revocation list
Description	Updates the CV certificate revocation list in the terminal.
Initiating Actor	Service Operator Back-Office Main Module Customer Contract Partner Back-Office Main Module
Reacting Actor	Service Operator Terminal Main Module Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Terminal CV certificate revocation list : tCvCertificateRevocationList
Outputs	Terminal CV certificate revocation list response : tCvCertificateRevocationListResponse
Error Cases	Terminal CV certificate revocation list exception : tCvCertificateRevocationListException
Activity Diagram	T-Main::Update CV certificate revocation list





7 Basic Bundle Terminal - UM with Application

Functionality bundle that covers all use cases forming the basis for handling user media with an application (e.g. chip cards).

7.1 Overview

Get entitlement and check attestations

Establish session based on certificates

Establish session and get entitlement directory

Export derived key

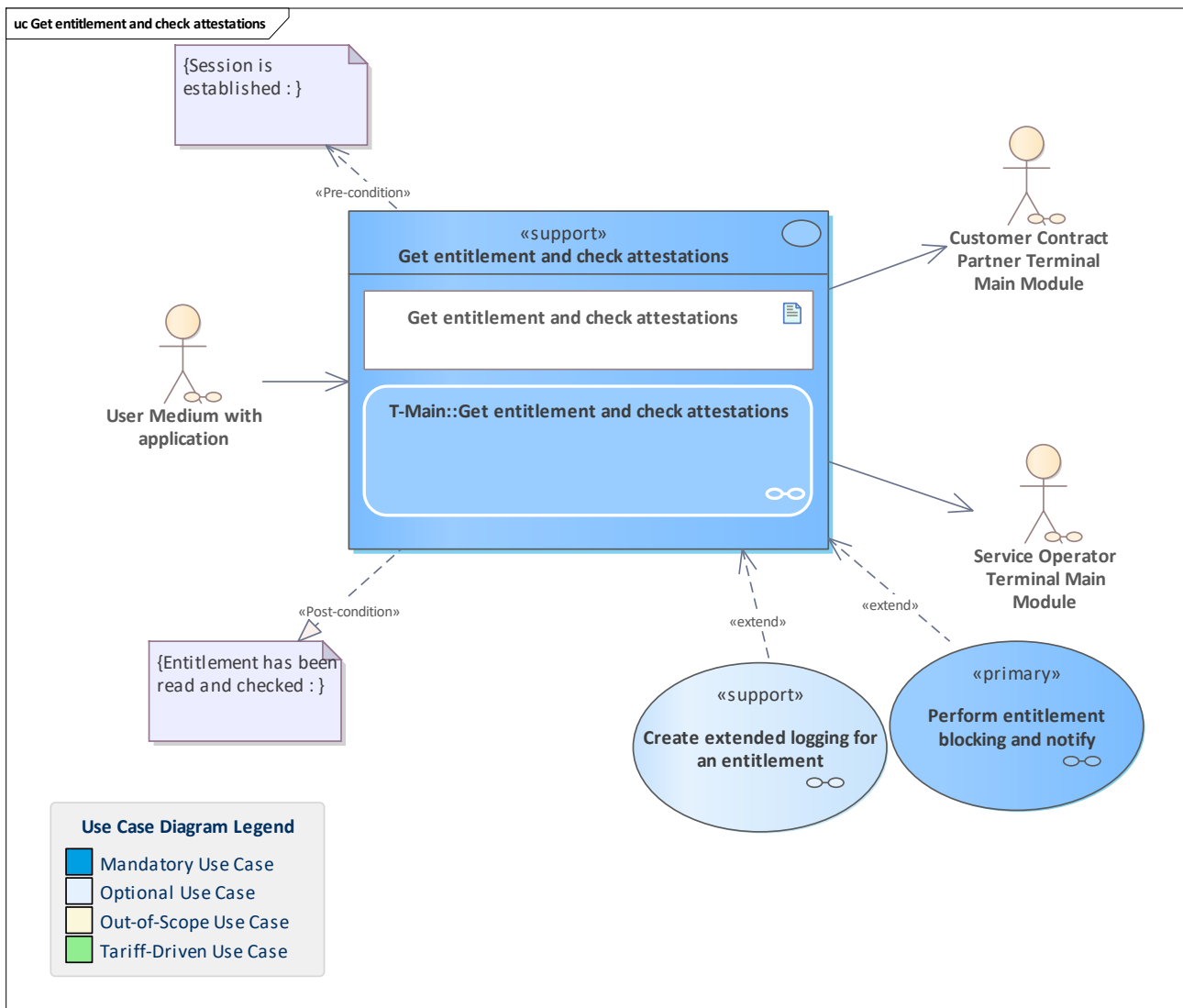
Perform application blocking and notify

Perform entitlement blocking and notify

Optional: Log defective user medium with application

7.2 Use Cases

7.2.1 Get entitlement and check attestations

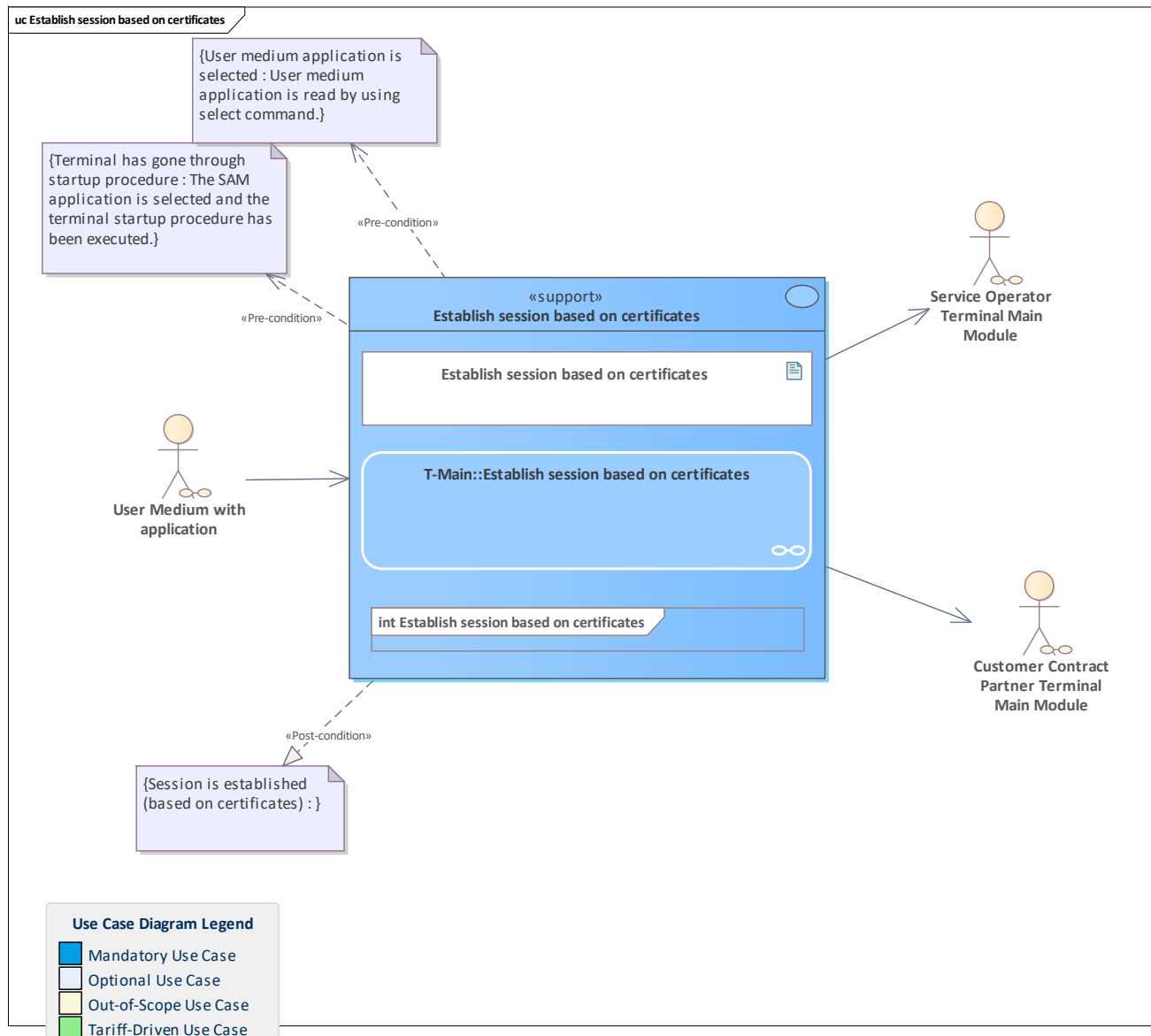


Use Case	Get entitlement and check attestations
Description	This supporting use case describes that <ul style="list-style-type: none"> the requested entitlement is read from the user medium and validated available attestations are checked against the hotlist in case of a hotlist match, the entitlement is blocked
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	Session is established Entitlement has been read and checked Session is established
Postconditions	Entitlement has been read and checked
Linked Use Cases (Extended By)	Perform entitlement blocking and notify / Create extended logging for an entitlement
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Entitlement has been read and checked



Base Activity	
Inputs	<u>Entitlement directory entry : EntitlementDirectoryEntry</u>
Outputs	<u>Entitlement : Entitlement</u>
Error Cases	<u>Blocked entitlement</u>
Activity Diagram	<u>T-Main::Get entitlement and check attestations</u>

7.2.2 Establish session based on certificates

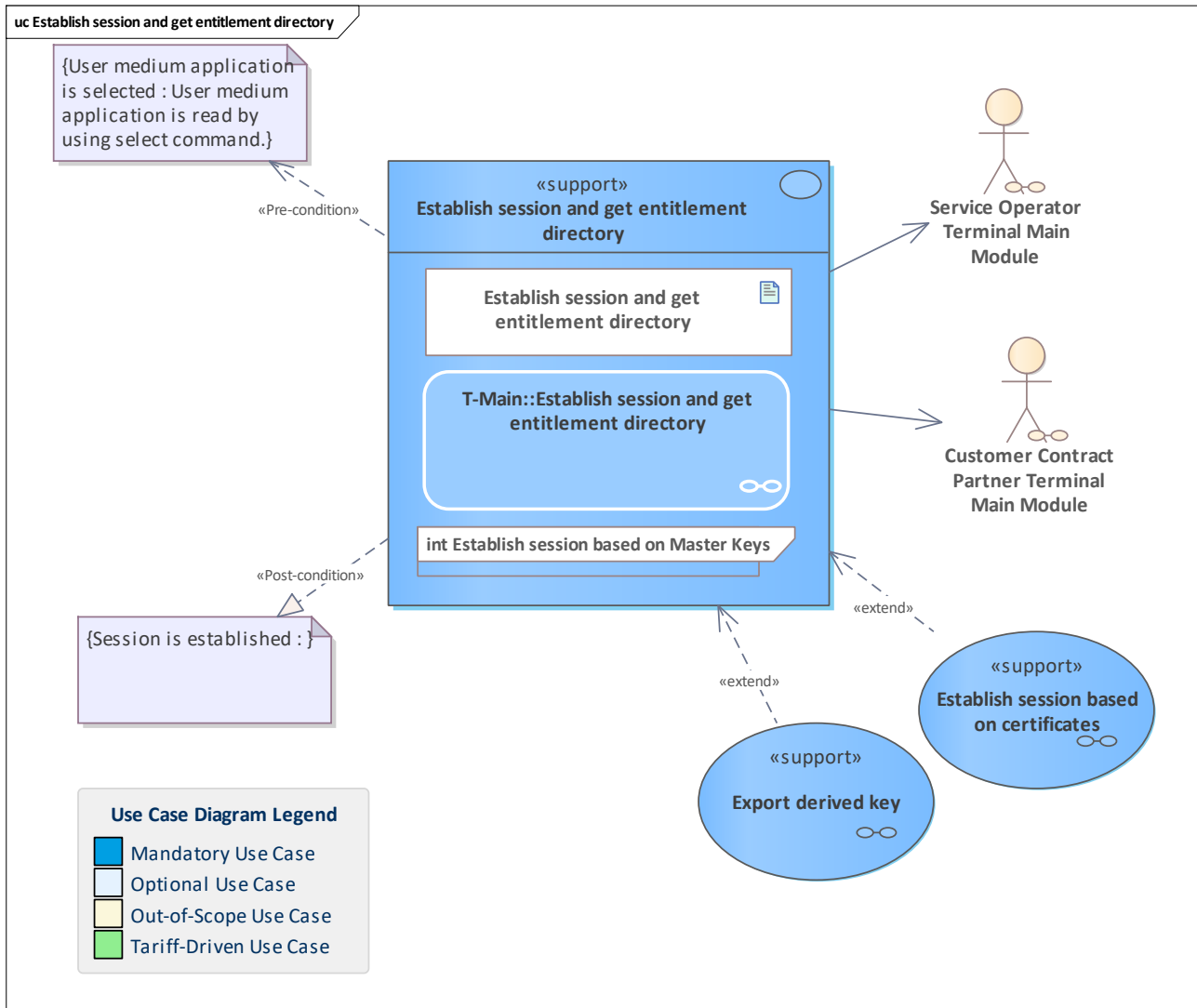


Use Case	<u>Establish session based on certificates</u>
Description	The terminal establishes a session between user medium and SAM based on certificates. This process can optionally be extended to use ephemeral keys to increase the security properties of the session. For the sake of simplicity, this is not shown. Whether to use ephemeral keys is at the discretion of the terminal.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Service Operator Terminal Main Module</u> <u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>Terminal has gone through startup procedure</u> <u>User medium application is selected</u>
Postconditions	<u>Session is established (based on certificates)</u>
Linked Use Cases (Extended By)	



Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Session is established (based on certificates)
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	T-Main::Establish session based on certificates

7.2.3 Establish session and get entitlement directory

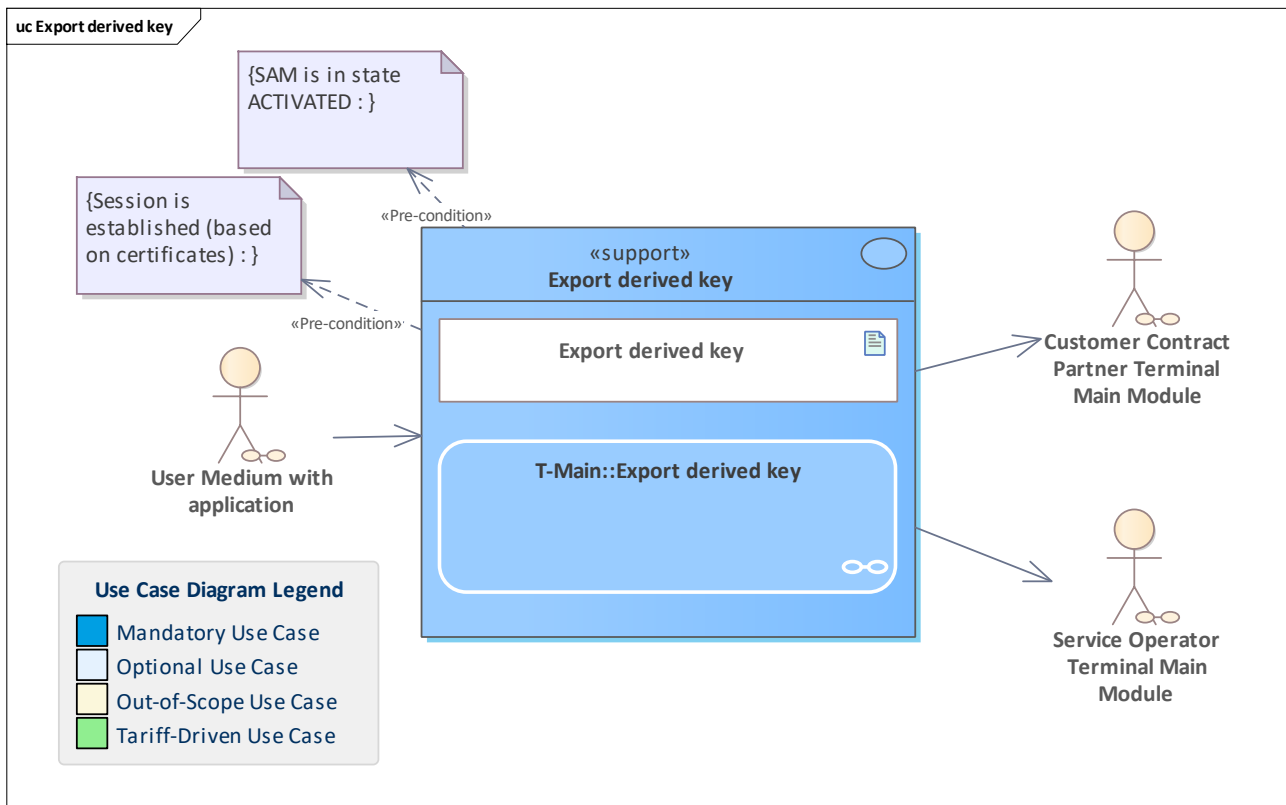


Use Case	<u>Establish session and get entitlement directory</u>
Description	The terminal establishes a session between the user medium and SAM and securely retrieves the entitlement directory. If possible, the session is established based on symmetric master keys. Otherwise, the process falls back to certificate-based session establishment and tries to derive a key for the user medium such that future sessions can be established using that key. The terminal checks that the core parts of the user medium application directory (initially retrieved via SELECT) are authentic.
Initiating Actor	
Reacting Actor	<u>Service Operator Terminal Main Module</u> <u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>User medium application is selected</u> <u>Session is established</u> <u>User medium application is selected</u>
Postconditions	<u>Session is established</u>



Linked Use Cases (Extended By)	Establish session based on certificates / Export derived key
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Session is established
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	T-Main::Establish session and get entitlement directory

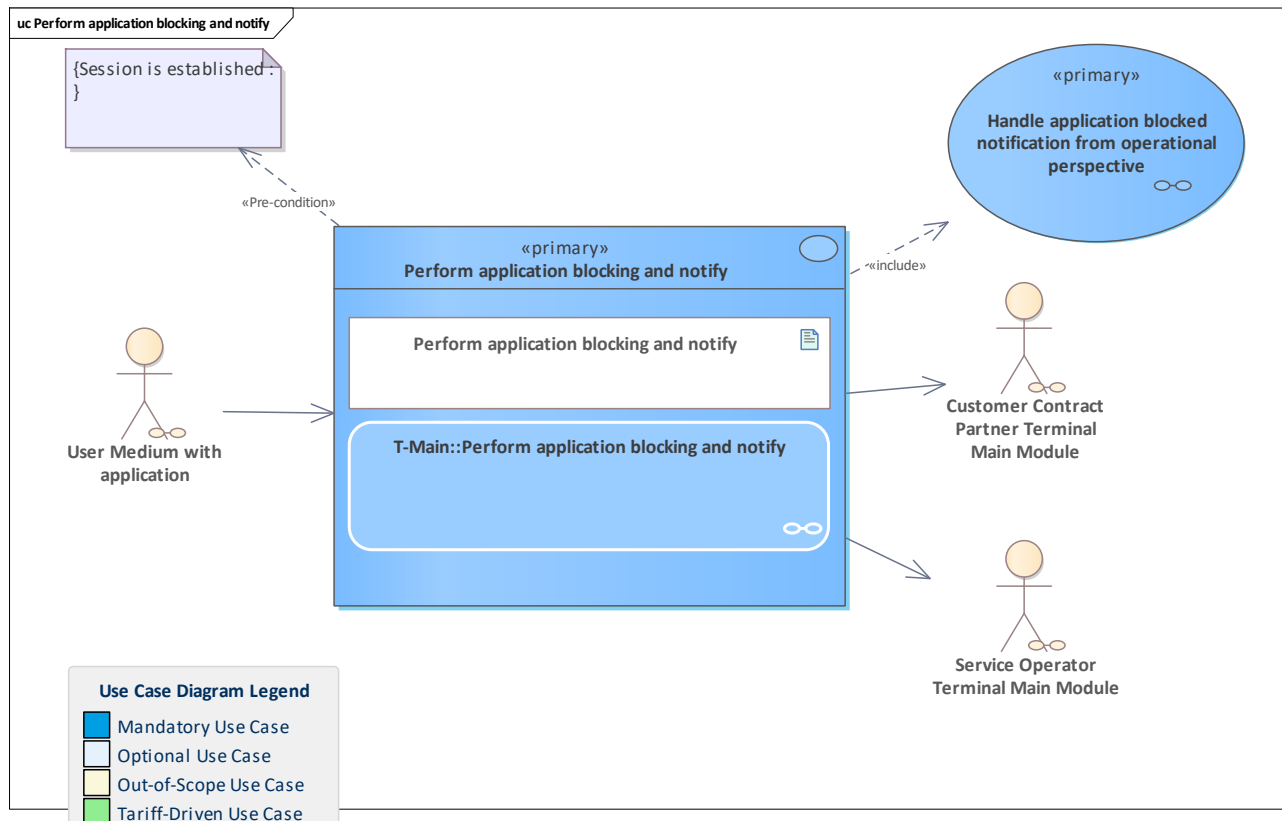
7.2.4 Export derived key



Use Case	Export derived key
Description	The terminal derives a key and exports it to the user medium application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established SAM is in state ACTIVATED Session is established (based on certificates)
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Key reference : KeyReference
Outputs	
Error Cases	
Activity Diagram	T-Main::Export derived key



7.2.5 Perform application blocking and notify

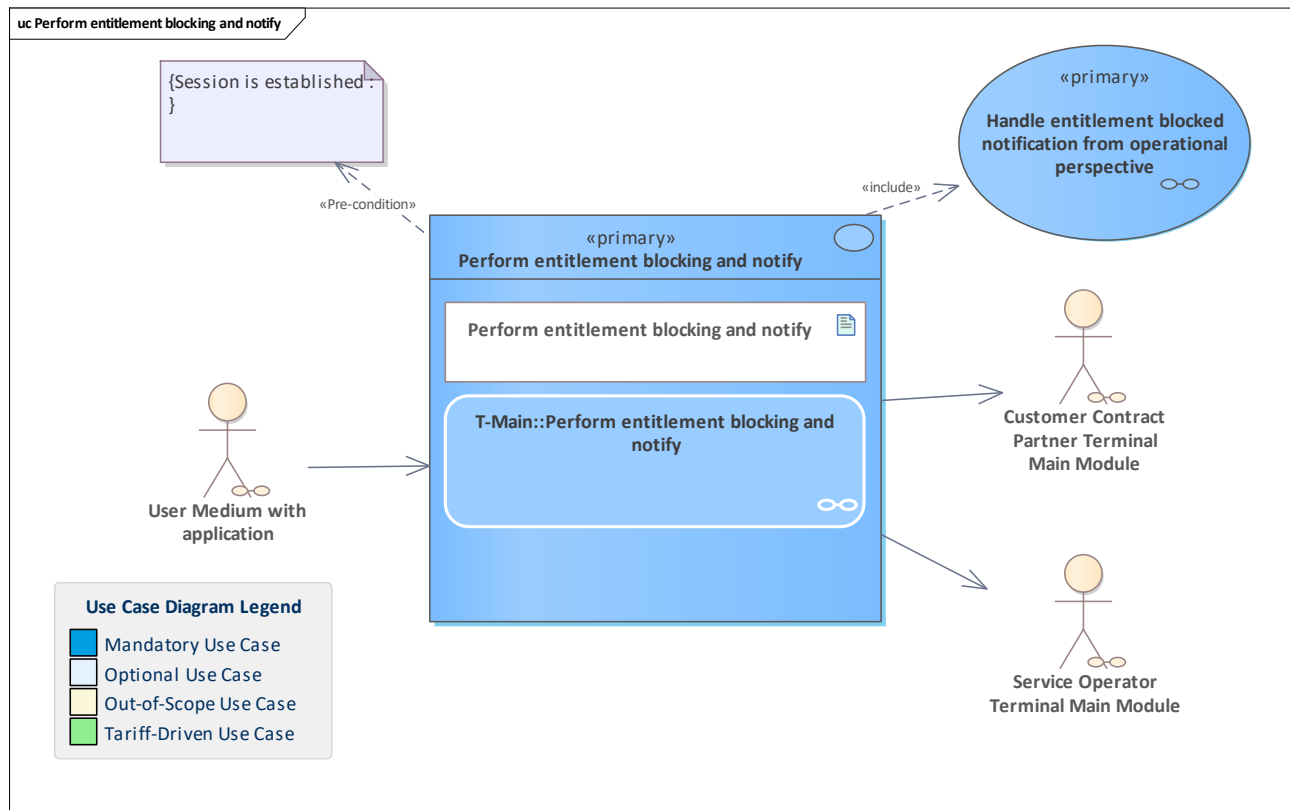


Use Case	Perform application blocking and notify
Description	<p>Use case for a CCP or SO terminal. A user medium application is detected either in the application hotlist or a relevant entry is found in the SAM hotlist or the organisation hotlist. Therefore, the user medium application must be blocked physically by switching its state.</p> <p>The terminal performs this action and notifies the back-office system of the terminal operator (CCP or SO).</p> <p>If the transaction is aborted, this has also to be notified to the terminal operator (CCP or SO).</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle application blocked notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	SAM Id : AppInstanceId Hotlist cycle number : ListCycleNumber Organisation ID : OrganisationId



Outputs	
Error Cases	
Activity Diagram	T-Main::Perform application blocking and notify

7.2.6 Perform entitlement blocking and notify

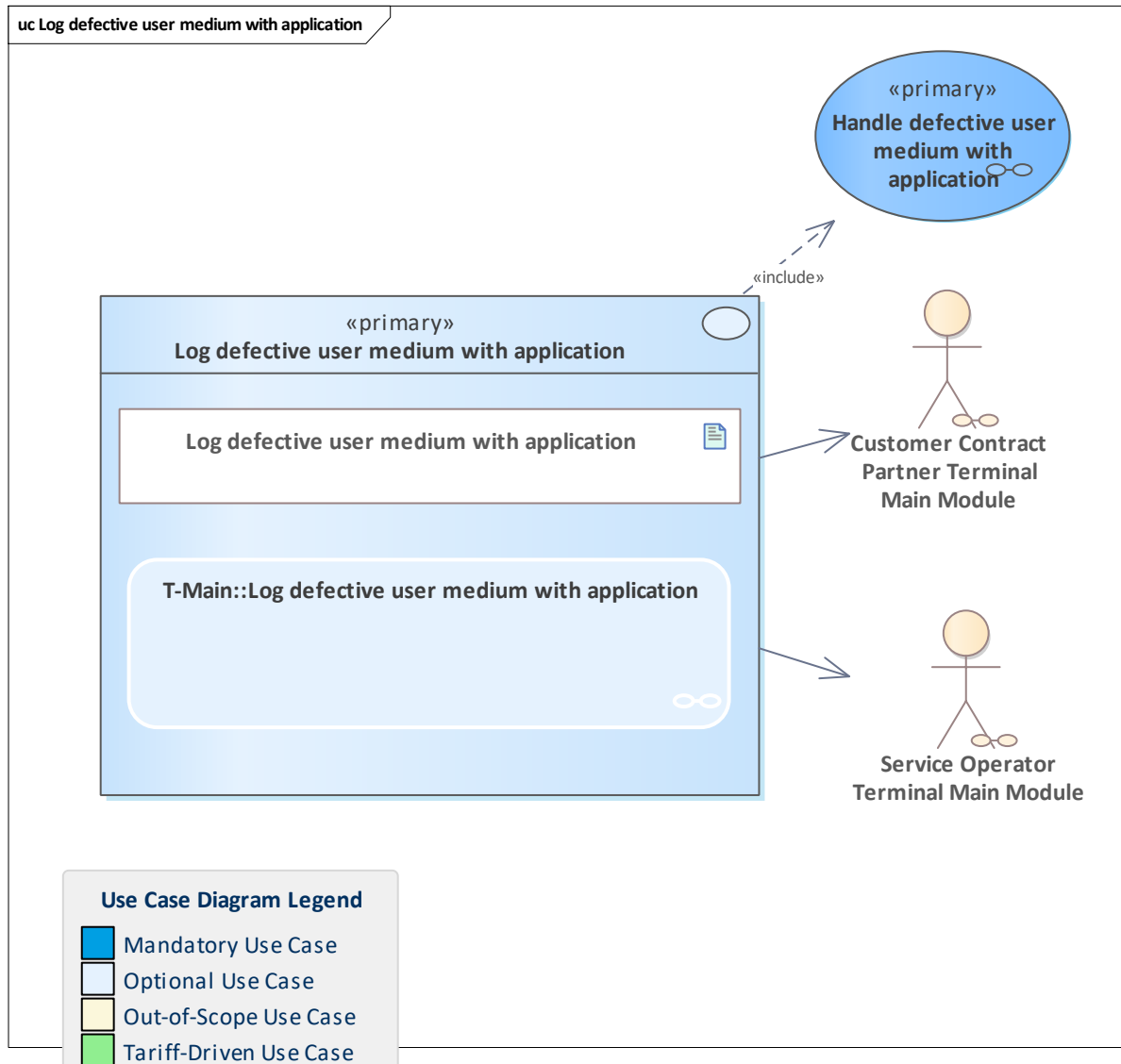


Use Case	Perform entitlement blocking and notify
Description	<p>Use case for a CCP or SO terminal.</p> <p>An entitlement on a user medium with an application is detected either in the entitlement hotlist or a relevant entry is found in the SAM hotlist or the organisation hotlist. Therefore, the entitlement must be blocked physically by switching its state.</p> <p>The terminal performs this action and notifies the back-office system of the terminal operator (CCP or SO).</p> <p>If the transaction is aborted, this has also to be notified to the terminal operator (CCP or SO).</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle entitlement blocked notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Hotlist cycle number : ListCycleNumber Org ID : OrganisationId SAM ID : AppInstanceId Entitlement directory entry : EntitlementDirectoryEntry



Outputs	
Error Cases	
Activity Diagram	T-Main::Perform entitlement blocking and notify

7.2.7 Optional: Log defective user medium with application



Use Case	Log defective user medium with application
Description	<p>If a user medium with an application is defective, the terminal action data and medium ID are registered and sent to the back-office system.</p> <p>The recording of the media ID must be supported by person-operated terminals in such a way that the organisation ID of the card manufacturer which individualises cards in the region can be pre-set in the terminal. It must also be possible to manually enter another organisation ID or alphanumeric characters (possibly the case for foreign media), rather than use the pre-set organisation ID. The pre-set ID must, therefore, be able to be overwritten! Separate entry fields are recommended for the string before the first hyphen (registration authority as organisation ID) and the string after the first hyphen (subject number as an integer,</p>



	<p>including the checksum digit, separated by a hyphen). The terminals must ensure error-tolerant input processing of the media ID, with or without full stops or hyphens. The last digit of the media ID can be a checksum digit, which is calculated according to the Luhn algorithm (calculation using the media ID without the last printed (entered) digit). The checksum digit is intended to prevent input errors during entry. It must be possible to complete the entry, even if the checksum digit is displayed as being incorrect.</p>
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle defective user medium with application
Linked Use Cases (Realises)	
Base Activity	
Inputs	Medium ID : MediumId
Outputs	
Error Cases	
Activity Diagram	T-Main::Log defective user medium with application

8 Basic Bundle Terminal - Extended Logging

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

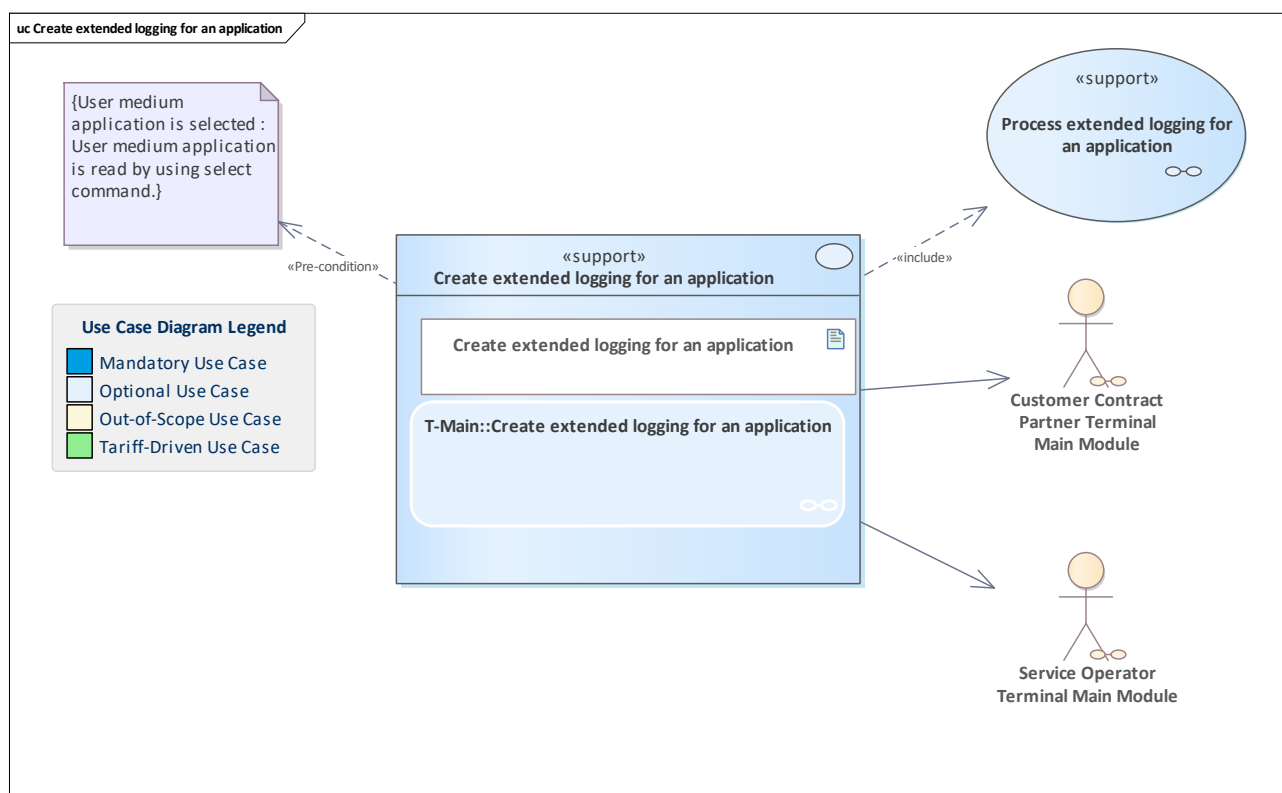
8.1 Overview

Optional: Create extended logging for an application

Optional: Create extended logging for an entitlement

8.2 Use Cases

8.2.1 Optional: Create extended logging for an application

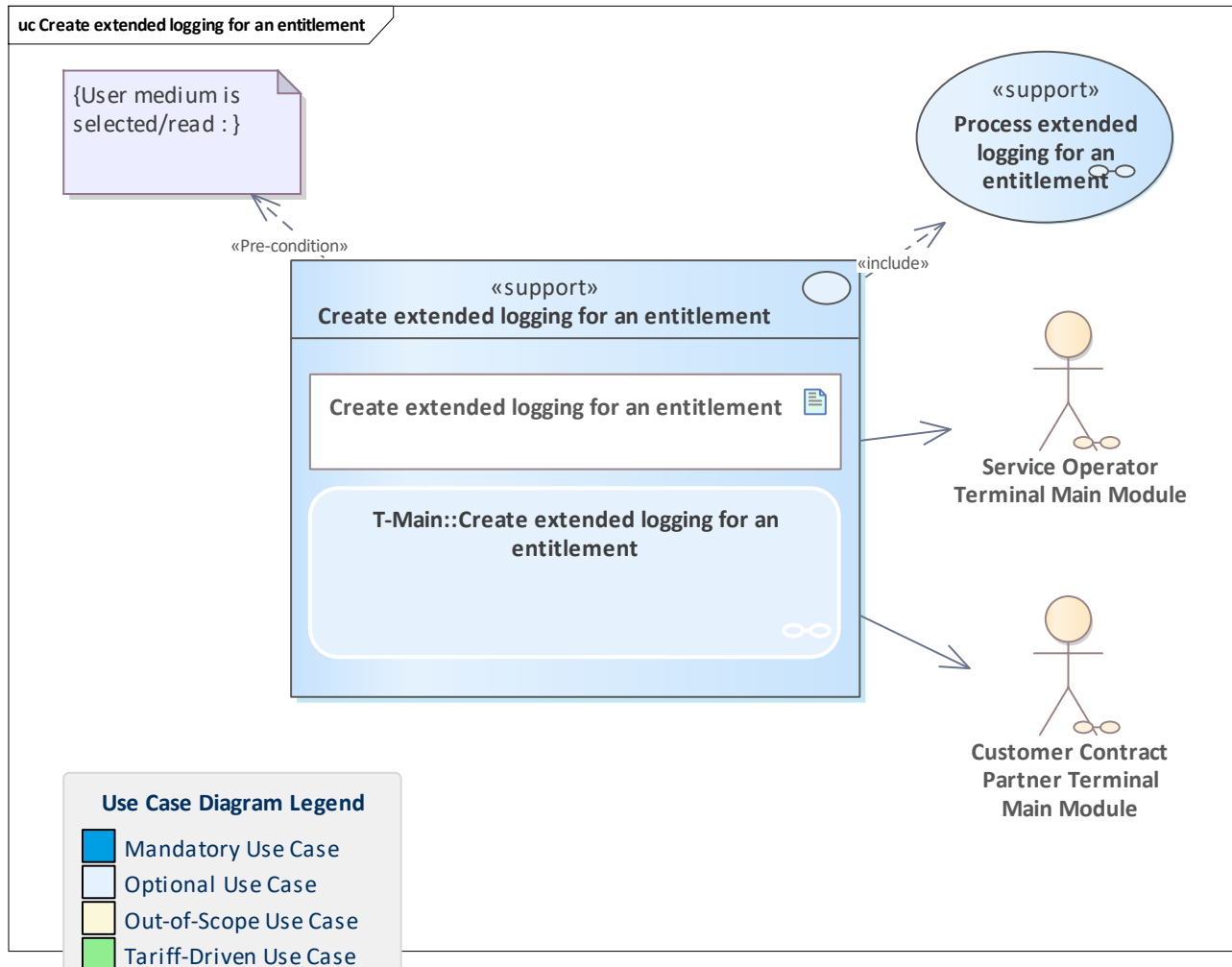


Use Case	<u>Create extended logging for an application</u>
Description	<p>For monitoring purposes, logging related to an invalid application should be created by a terminal and analysed by the back-office system.</p> <p>The extended logging of applications is triggered if a blocked or terminated application was presented or the validity period of the application has been expired.</p>



	The detailed reason can be found in ValidationEnum .
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	User medium application is selected
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Process extended logging for an application
Linked Use Cases (Realises)	
Base Activity	
Inputs	Validation code : ValidationCode
Outputs	
Error Cases	
Activity Diagram	T-Main::Create extended logging for an application

8.2.2 Optional: Create extended logging for an entitlement



Use Case	<u>Create extended logging for an entitlement</u>
Description	<p>For monitoring purposes, logging related to invalid entitlements should be created by a terminal and analysed by the back-office system.</p> <p>The extended logging of entitlements is triggered if a blocked entitlement was presented (only chip-based), the validity period of the entitlement has been expired or the tariff conditions of the entitlement were invalid. The detailed reason can be found in ValidationEnum.</p> <p>If the AppInstanceId is available, this value must also be provided.</p>
Initiating Actor	
Reacting Actor	Service Operator Terminal Main Module Customer Contract Partner Terminal Main Module
Preconditions	User medium is selected/read
Postconditions	
Linked Use Cases (Extended By)	



Linked Use Cases (Includes)	Process extended logging for an entitlement
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement Id : EntitlementId Validation code : ValidationCode
Outputs	
Error Cases	
Activity Diagram	T-Main::Create extended logging for an entitlement



9 Basic Bundle CCP-Terminal - Foundation

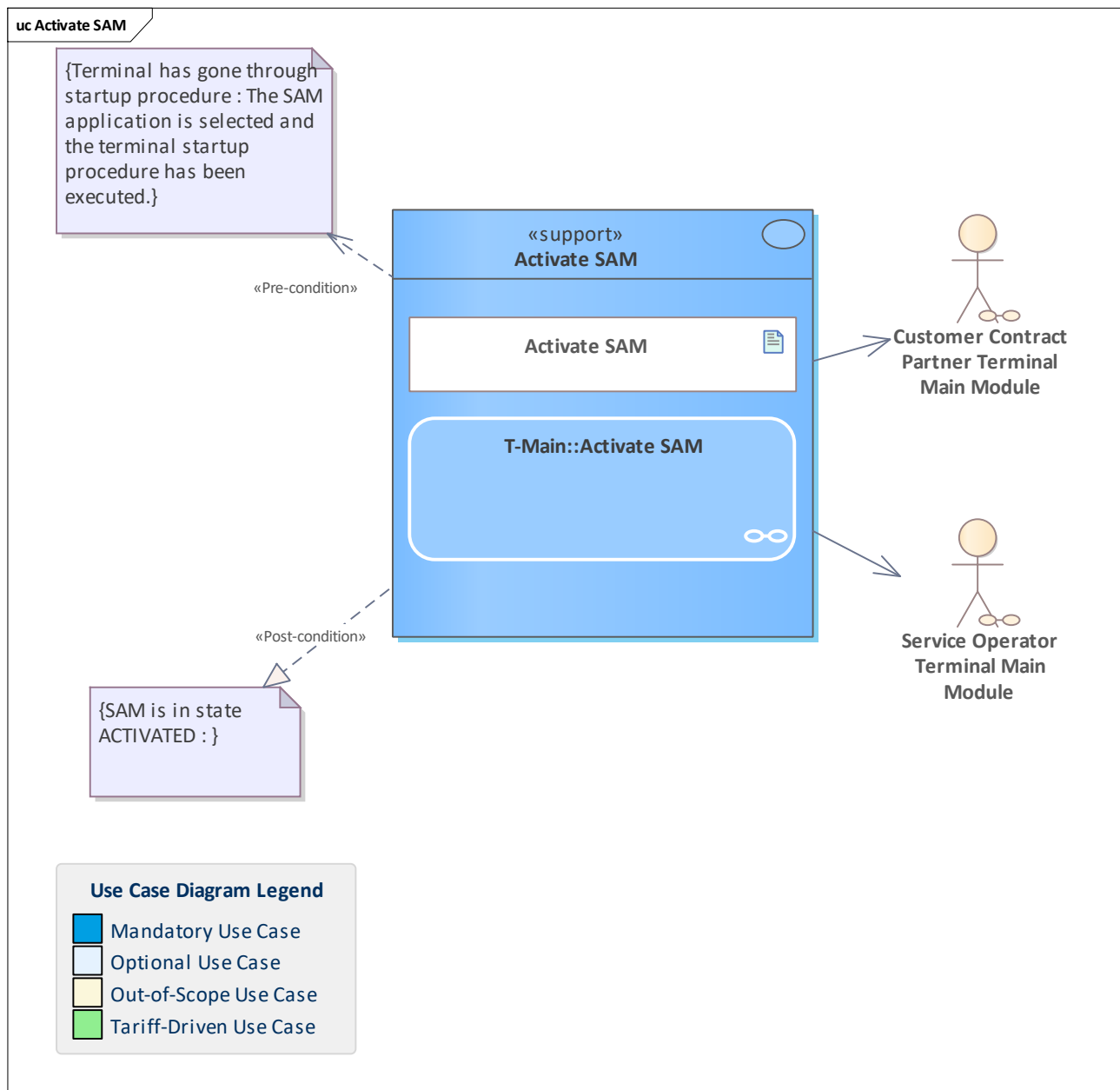
Functionality bundle that covers CCP terminal use cases forming the basis for handling sales, payment methods and CICO.

9.1 Overview

[Activate SAM](#)

9.2 Use Cases

9.2.1 Activate SAM



Use Case	<u>Activate SAM</u>
Description	The terminal activates the SAM to enable the use of the full feature set of the SAM.
Initiating Actor	
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u> <u>Service Operator Terminal Main Module</u>
Preconditions	<u>Terminal has gone through startup procedure</u> <u>SAM is in state ACTIVATED</u>
Postconditions	<u>SAM is in state ACTIVATED</u>
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	<u>SAM is in state ACTIVATED</u>



Base Activity	
Inputs	
Outputs	
Error Cases	Activation failed
Activity Diagram	T-Main::Activate SAM

10 Basic Bundle CCP-Terminal - UM with Application

Functionality bundle that covers use cases forming the basis for handling sales, payment methods, etc. when using user media with an application (e.g. chip cards).

10.1 Overview

Check user medium with application as CCP

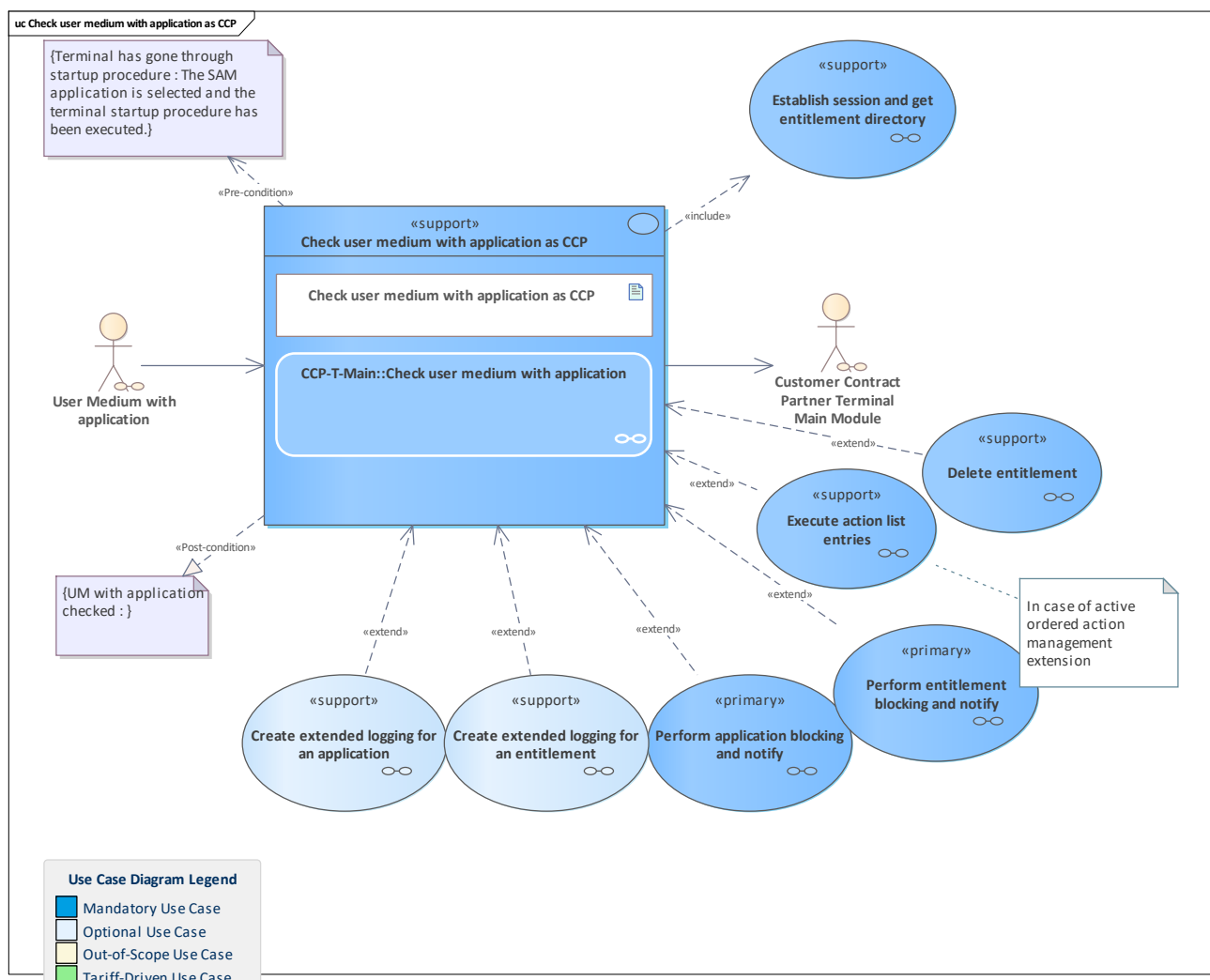
Delete entitlement

Display application data

Display entitlement

10.2 Use Cases

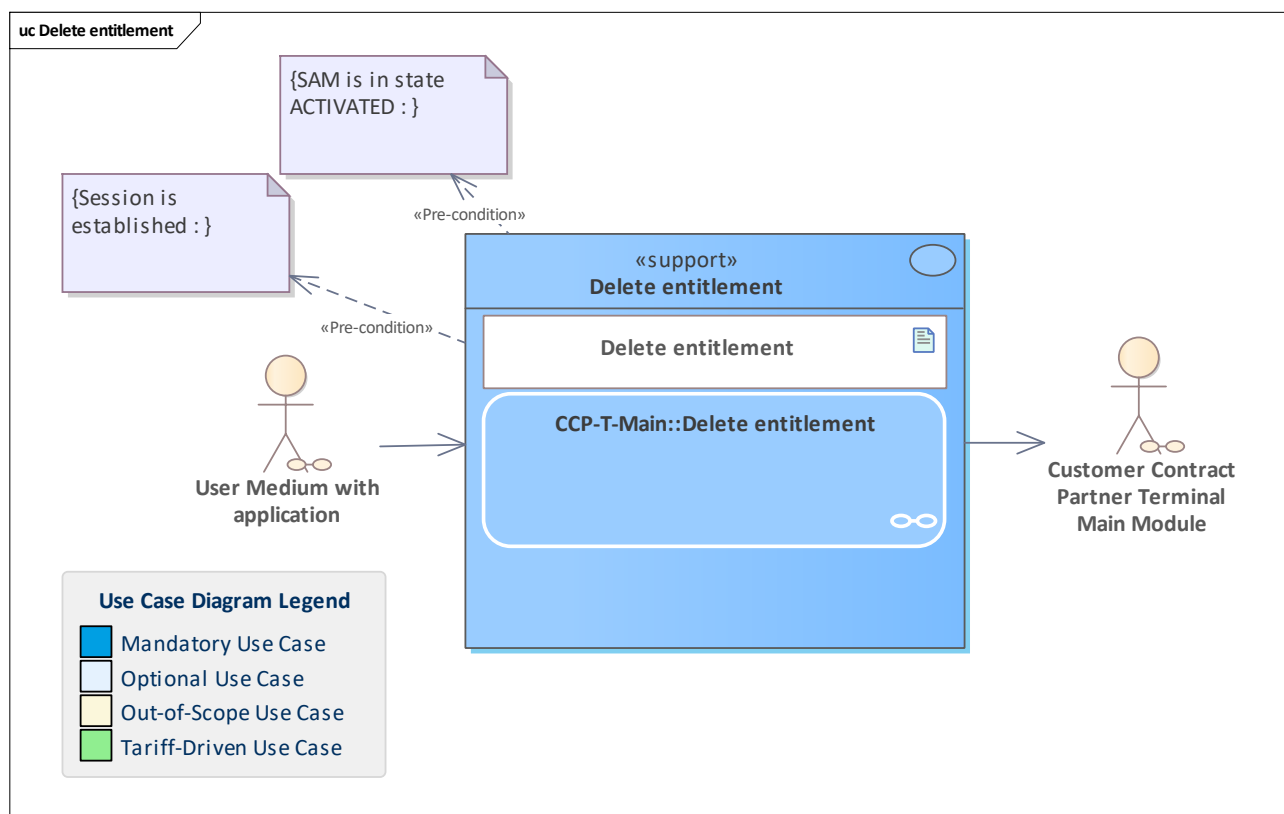
10.2.1 Check user medium with application as CCP





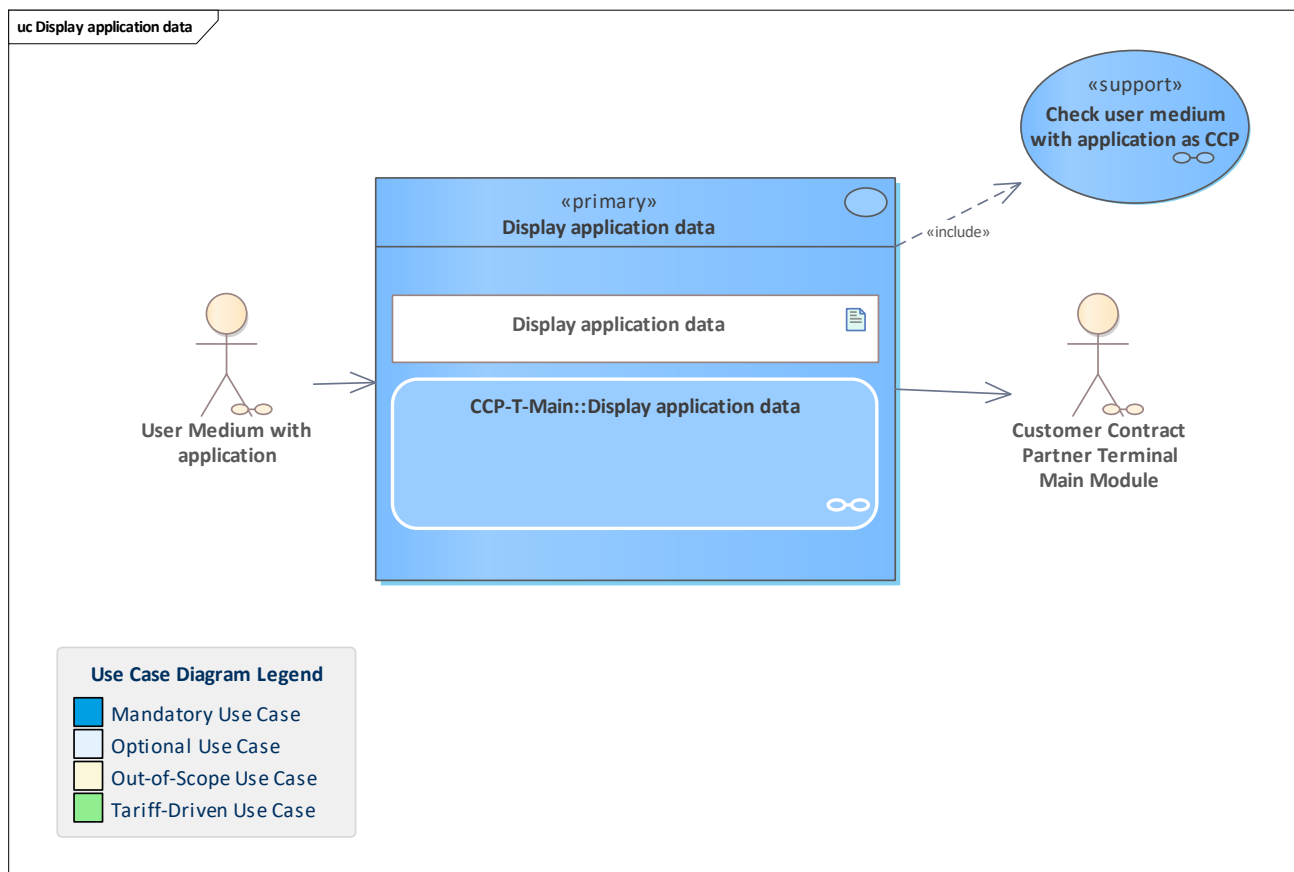
Use Case	Check user medium with application as CCP
Description	<p>In the following, the processes are summarised for all use cases with the user medium with an application, which are generally to be realised only once at the beginning of a process.</p> <p>This use case has the following steps:</p> <ul style="list-style-type: none">• Establishing a secure messaging session• Reading the application directory and entitlement directory• Checking the application directory (hotlist, status, temporal validity)• Checking the entitlement directory (hotlist, status, temporal validity)• Potentially executing pending action orders
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Terminal has gone through startup procedure
Postconditions	UM with application checked
Linked Use Cases (Extended By)	Create extended logging for an application / Execute action list entries / Perform application blocking and notify / Create extended logging for an entitlement / Perform entitlement blocking and notify / Delete entitlement
Linked Use Cases (Includes)	Establish session and get entitlement directory
Linked Use Cases (Realises)	UM with application checked
Base Activity	
Inputs	
Outputs	List of valid entitlement directory entries : EntitlementDirectoryEntry
Error Cases	Application blocked, but within validity period Invalid application
Activity Diagram	CCP-T-Main::Check user medium with application

10.2.2 Delete entitlement



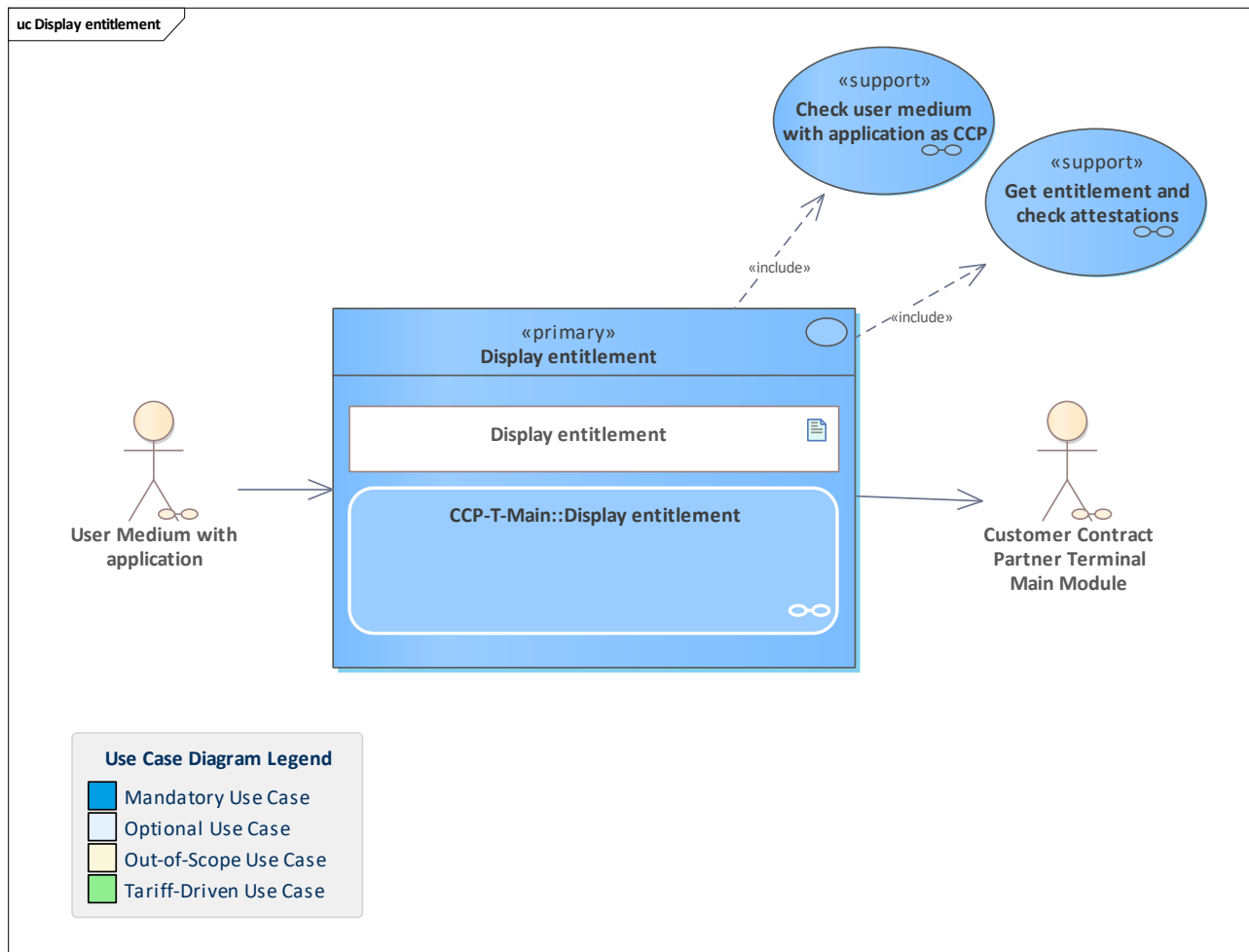
Use Case	Delete entitlement
Description	The terminal deletes an entitlement from the user medium application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entry ID : EntryId
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Delete entitlement

10.2.3 Display application data



Use Case	Display application data
Description	The application data that is relevant for the customer is displayed using the application directory.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Display application data

10.2.4 Display entitlement



Use Case	Display entitlement
Description	Display an entitlement on a user medium with an application, independent of its status and validity period.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	



Activity Diagram	CCP-T-Main::Display_entitlement
-------------------------	---



11 Basic Bundle CCP-Terminal - UM in Customer Center

This functionality bundle includes use cases that are normally only carried out in the customer centre.

11.1 Overview

Unblock application

Perform application unblocking and notify

Unblock entitlement

Perform entitlement unblocking and notify

Optional: Take back application

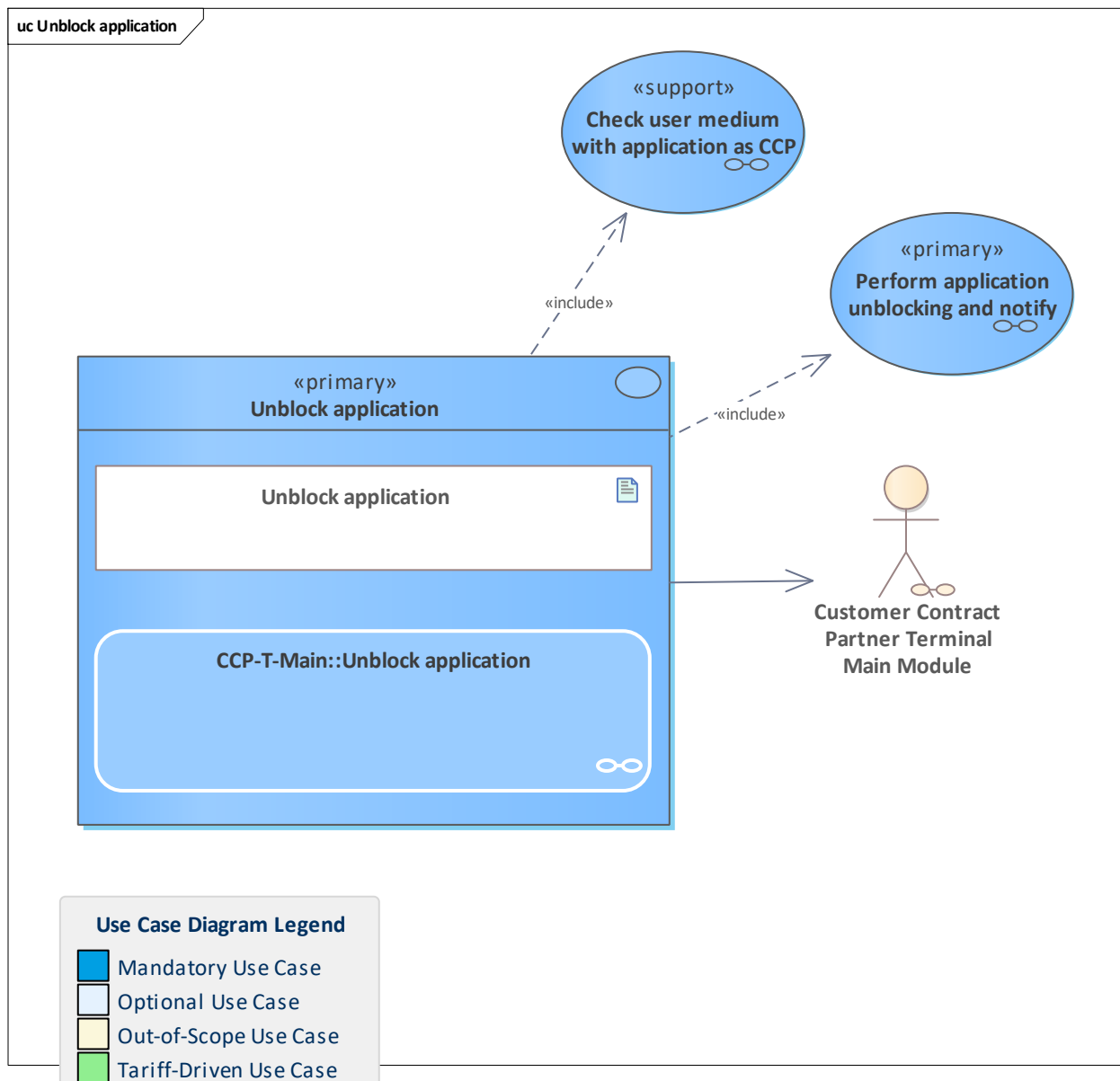
Optional: Perform application termination and notify

Optional: Configure user medium application

Optional: Exchange user medium with application

11.2 Use Cases

11.2.1 Unblock application

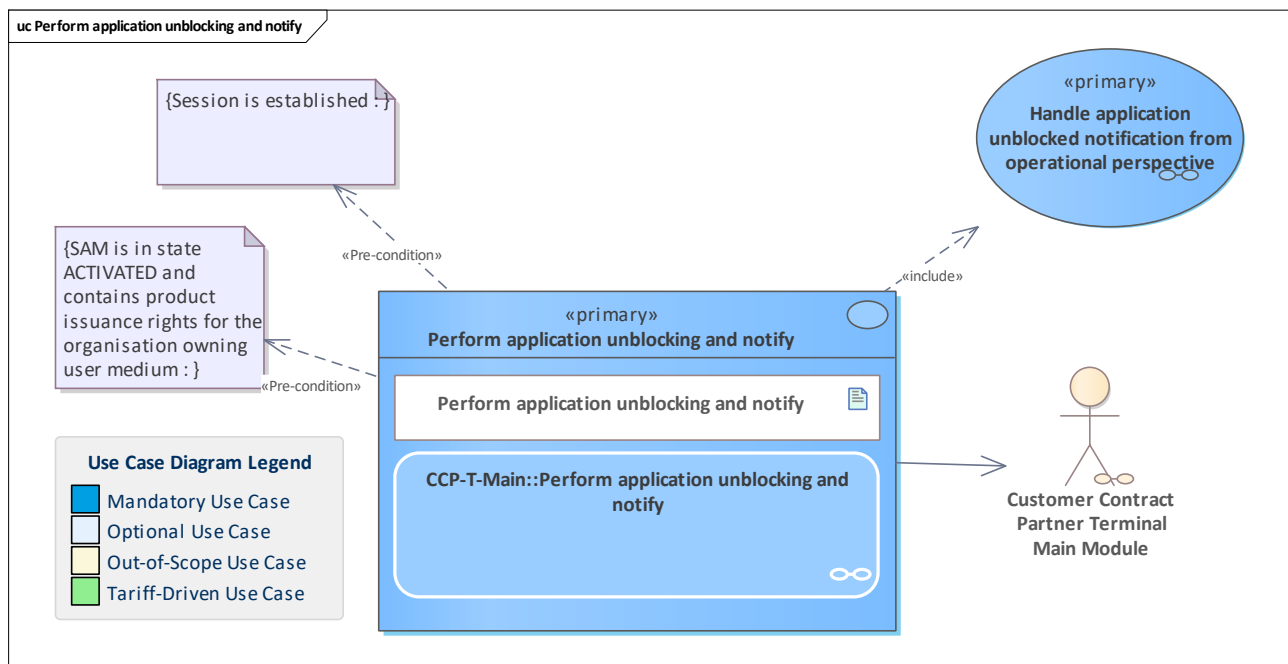


Use Case	Unblock application
Description	A user medium with an application will be checked in the terminal. In case of a blocked application within the validity period, the terminal checks if the application can be unblocked. If yes, the terminal performs application unblocking and notifies the back-office system. Otherwise, the unblocking process of the application is stopped. Note: only the pCCP is allowed to unblock its owned applications.
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform application unblocking and notify / Check user medium with application as CCP
Linked Use Cases	



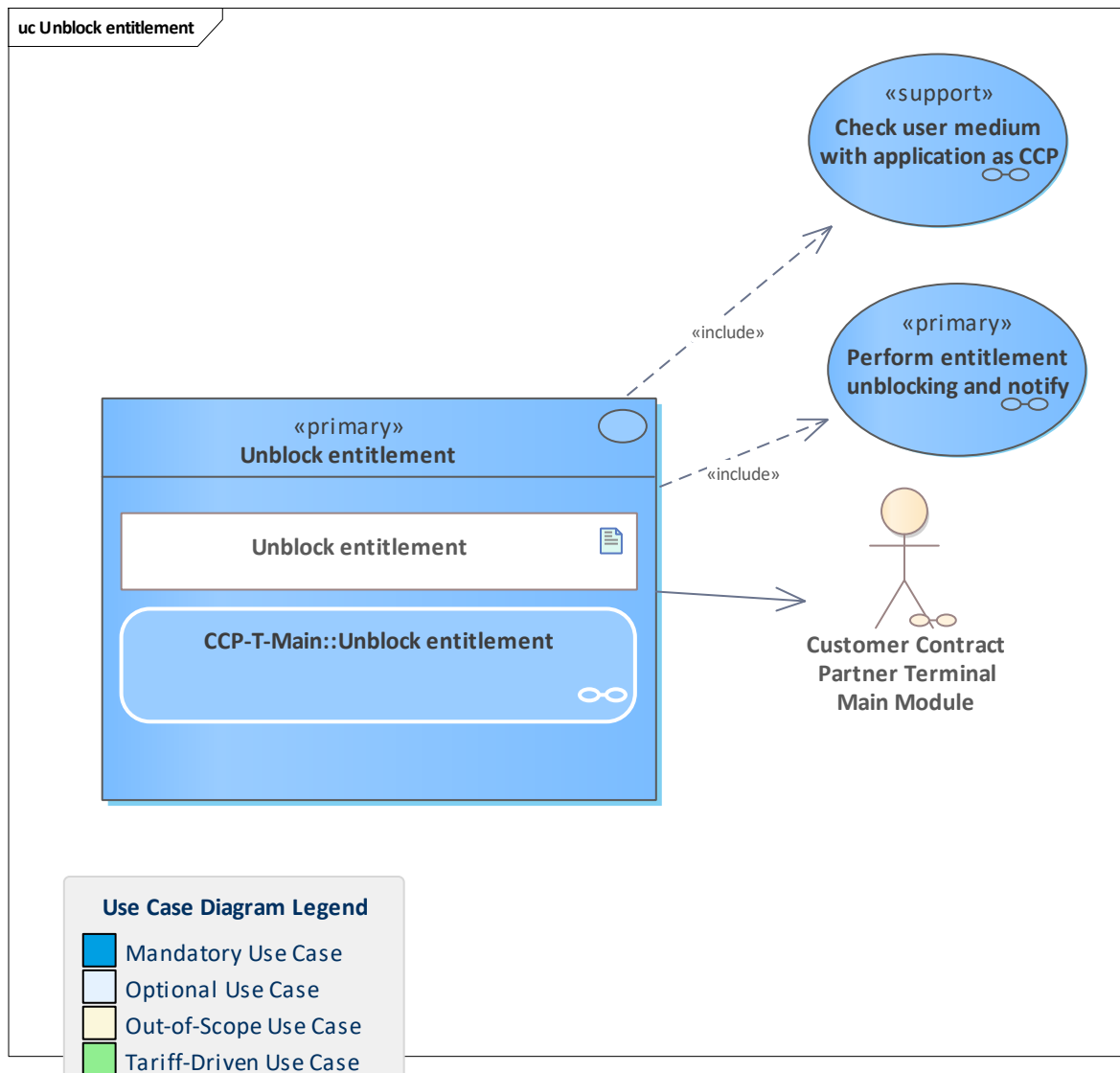
(Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Unblock application

11.2.2 Perform application unblocking and notify



Use Case	Perform application unblocking and notify
Description	The pCCP decides to unblock the application. The terminal runs the transaction to unblock the user medium application and informs the back-office system about the result.
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Session is established SAM is in state ACTIVATED and contains product issuance rights for the organisation owning user medium
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle application unblocked notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform application unblocking and notify

11.2.3 Unblock entitlement

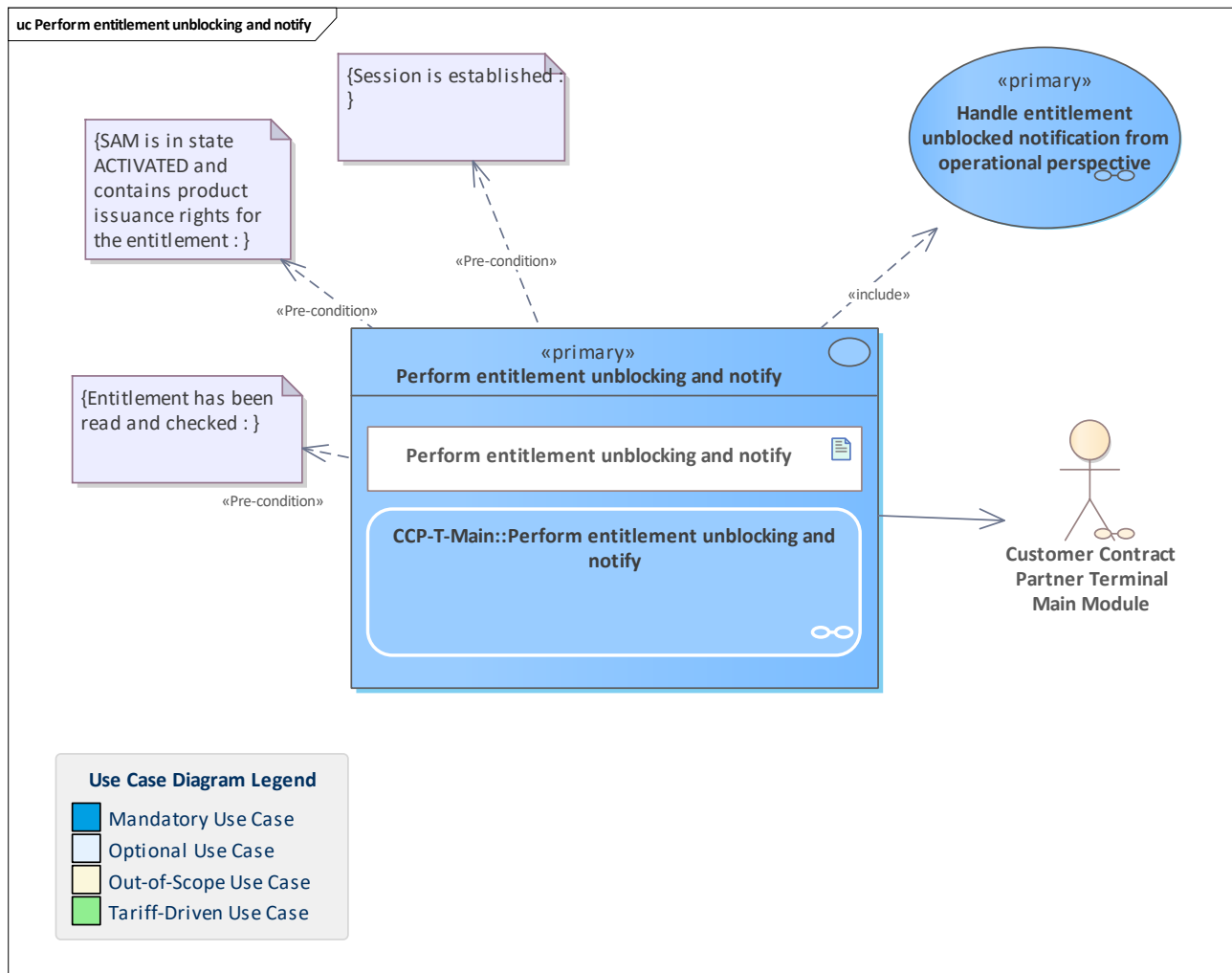


Use Case	<u>Unblock entitlement</u>
Description	<p>A user medium with an application will be checked in the terminal. All previously blocked entitlements are determined and read. Each entitlement in question is checked if it can be unblocked. Otherwise, the unblocking of the entitlement in question is skipped.</p> <p>If at least one entitlement exists for unblocking, the terminal performs entitlement unblocking and notifies the CCP back-office system.</p> <p>Note: only the pCCP is allowed to unblock its owned entitlements.</p>
Initiating Actor	
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases	<u>Check user medium with application as CCP</u> / <u>Perform entitlement</u>



(Includes)	unblocking and notify
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Unblock entitlement

11.2.4 Perform entitlement unblocking and notify

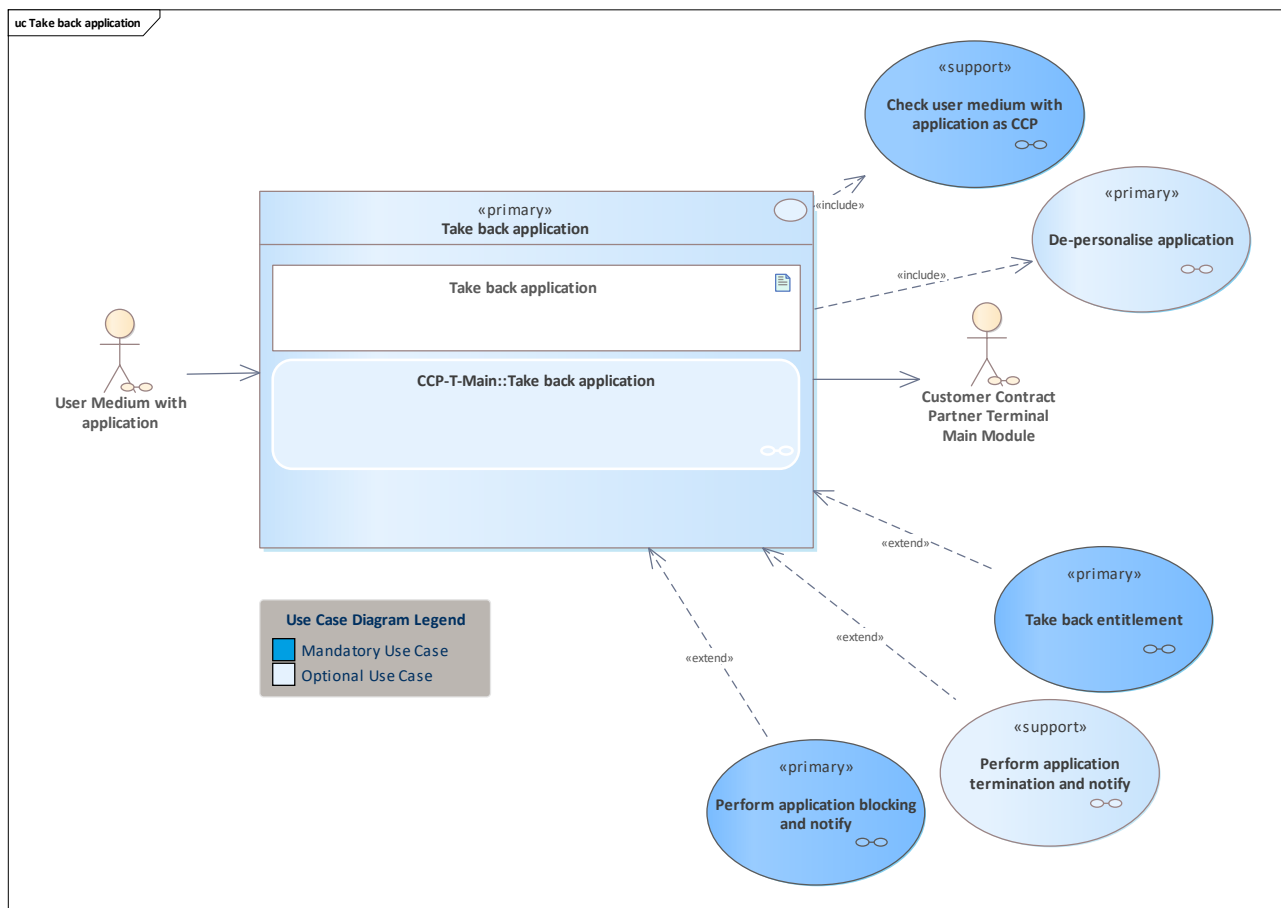


Use Case	Perform entitlement unblocking and notify
Description	The terminal runs the transaction to unblock the entitlement in question and informs the back-office system about the result. If the transaction is aborted, the back-office system is also notified for consistent monitoring data. Note: only the pCCP is allowed to unblock its owned entitlements.
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked Session is established SAM is in state ACTIVATED and contains product issuance rights for the entitlement Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle entitlement unblocked notification from operational perspective
Linked Use Cases	



(Realises)	
Base Activity	
Inputs	Entitlement directory entry : EntitlementDirectoryEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform entitlement unblocking and notify

11.2.5 Optional: Take back application

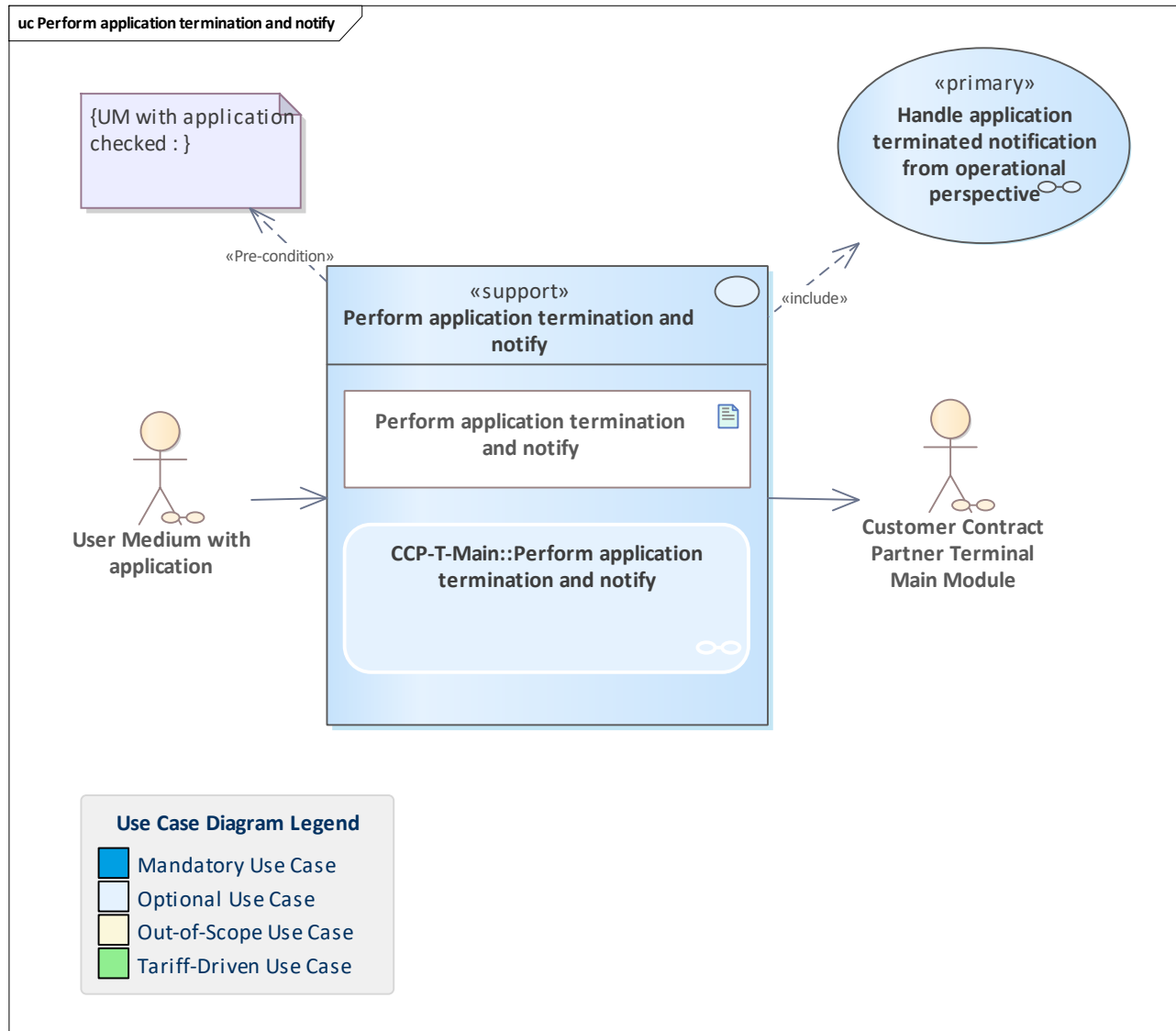


Use Case	Take back application
Description	<p>Use case for a CCP terminal to take back an application. This process may only be performed by the organisation that issued the application.</p> <p>An application (of a user medium) is terminated when the underlying user medium is taken back from a customer (and has reached its end of life) or when the (((etiCORE-specific application is removed from a customer-owned user medium. This includes removing all customer-specific information from an (((etiCORE application.</p> <p>The actual process of terminating an application leads to marking the application as terminated.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Perform application termination and notify / Take back entitlement / Perform application blocking and notify / Perform application blocking and notify
Linked Use Cases (Includes)	Check user medium with application as CCP / Check user medium with application as CCP / De-personalise application
Linked Use Cases (Realises)	



Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Take back application

11.2.6 Optional: Perform application termination and notify

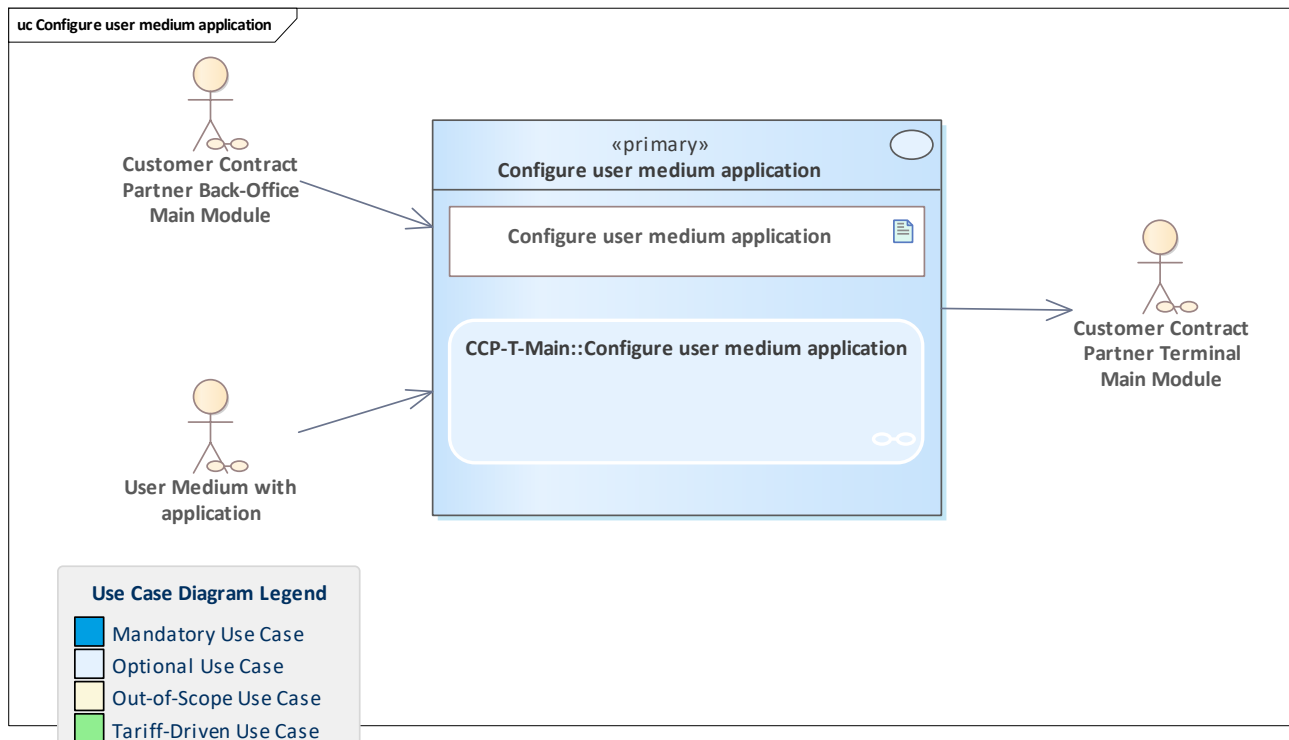


Use Case	<u>Perform application termination and notify</u>
Description	<p>Perform the transaction to terminate a user medium application and notify interested parties.</p> <p>Note: the application termination action is automatically committed by the user medium application after execution. Thus, no timeout scenario has to be handled. Instead, for application terminations an answer by the user medium that can not be validated by the SAM may indicate something similar to a timeout scenario: the termination may be permanent, but no attestation about it can be provided.</p>
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>UM with application checked</u>
Postconditions	



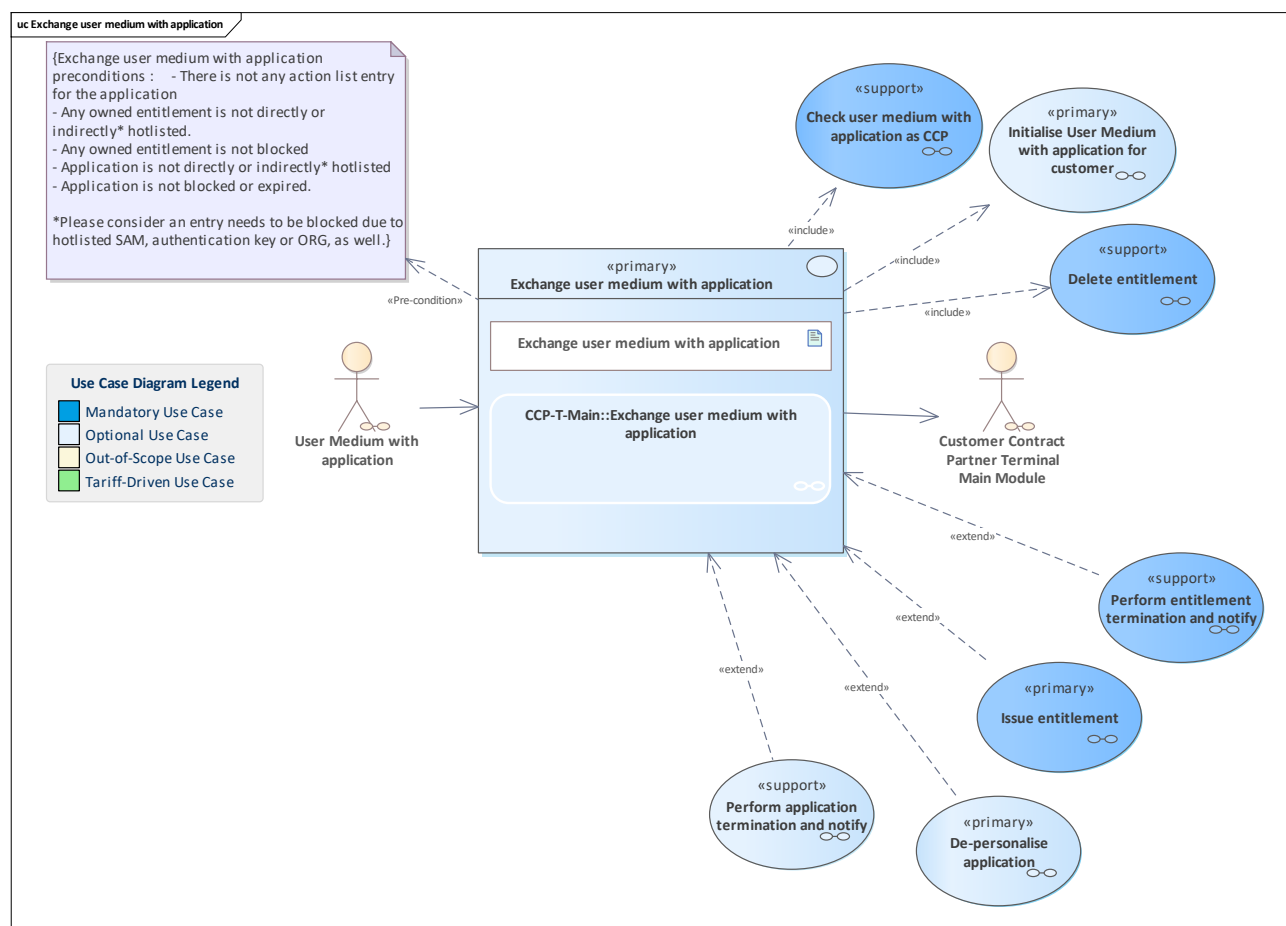
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle application terminated notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform application termination and notify

11.2.7 Optional: Configure user medium application



Use Case	Configure user medium application
Description	A user medium application is (re-)configured by a terminal. For this use case, the terminal needs to interact with the MMS, which is only possible after organisational and contractual preparations, see MMS Specification .
Initiating Actor	Customer Contract Partner Back-Office Main Module User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Terminal configure user medium application : tConfigureUserMediumWithApplication
Outputs	Terminal configure user medium application response : tConfigureUserMediumWithApplicationResponse
Error Cases	Terminal configure user medium application exception : tConfigureUserMediumWithApplicationException
Activity Diagram	CCP-T-Main::Configure user medium application

11.2.8 Optional: Exchange user medium with application



Use Case	Exchange user medium with application
Description	<p>A user medium with an application is to be exchanged with the new one. This can be done only by the pCCP.</p> <p>This process considers not only the migration of application data but also entitlements. Please note that only owned entitlements can be migrated.</p> <p>Please note that it is assumed that there is no need for any payment transaction.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Exchange user medium with application preconditions
Postconditions	
Linked Use Cases (Extended By)	Issue entitlement / Perform entitlement termination and notify / De-personalise application / Delete entitlement / Delete entitlement / Perform application termination and notify
Linked Use Cases (Includes)	Check user medium with application as CCP / Check user medium with application as CCP / Initialise User Medium with application for customer / Delete entitlement
Linked Use Cases	



(Realises)	
Base Activity	
Inputs	Password for the new user medium User authentication data configuration
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Exchange user medium with application



12 Basic Bundle CCP-Terminal - UM with Customer Data

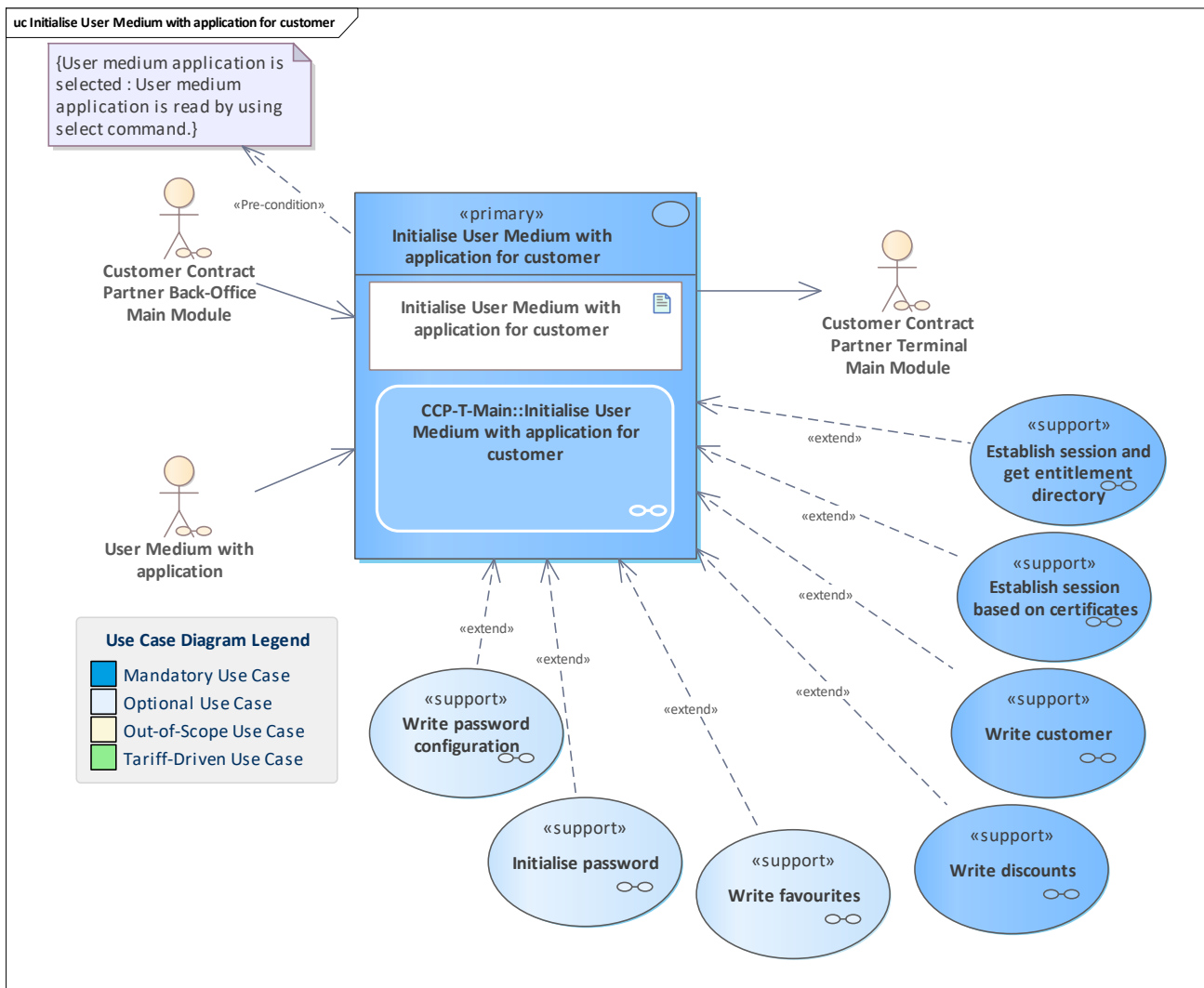
This optional functionality bundle includes use cases that manage extended customer data on the user medium.

12.1 Overview

Initialise User Medium with application for customer
Read information from user medium with application
Write customer
Delete customer
Display customer
Write discounts
Delete discounts
Read customer and discounts
Display discounts
Change customer and discounts
De-personalise application

12.2 Use Cases

12.2.1 Initialise User Medium with application for customer

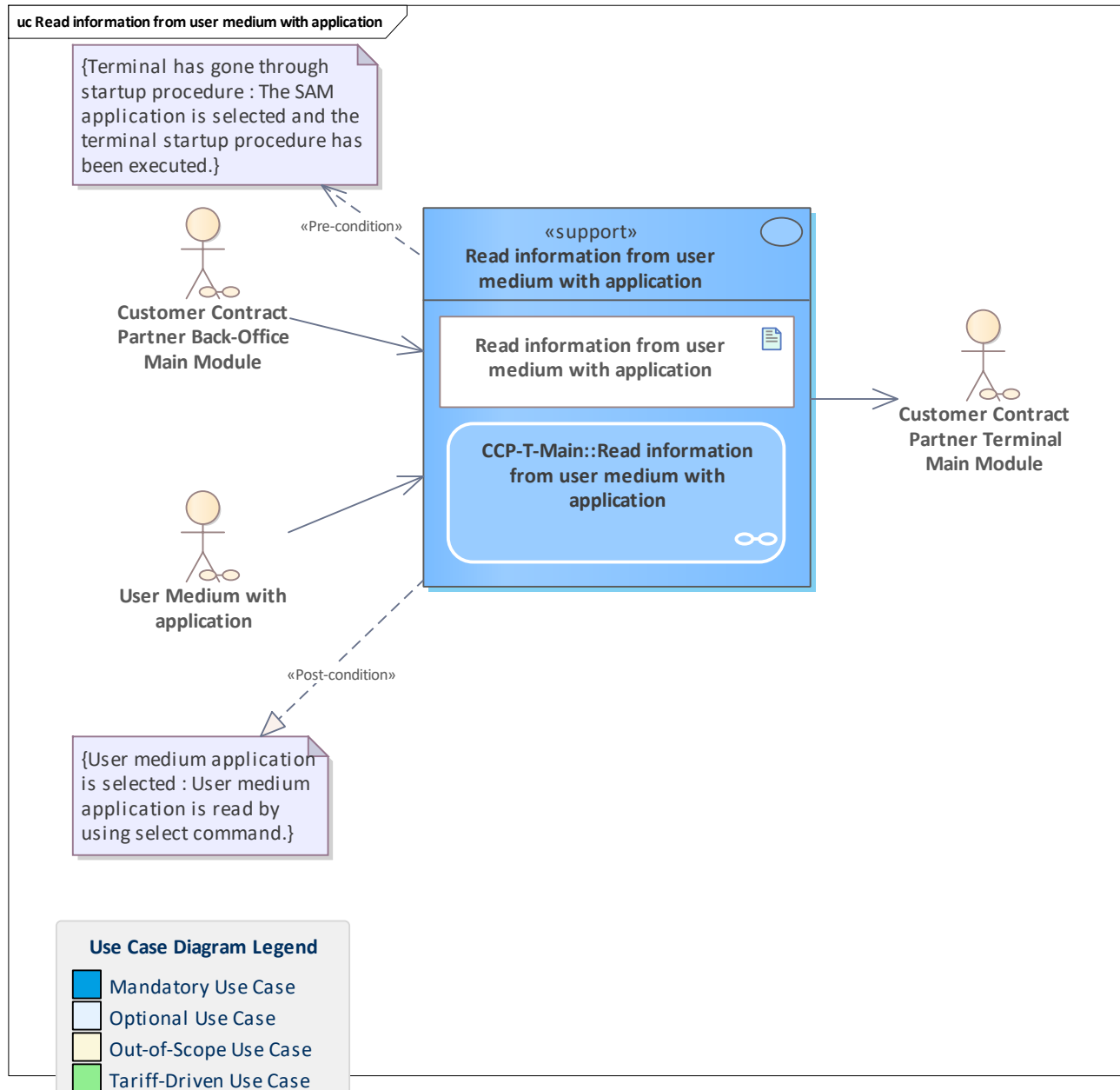


Use Case	Initialise User Medium with application for customer
Description	<p>This process may write customer-related information as needed to a fully configured user medium application:</p> <ul style="list-style-type: none"> customer discounts favourites password configuration the initial password <p>Note: if the password or its configuration shall be set, the session needs to be established based on certificates, otherwise a symmetric key-based session establishment suffices.</p>
Initiating Actor	User Medium with application Customer Contract Partner Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	User medium application is selected User medium application is selected User medium application is selected
Postconditions	
Linked Use Cases	Establish session based on certificates / Write favourites / Write



(Extended By)	discounts / Write customer / Initialise password / Write password configuration / Establish session based on certificates / Establish session and get entitlement directory
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Terminal initialise User Medium with application for customer : tInitialiseUserMediumWithApplicationForCustomer
Outputs	Terminal initialise User Medium with application for customer response : tInitialiseUserMediumWithApplicationForCustomerResponse
Error Cases	Terminal initialise User Medium with application for customer exception : tInitialiseUserMediumWithApplicationForCustomerException
Activity Diagram	CCP-T-Main::Initialise User Medium with application for customer

12.2.2 Read information from user medium with application

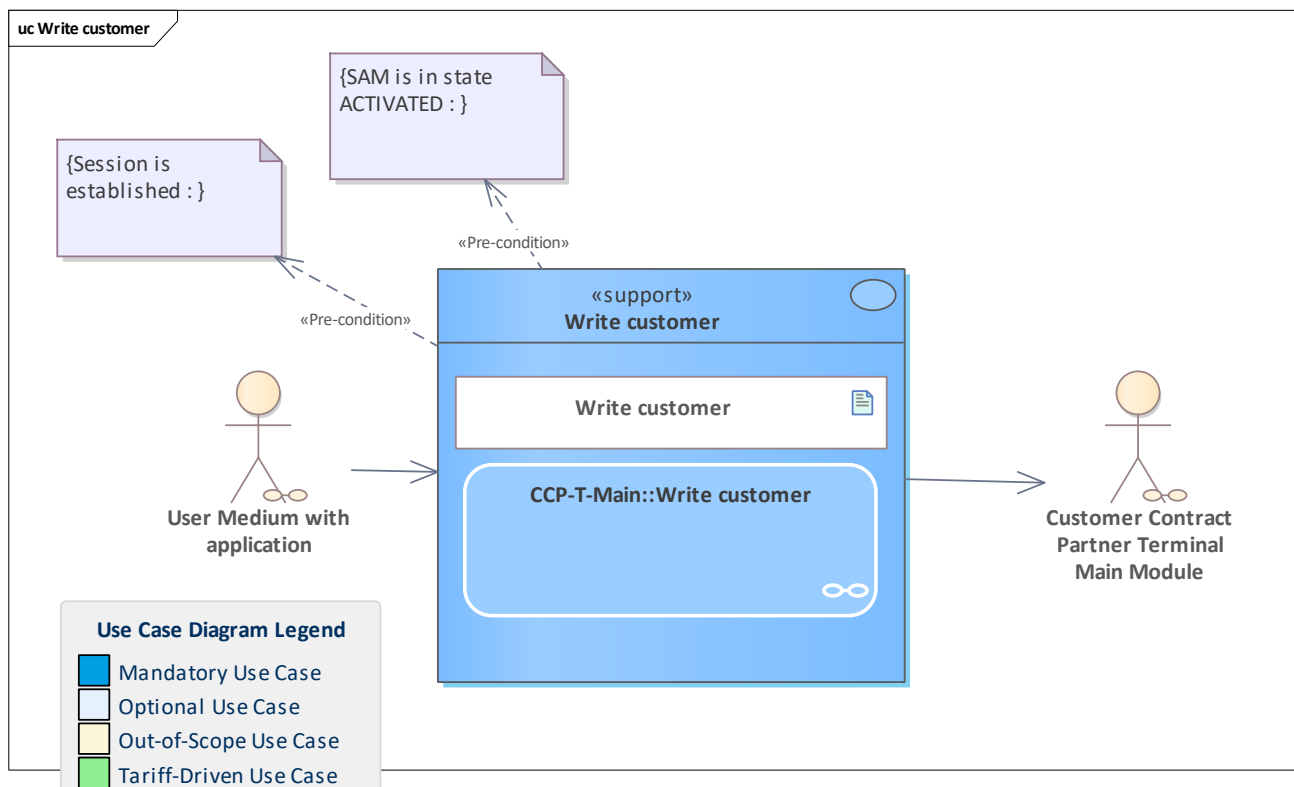


Use Case	Read information from user medium with application
Description	Read the application instance ID and validity period of a user medium and pass it back to the requesting CCP back-office system. Used in the context of user medium personalisation.
Initiating Actor	User Medium with application Customer Contract Partner Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Terminal has gone through startup procedure
Postconditions	User medium application is selected
Linked Use Cases (Extended By)	



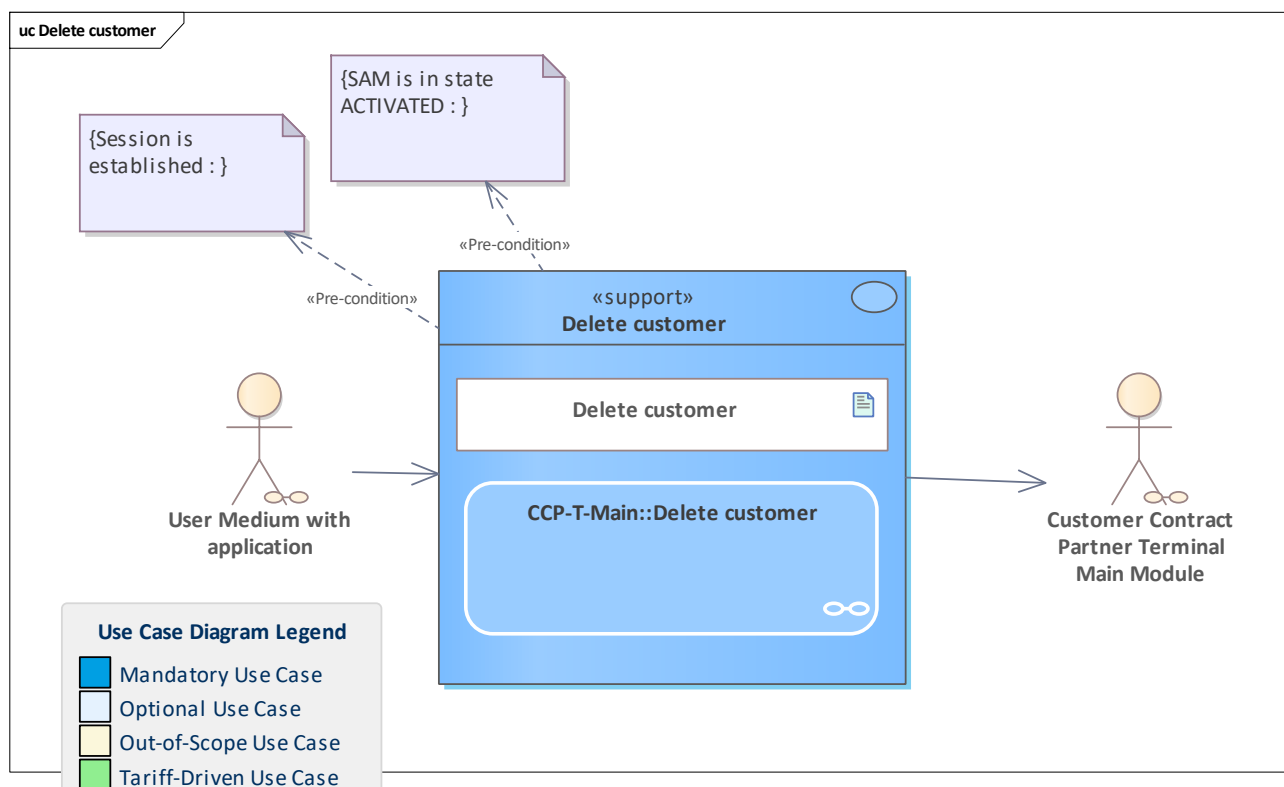
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	User medium application is selected
Base Activity	
Inputs	Terminal read user medium with application information : tReadUserMediumWithApplicationInformation
Outputs	Terminal read user medium with application information response : tReadUserMediumWithApplicationInformationResponse
Error Cases	Terminal read user medium with application information exception : tReadUserMediumWithApplicationInformationException
Activity Diagram	CCP-T-Main::Read information from user medium with application

12.2.3 Write customer



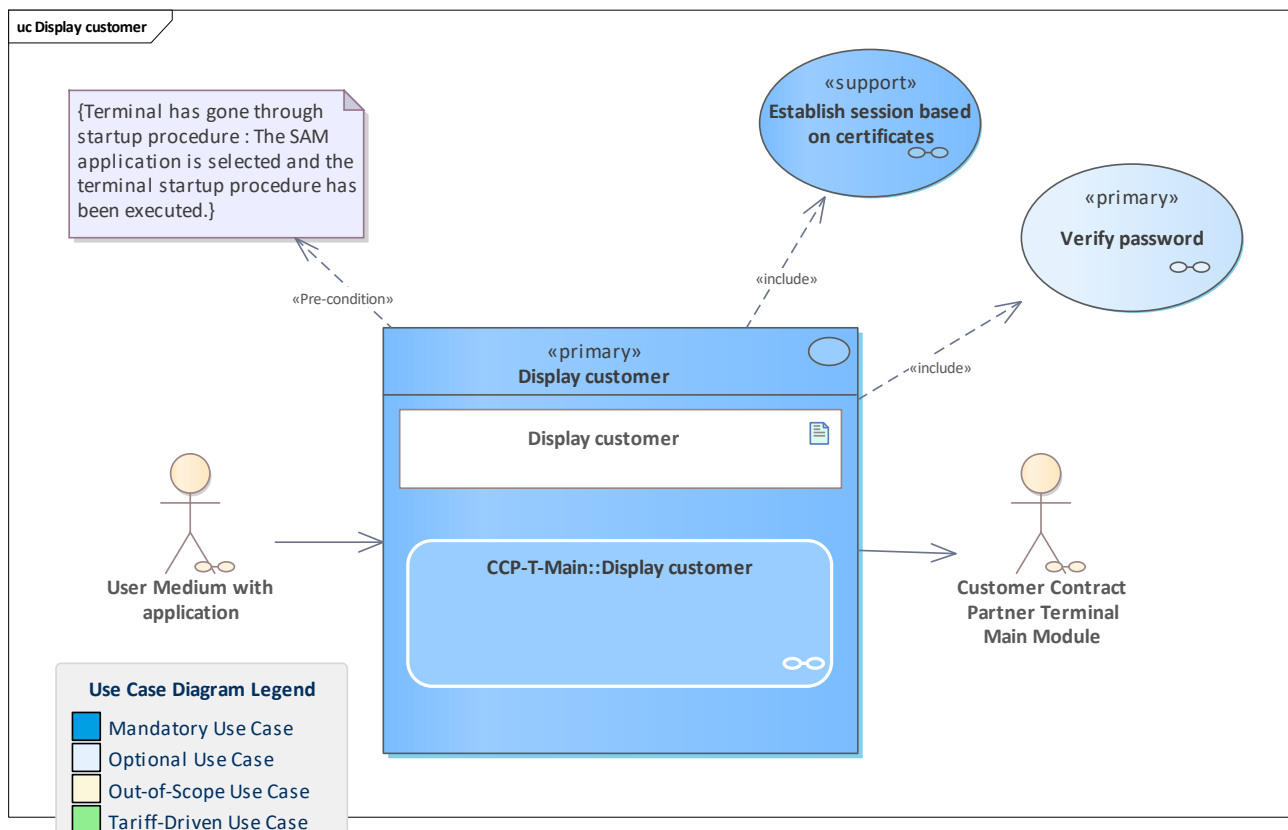
Use Case	Write customer
Description	Write the customer data object to the user medium with an application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Customer : Customer
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Write customer

12.2.4 Delete customer



Use Case	Delete customer
Description	Delete the customer data object on the user medium with an application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Delete customer

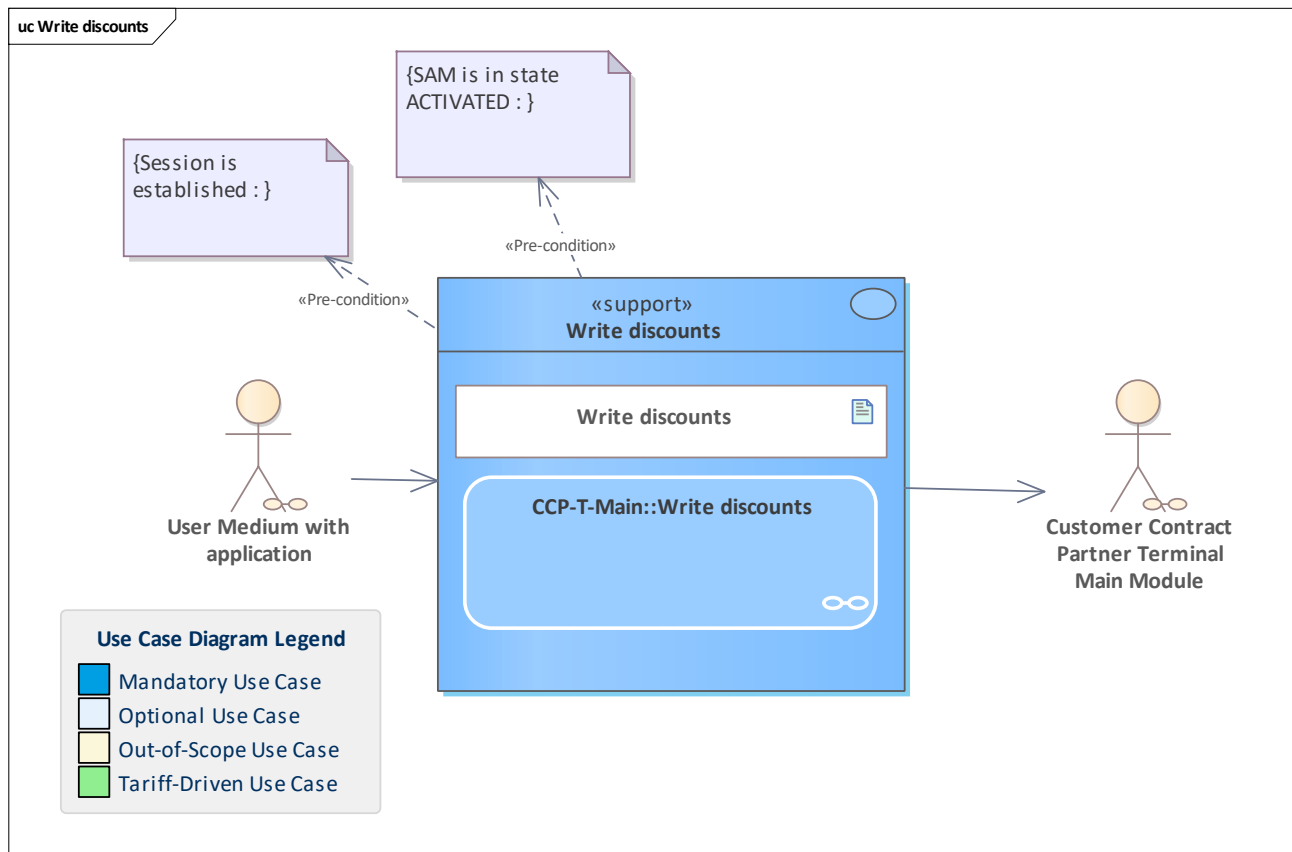
12.2.5 Display customer



Use Case	Display customer
Description	The customer wants to display his user medium-located data on a CCP-T. The customer must enter his password/PIN to show the customer data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Terminal has gone through startup procedure Terminal has gone through startup procedure
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Verify password / Establish session based on certificates
Linked Use Cases (Realises)	
Base Activity	
Inputs	Password
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Display customer



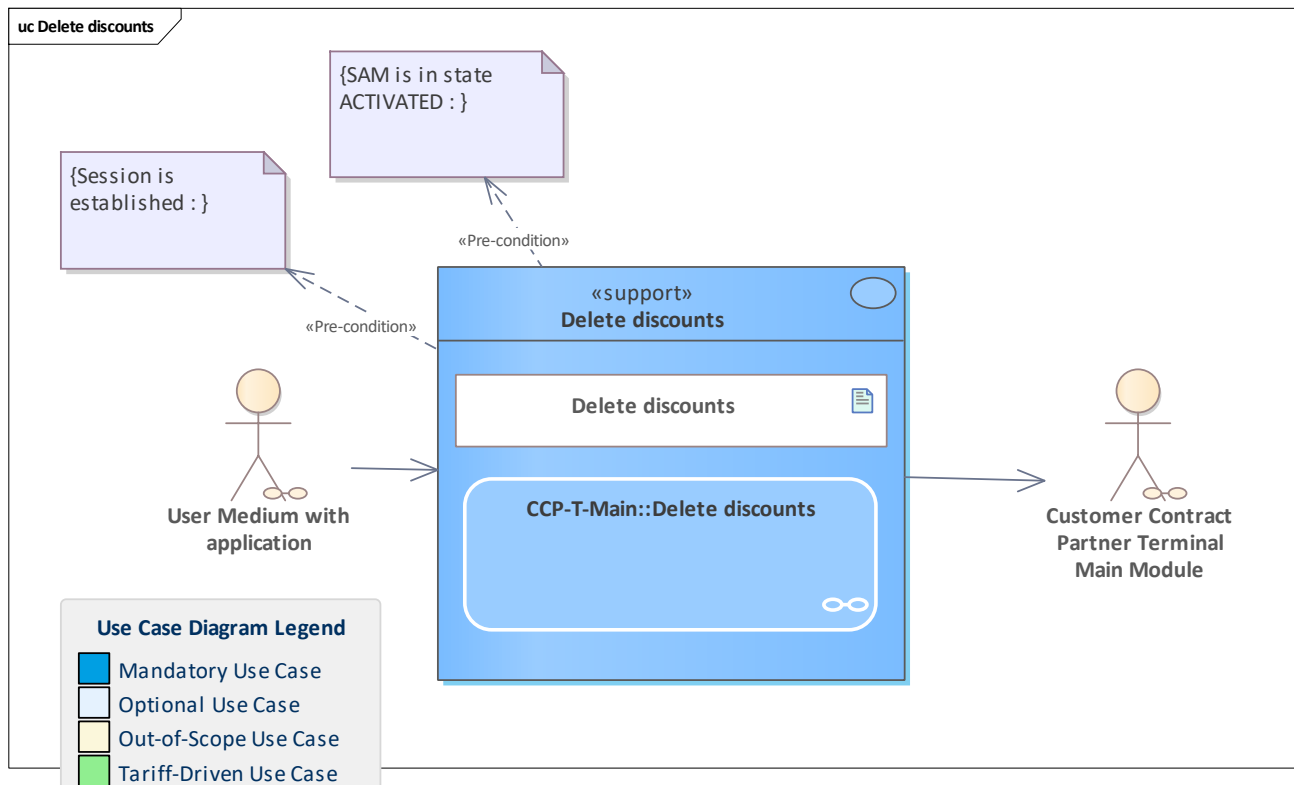
12.2.6 Write discounts



Use Case	Write discounts
Description	Write the discounts data object to the user medium with an application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Discounts : Discounts
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Write discounts

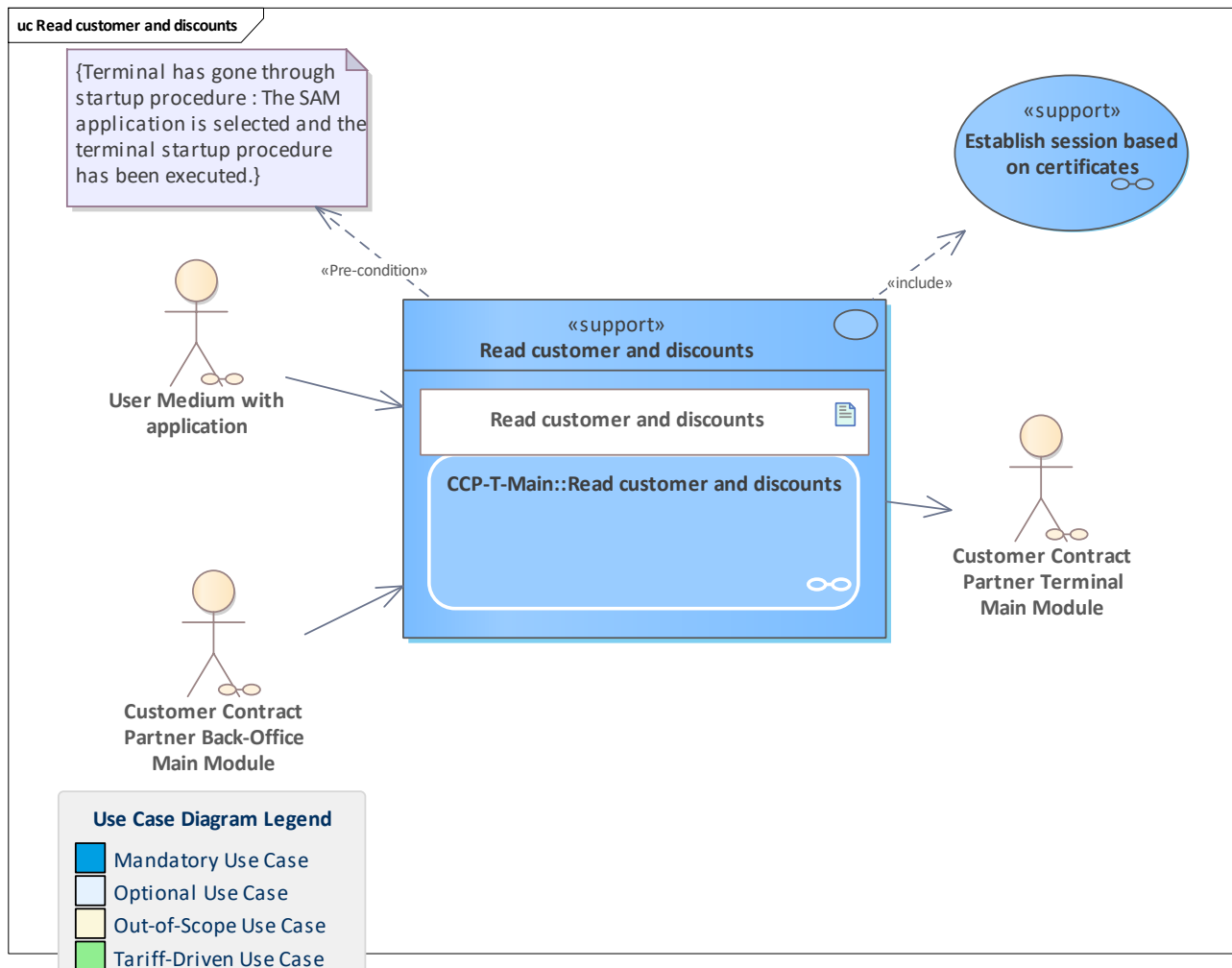


12.2.7 Delete discounts



Use Case	Delete discounts
Description	Delete the discounts data object on the user medium with an application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Delete discounts

12.2.8 Read customer and discounts

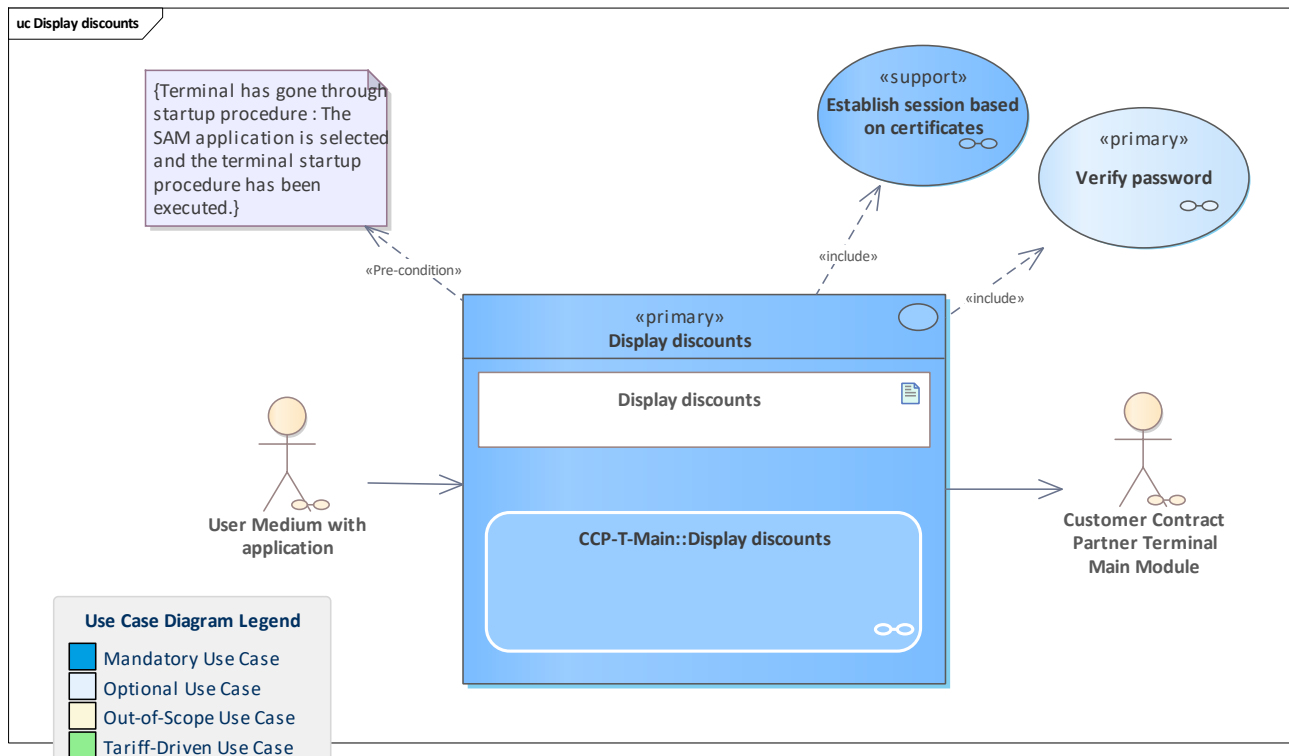


Use Case	Read customer and discounts
Description	The customer and his discounts are read by an authorised terminal to support primary use cases.
Initiating Actor	User Medium with application Customer Contract Partner Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Terminal has gone through startup procedure Terminal has gone through startup procedure
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Establish session based on certificates
Linked Use Cases (Realises)	
Base Activity	
Inputs	tReadCustomerAndDiscounts : tReadCustomerAndDiscounts
Outputs	Terminal read customer and discounts response : tReadCustomerAndDiscountsResponse



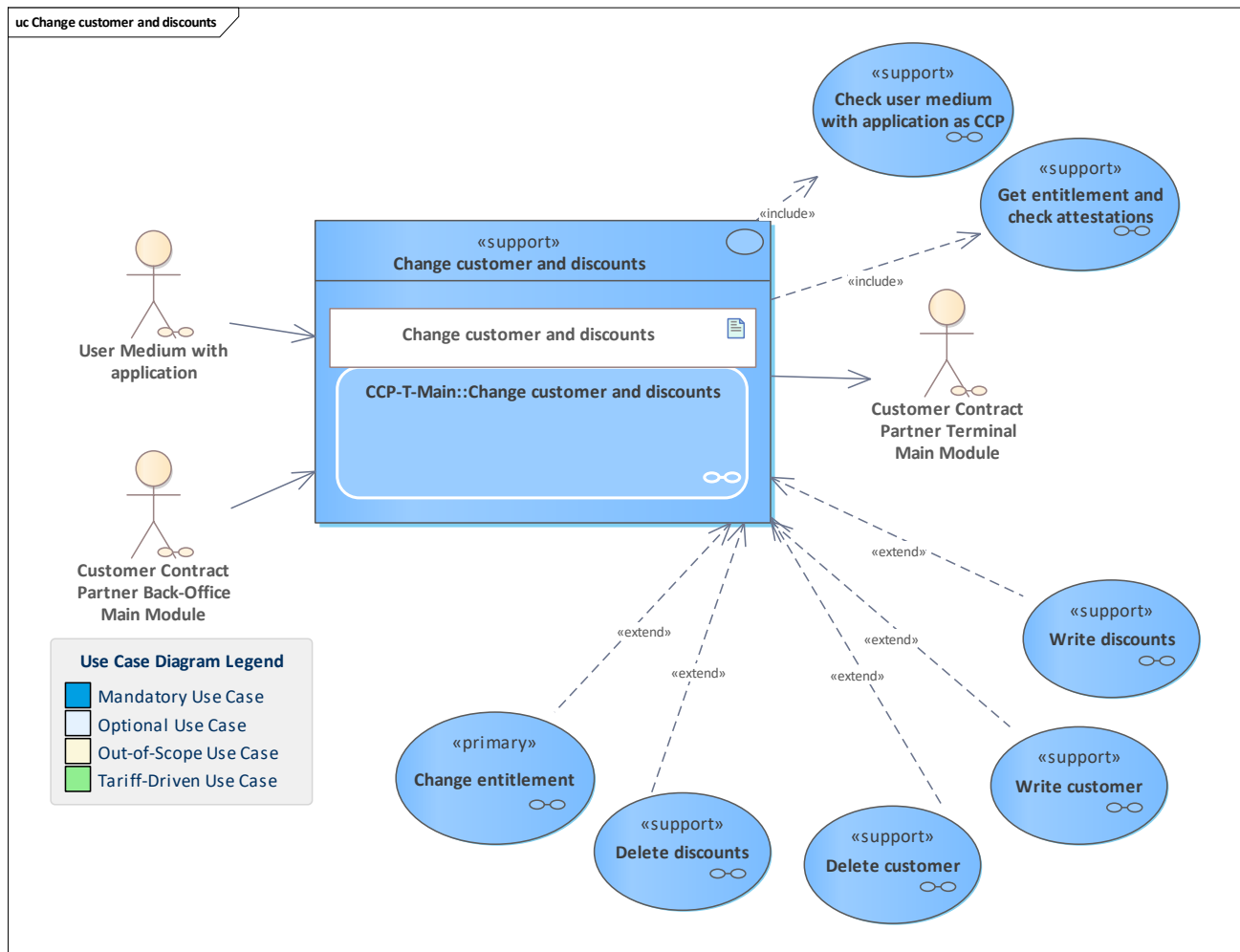
Error Cases	Terminal read customer and discounts exception : tReadCustomerAndDiscountsException
Activity Diagram	CCP-T-Main::Read customer and discounts

12.2.9 Display discounts



Use Case	Display discounts
Description	The customer wants to display his user medium-located discount information on a CCP-T. The customer must enter his password/PIN to show the discount information.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Terminal has gone through startup procedure Terminal has gone through startup procedure
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Verify password / Establish session based on certificates
Linked Use Cases (Realises)	
Base Activity	
Inputs	Password
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Display discounts

12.2.10 Change customer and discounts

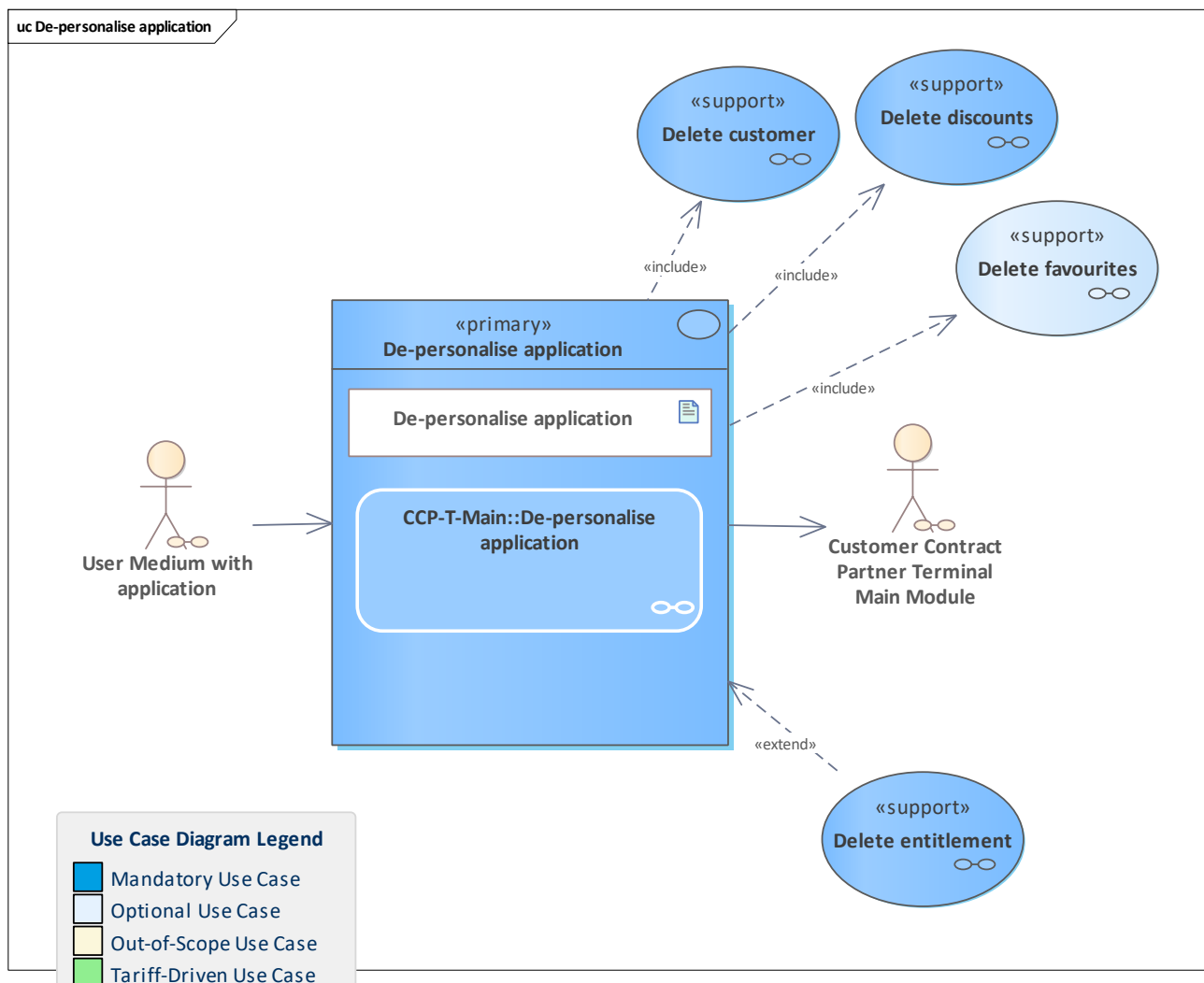


Use Case	Change customer and discounts
Description	Supporting use case of the CCP terminal triggered by the back-office system. Customer and discounts are changed on the user medium with application.
Initiating Actor	User Medium with application Customer Contract Partner Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Change entitlement / Delete customer / Delete discounts / Write discounts / Write customer
Linked Use Cases (Includes)	Check user medium with application as CCP / Check user medium with application as CCP / Get entitlement and check attestations / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	Write customer and discounts : tWriteCustomerAndDiscounts



Outputs	Terminal write customer and discounts response : tWriteCustomerAndDiscountsResponse
Error Cases	Terminal write customer and discounts exception : tWriteCustomerAndDiscountsException
Activity Diagram	CCP-T-Main::Change customer and discounts

12.2.11 De-personalise application



Use Case	De-personalise application
Description	Use case for the CCP terminal to remove all customer-related information from the user medium application. This use case can be utilised to reuse the user medium for another customer or to remove personal data before terminating the user medium for data protection reasons.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Delete entitlement
Linked Use Cases (Includes)	Delete customer / Delete discounts / Delete favourites
Linked Use Cases (Realises)	
Base Activity	
Inputs	



Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::De-personalise application

13 Basic Bundle CCP-Terminal - UM with Password

This optional functionality bundle covers all use cases for the password or PIN handling on a user medium with application. Inside this bundle, the use cases are mandatory.

Note: This functionality bundle depends on [Basic Bundle CCP-Terminal - UM with Customer Data](#).

13.1 Overview

[Initialise password](#)

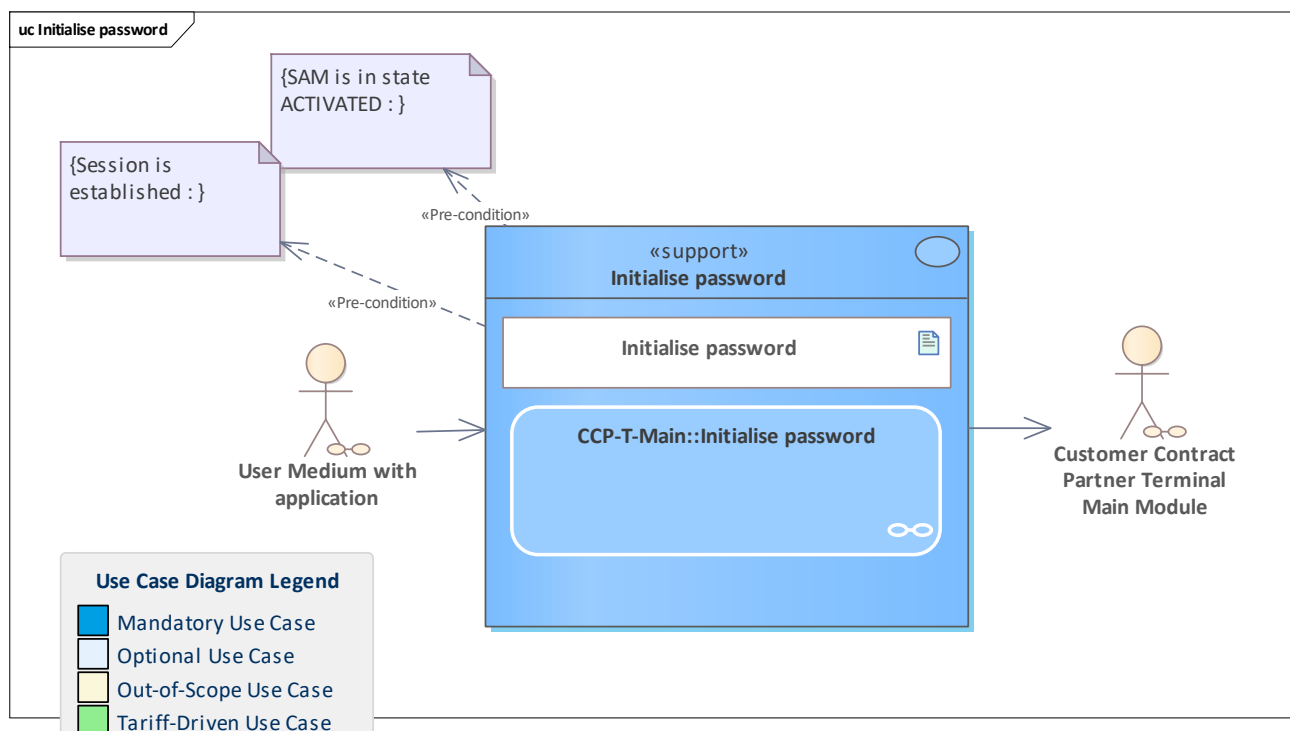
[Write password configuration](#)

[Change password](#)

[Verify password](#)

13.2 Use Cases

13.2.1 Initialise password

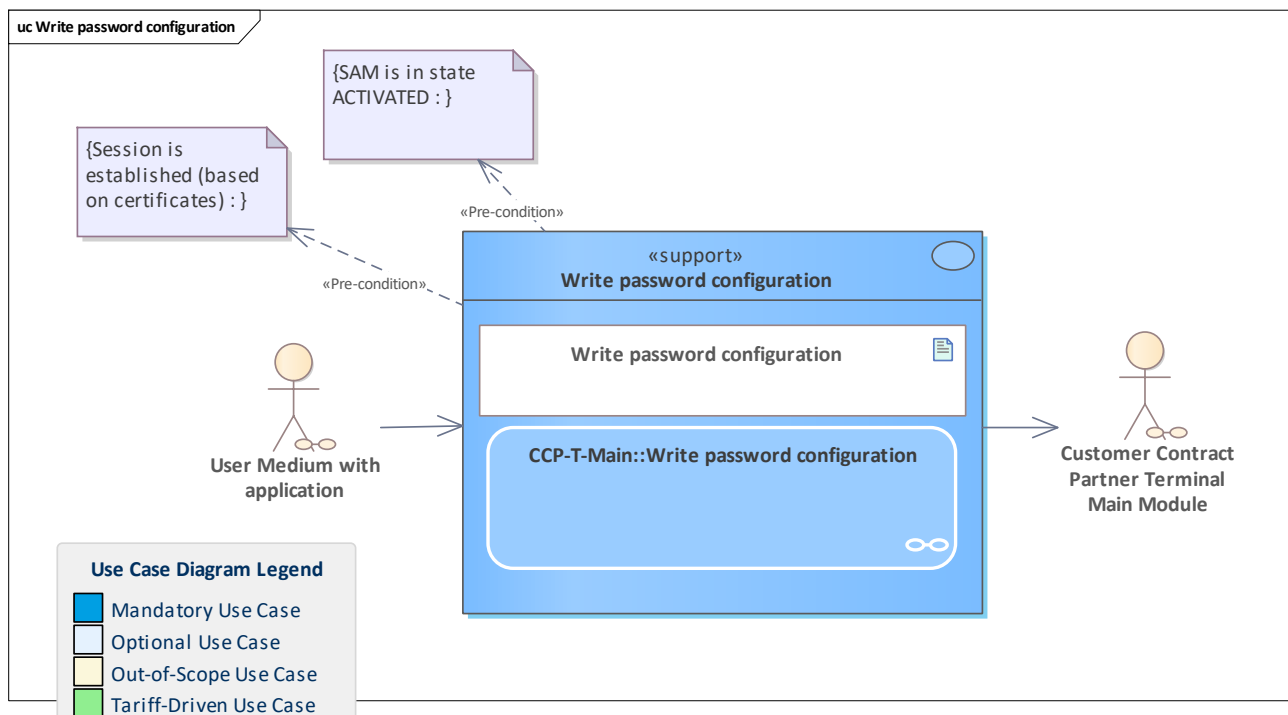


Use Case	Initialise password
Description	The terminal initialises the password for the user medium application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED



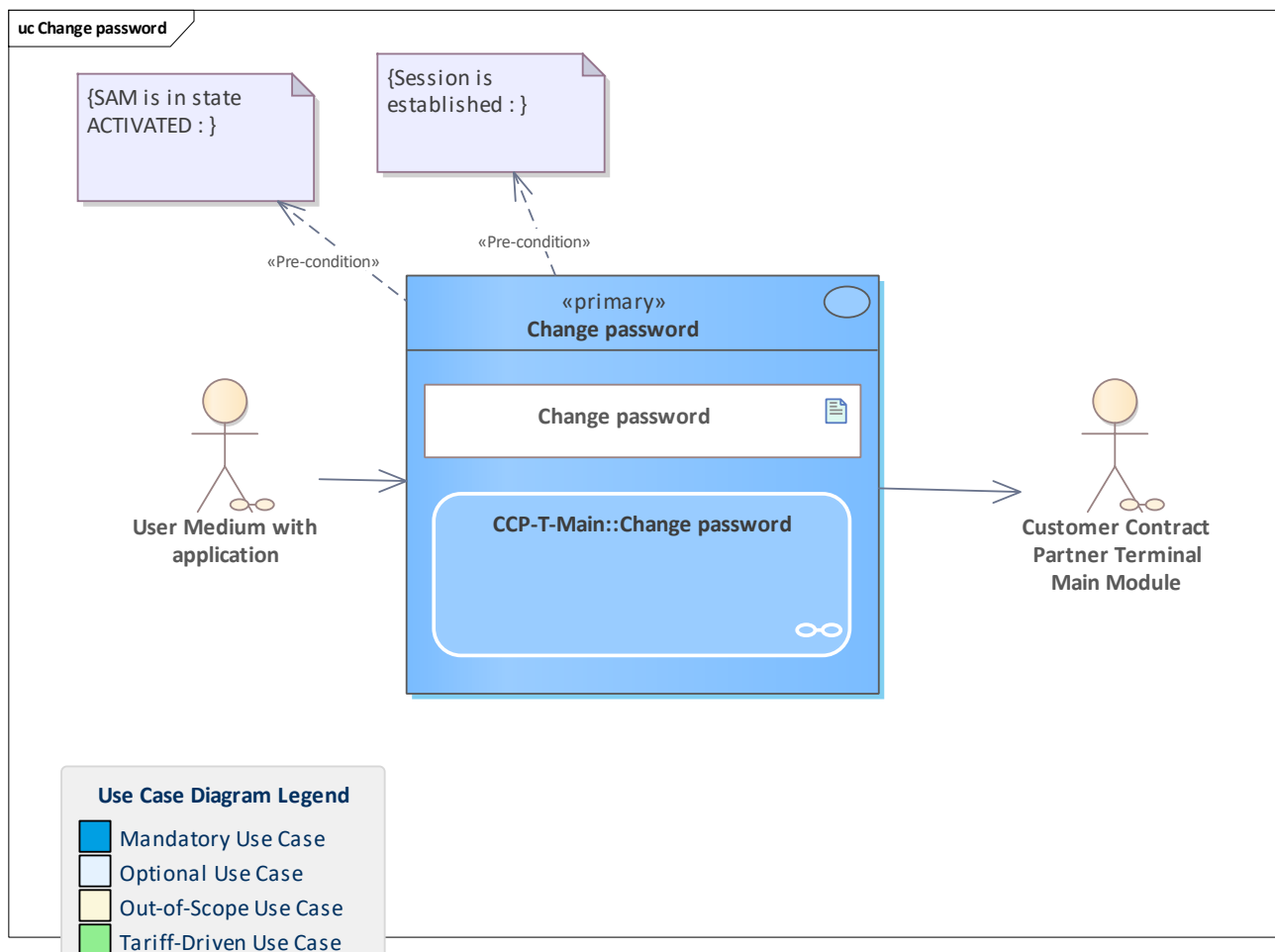
	Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Password
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Initialise password

13.2.2 Write password configuration



Use Case	Write password configuration
Description	The terminal writes the password configuration (user authentication data) to the user medium application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established Session is established (based on certificates) SAM is in state ACTIVATED
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Password parameters : PasswordParameters
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Write password configuration

13.2.3 Change password

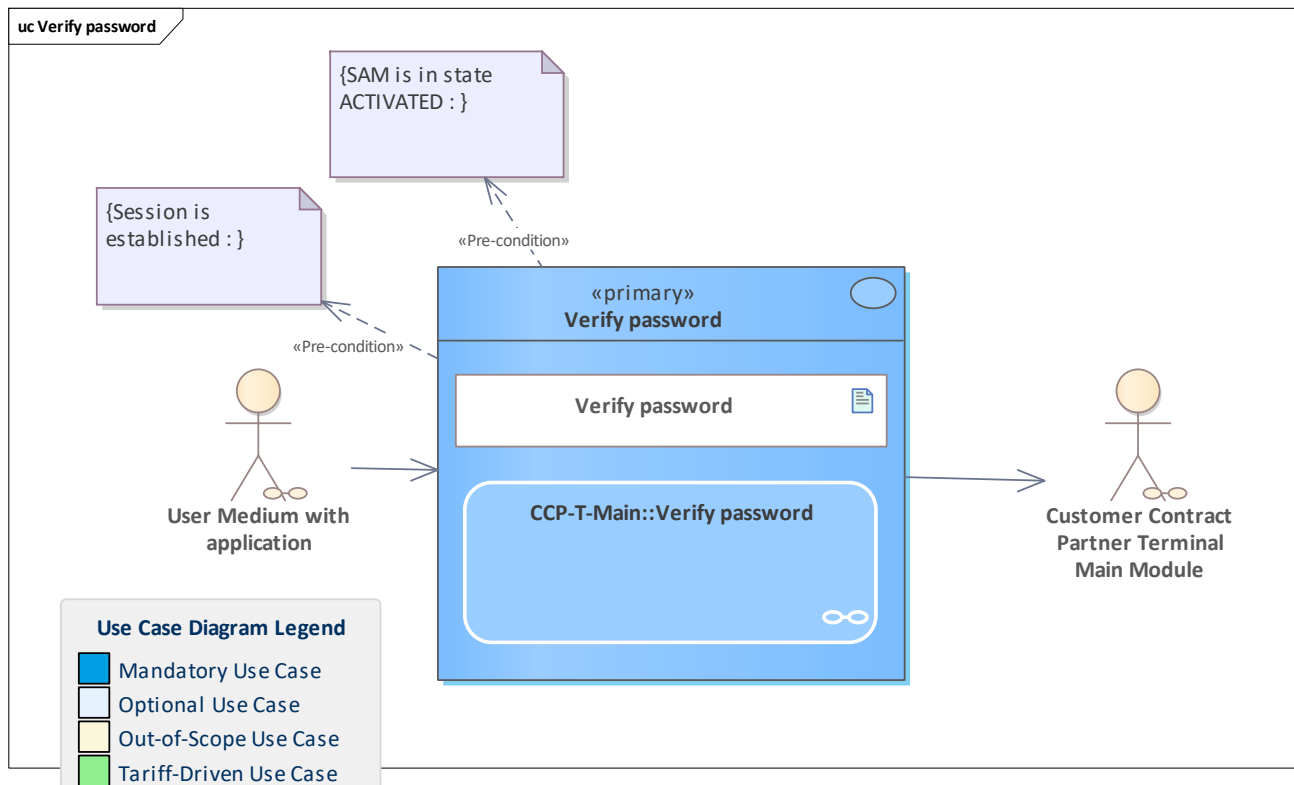


Use Case	Change password
Description	An end customer changes the password of a user medium with application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	New password Old password
Outputs	
Error Cases	



Activity Diagram	CCP-T-Main::Change_password
-------------------------	---

13.2.4 Verify password



Use Case	Verify password
Description	A customer enters her/his password and it will be verified by the user medium with application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Password
Outputs	
Error Cases	Invalid password
Activity Diagram	CCP-T-Main::Verify password



14 Electronic Ticket Bundle CCP-Terminal

Functionality bundle that covers the use cases for a CCP terminal with electronic tickets placed on a user medium with application (e.g. chip card).

14.1 Overview

Issue entitlement

Perform entitlement issuance and notify

Take back entitlement

Perform entitlement termination and notify

Reimburse and terminate electronic ticket

Change entitlement

Display entitlement

Optional: Save electronic ticket as favourite

Optional: Write favourites

Optional: Change favourites

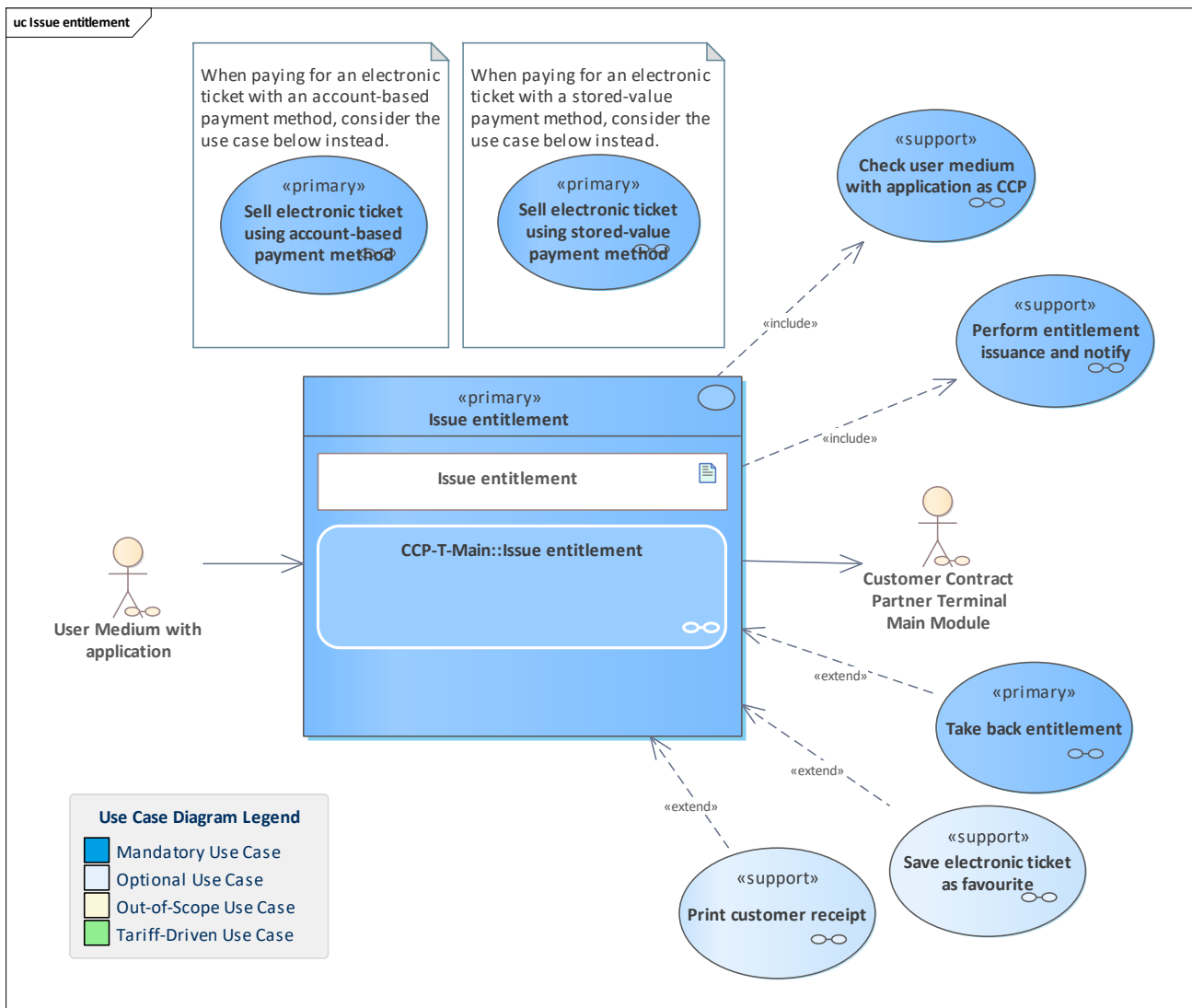
Optional: Delete favourites

Optional: Display favourites

Optional: Print customer receipt

14.2 Use Cases

14.2.1 Issue entitlement

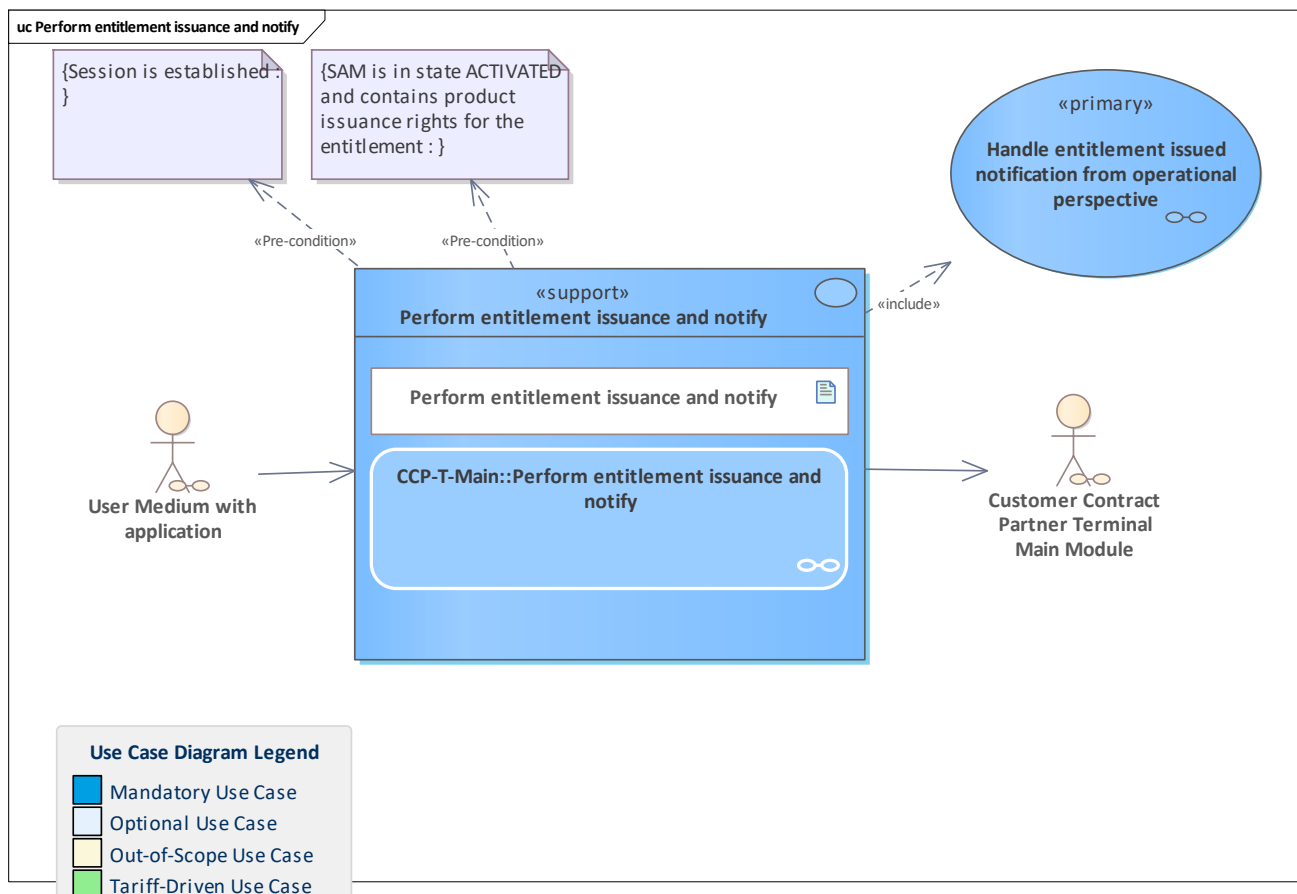


Use Case	<u>Issue entitlement</u>
Description	An entitlement is issued to a user medium. This can be an electronic ticket or a payment method. Note: if a stored-value payment method is issued and the current balance is not equal to zero, the payment means involved in the implicit recharging must match the ones given in the product parameters and may additionally be given as part of the entitlement issued metadata.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>SAM is in state ACTIVATED and contains product issuance rights for the entitlement</u>
Postconditions	
Linked Use Cases (Extended By)	<u>Take back entitlement</u> / <u>Take back entitlement</u> / <u>Print customer receipt</u> / <u>Save electronic ticket as favourite</u> / <u>Print customer receipt</u> / <u>Save electronic ticket as favourite</u> / <u>Save electronic ticket as favourite</u> / <u>Print customer receipt</u>
Linked Use Cases (Includes)	<u>Check user medium with application as CCP</u> / <u>Perform entitlement issuance and notify</u> / <u>Check user medium with application as CCP</u> / <u>Issue entitlement</u> / <u>Issue entitlement</u>



Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Replaced entitlement : EntitlementId</u> <u>Payment means</u> <u>Payment type : PaymentMeansCode</u> <u>Product billing type : AccountingProcedureTypeCode</u> <u>Infotext : Infotext</u> <u>Product parameters : ProductParameters</u> <u>Stored-value autload parameters : StoredValueAutoloadParameters</u> <u>Stored-value parameters : StoredValueParameters</u> <u>CCP organisation ID : OrganisationId</u> <u>Product ID : ProductId</u> <u>Entitlement effective time : EntitlementEffectiveTime</u> <u>Entitlement expiration time : EntitlementExpirationTime</u>
Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Issue entitlement</u>

14.2.2 Perform entitlement issuance and notify

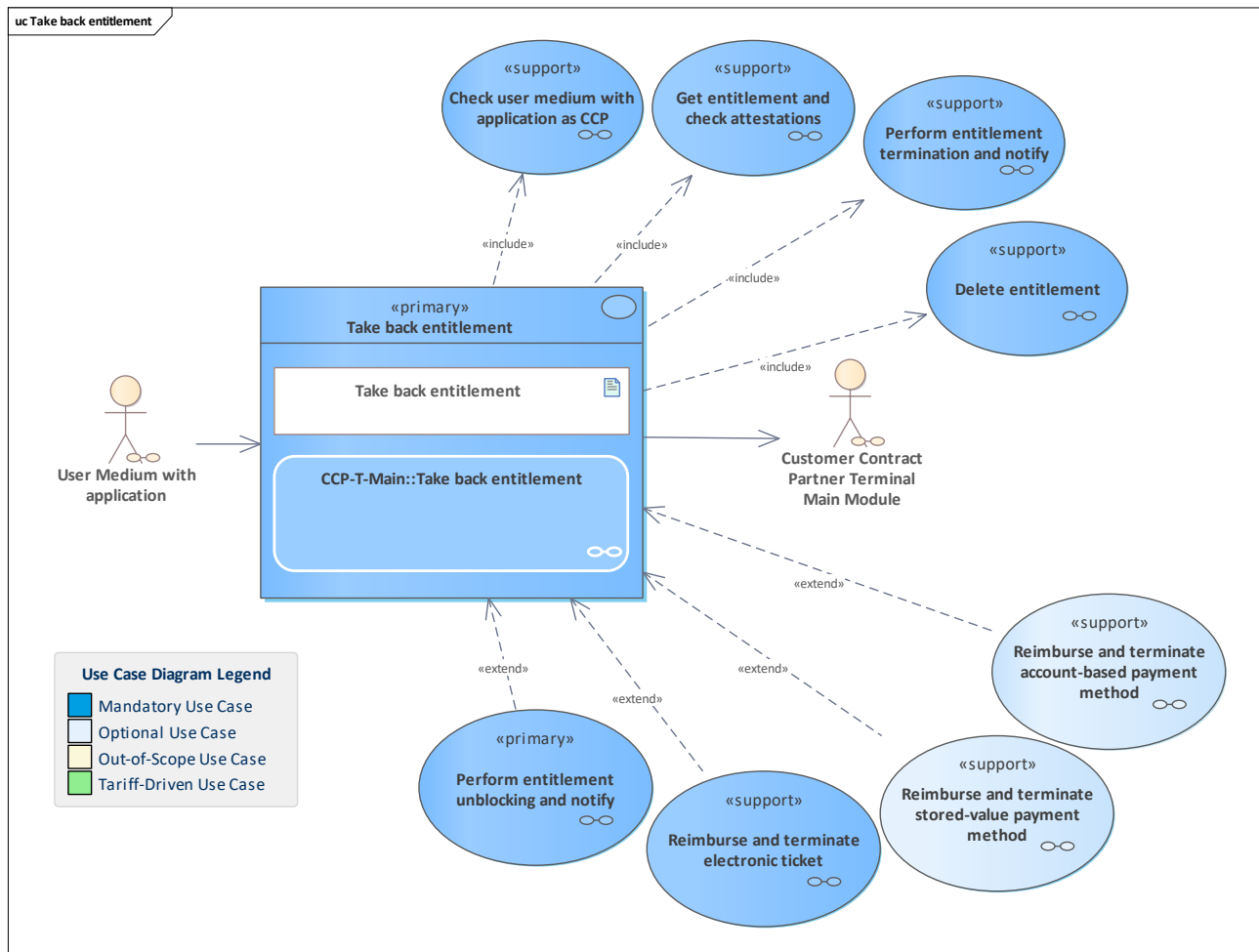


Use Case	<u>Perform entitlement issuance and notify</u>
Description	The CCP terminal performs a transaction to issue an entitlement to a user medium with application. The CCP back-office system is notified about the issuance. If the transaction is aborted, the CCP back-office system is also informed.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>Session is established</u> <u>SAM is in state ACTIVATED and contains product issuance rights for the entitlement</u>
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	<u>Handle entitlement issued notification from operational perspective</u>
Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Stored-value autoloading parameters :</u> <u>StoredValueAutoloadParameters</u> <u>Stored-value parameters : StoredValueParameters</u> <u>Replaced entitlement : EntitlementId</u> <u>Payment means</u>



	Payment type : PaymentMeansCode Product billing type : AccountingProcedureTypeCode Infotext : Infotext Product parameters : ProductParameters Entitlement expiration time : EntitlementExpirationTime Entitlement effective time : EntitlementEffectiveTime CCP organisation ID : OrganisationId Product ID : ProductId Entry ID : EntryId
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform entitlement issuance and notify

14.2.3 Take back entitlement

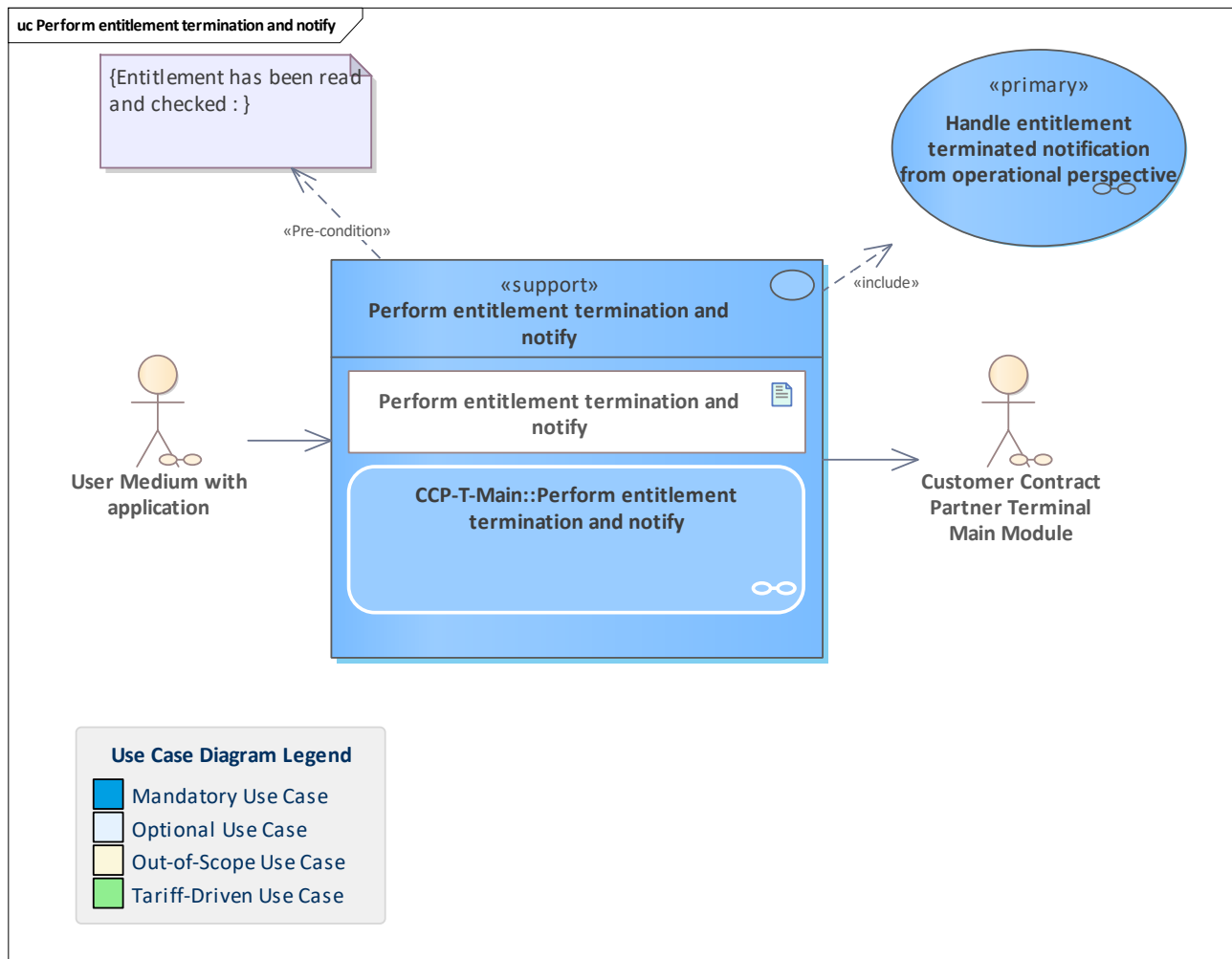


Use Case	Take back entitlement
Description	An entitlement is marked as terminated to prevent any further usage of the entitlement. Usually, the entitlement is deleted from the user medium application directly afterwards. This process may optionally involve reimbursement towards the customer in case it is executed at a terminal of the entitlement owner (pCCP).
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Reimburse and terminate account-based payment method / Reimburse and terminate stored-value payment method / Reimburse and terminate electronic ticket / Get entitlement and check attestations / Delete entitlement / Perform entitlement unblocking and notify
Linked Use Cases (Includes)	Perform entitlement termination and notify / Get entitlement and check attestations / Check user medium with application as CCP / Perform entitlement termination and notify / Perform entitlement termination and notify / Perform entitlement termination and notify / Delete entitlement



Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Take back entitlement

14.2.4 Perform entitlement termination and notify

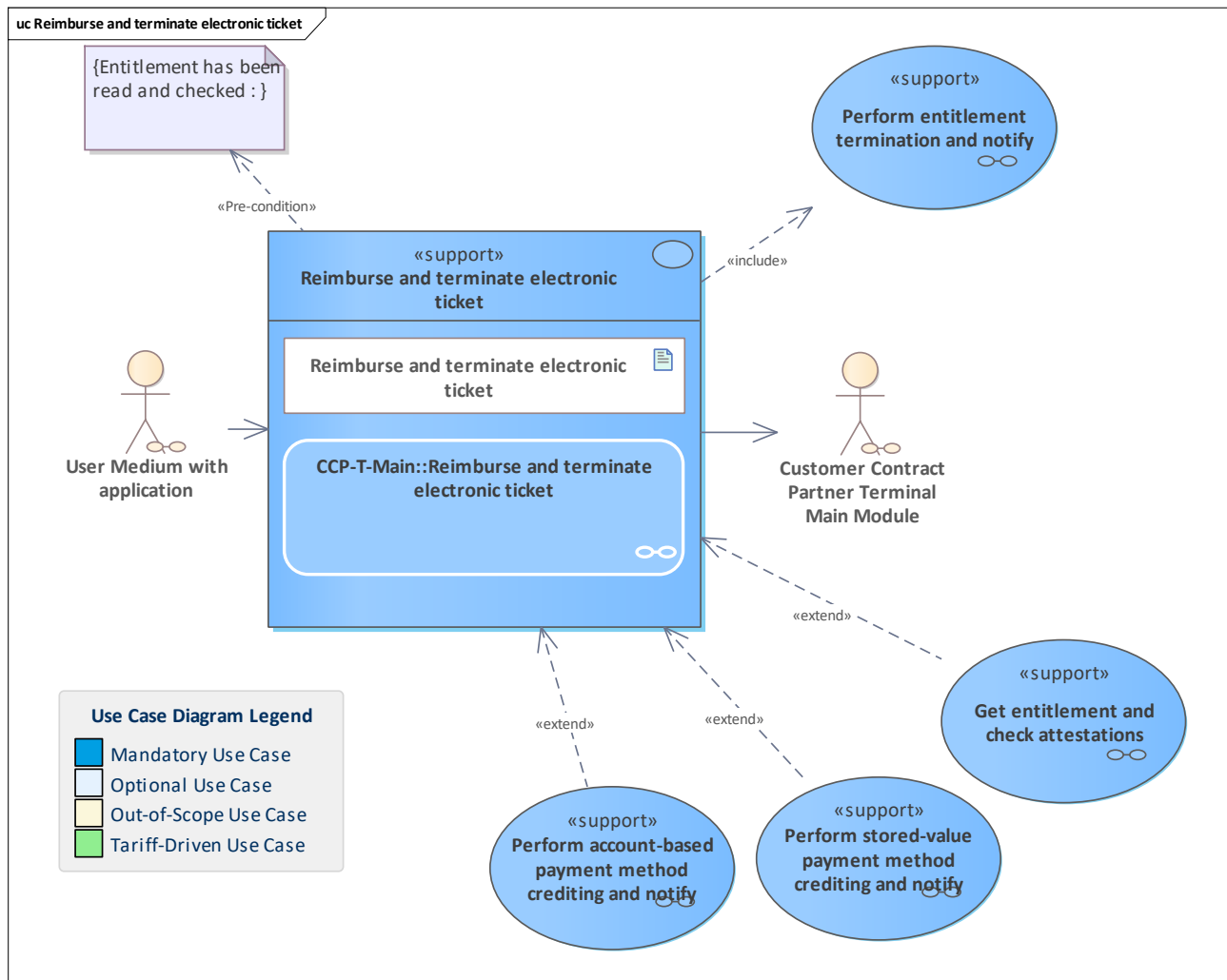


Use Case	Perform entitlement termination and notify
Description	Perform the transaction to terminate an entitlement in the CCP terminal and notify the responsible CCP back-office system (same CCP as for the terminal). If the transaction is aborted, the back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle entitlement terminated notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement directory entry : EntitlementDirectoryEntry



Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Perform entitlement termination and notify</u>

14.2.5 Reimburse and terminate electronic ticket

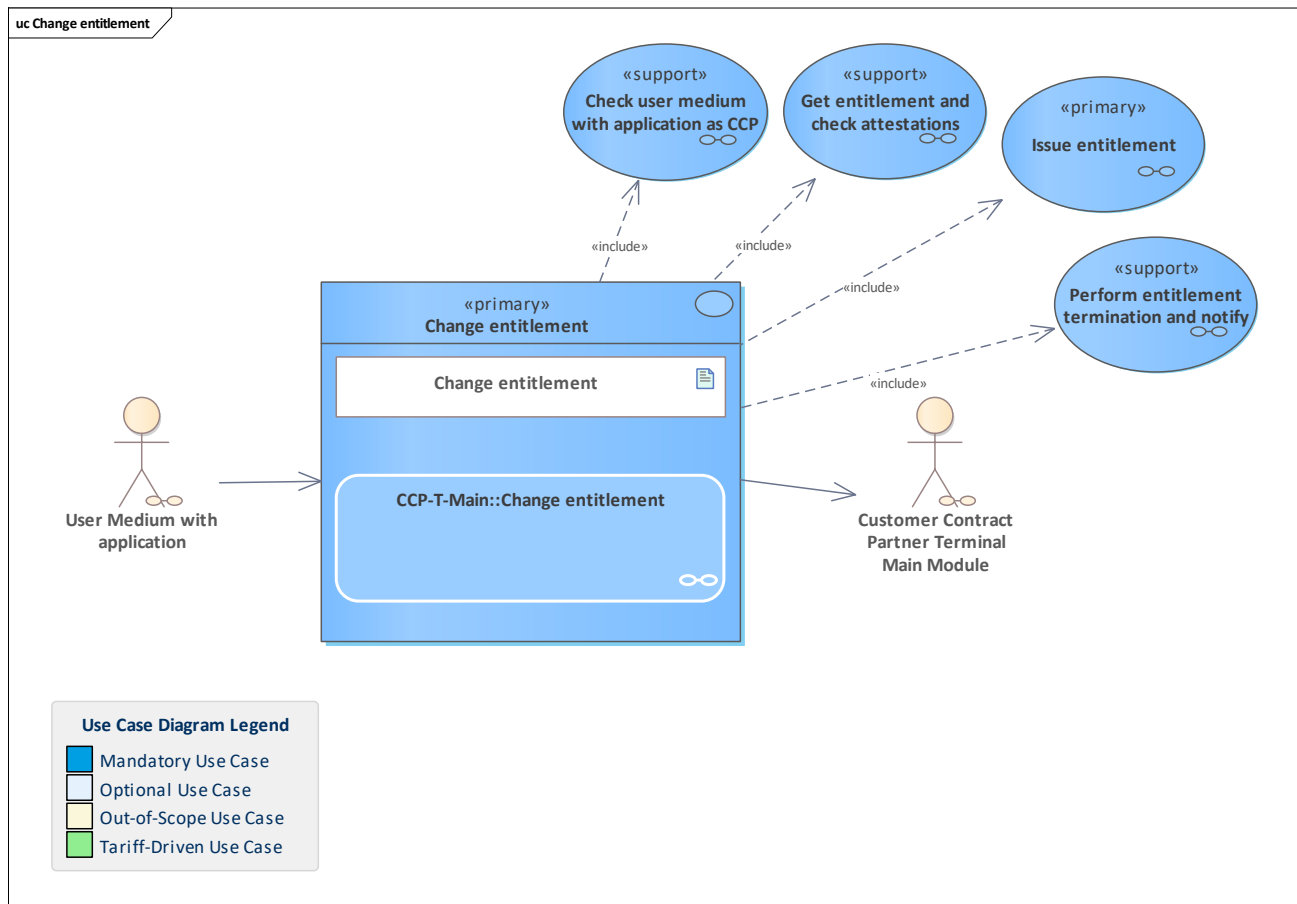


Use Case	Reimburse and terminate electronic ticket
Description	Reimburse and terminate an electronic ticket. The primary use case is Take back entitlement . Performed by the CCP terminal. The technical part is the termination of the electronic ticket has to be done in the terminal. Depending on the payment method that was used when purchasing the ticket, the reimbursement is done. If an ((etiCORE payment method was used, the refund is processed using this payment method. In this case, in addition to the termination, a credit action is done by the terminal. The termination is done and notified to the responsible CCP back-office system, as well as the potential credit action.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked Entitlement has been read and checked Entitlement has been read and checked Entitlement has been read and checked
Postconditions	Entitlement has been read and checked



Linked Use Cases (Extended By)	Perform account-based payment method crediting and notify / Perform stored-value payment method crediting and notify / Get entitlement and check attestations
Linked Use Cases (Includes)	Perform entitlement termination and notify
Linked Use Cases (Realises)	Entitlement has been read and checked
Base Activity	
Inputs	Electronic ticket entry : EntitlementDirectoryEntry Electronic ticket : Entitlement
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Reimburse and terminate electronic ticket

14.2.6 Change entitlement

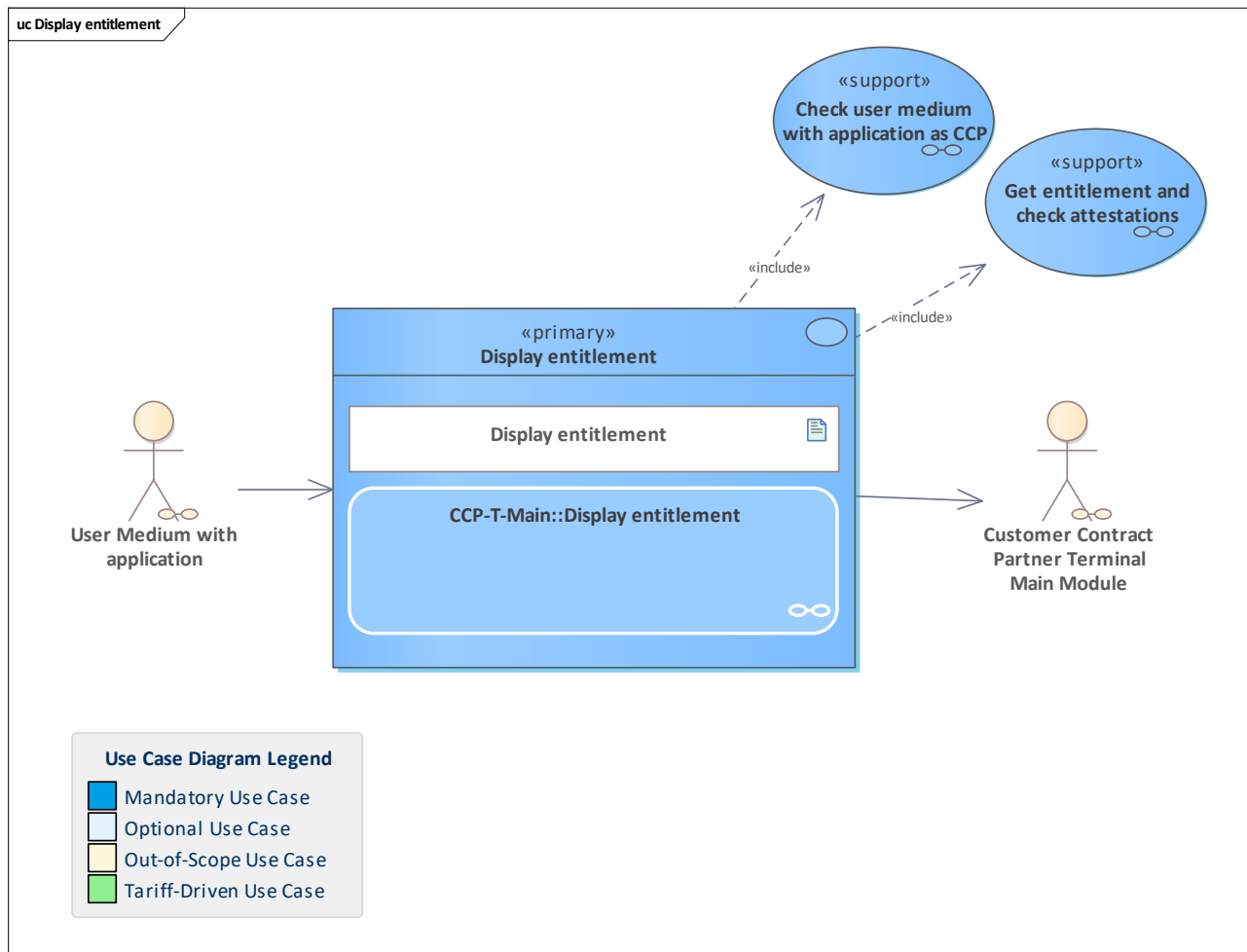


Use Case	Change entitlement
Description	<p>An entitlement needs to be replaced with a new one, for example, due to changed product parameters.</p> <p>Please note that it is assumed that there is no need for any payment transaction.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Issue entitlement / Check user medium with application as CCP / Perform entitlement termination and notify / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	



Error Cases	
Activity Diagram	<u>CCP-T-Main::Change entitlement</u>

14.2.7 Display entitlement

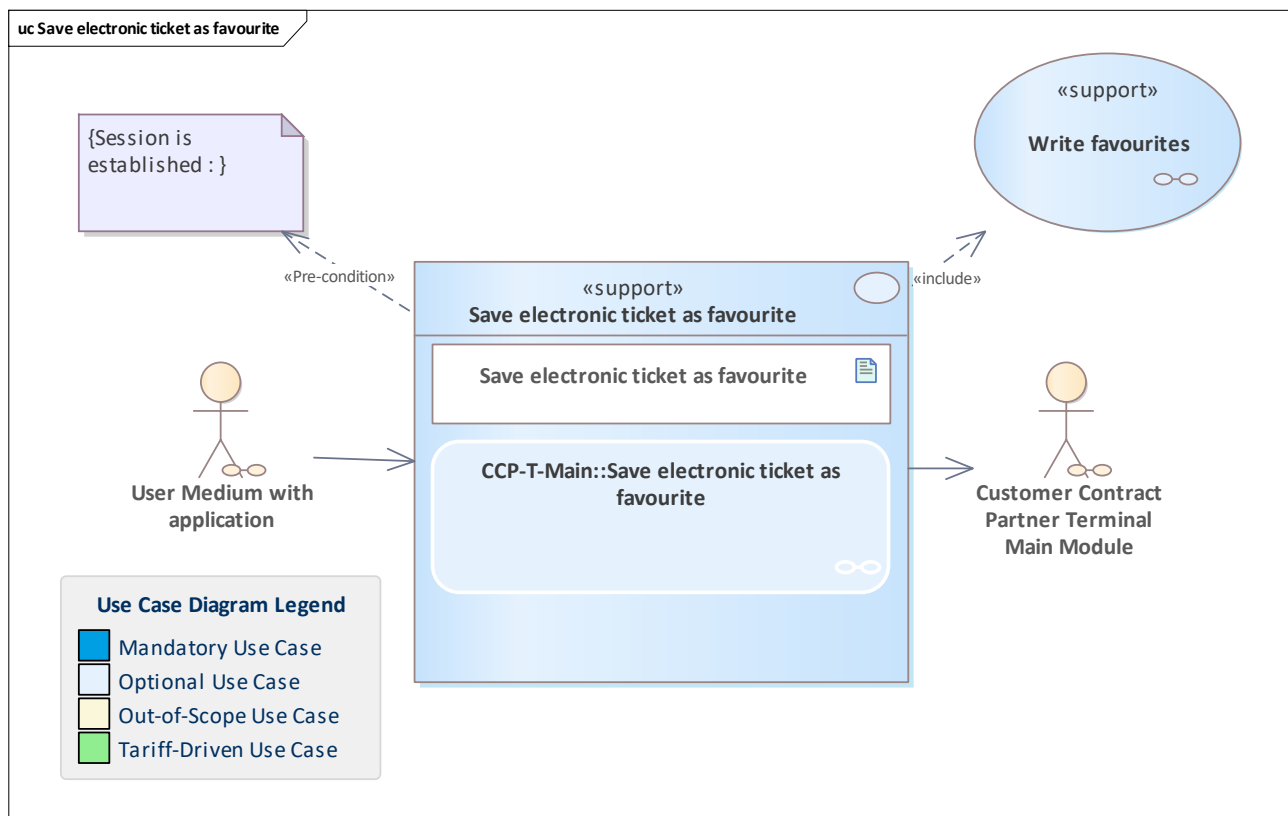


Use Case	Display entitlement
Description	Display an entitlement on a user medium with an application, independent of its status and validity period.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	



Activity Diagram	CCP-T-Main::Display_entitlement
-------------------------	---

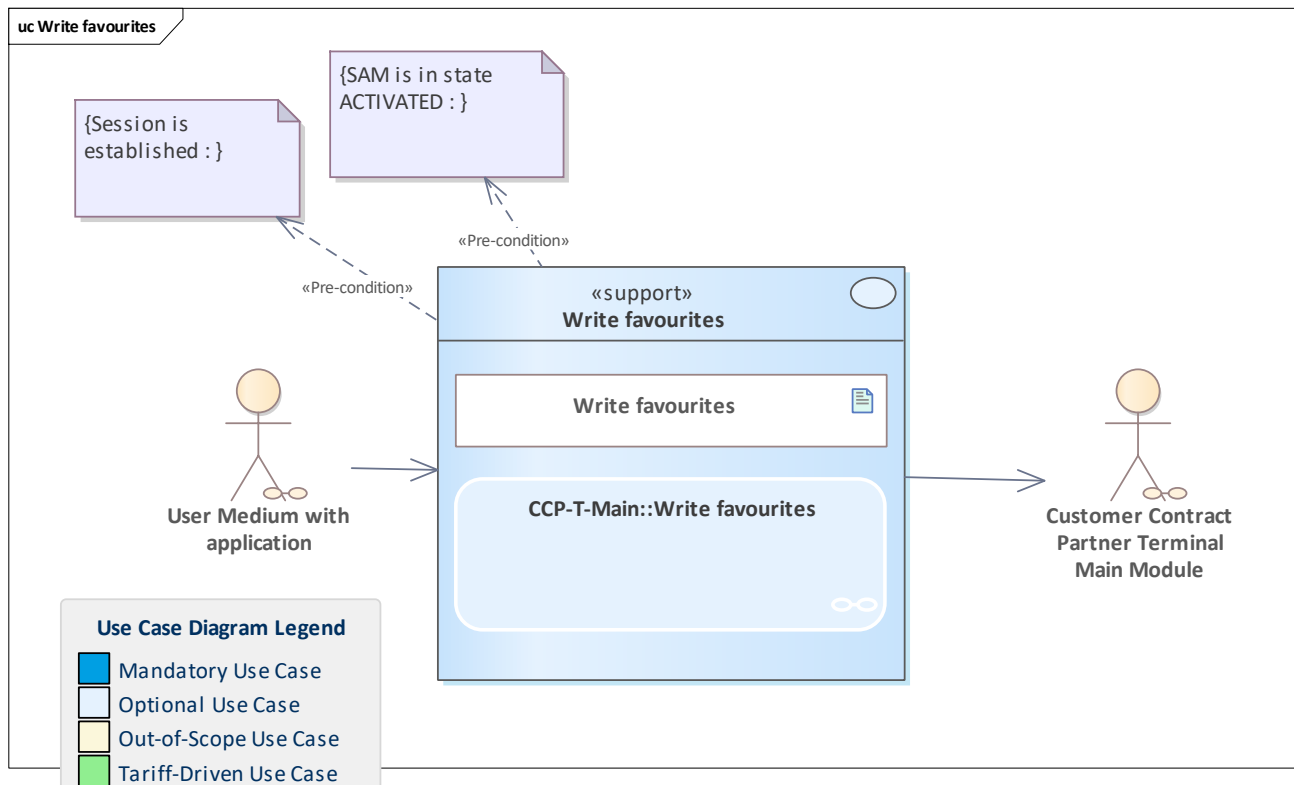
14.2.8 Optional: Save electronic ticket as favourite



Use Case	Save electronic ticket as favourite
Description	Support use case that allows an authorised CCP terminal to save the information about an electronic ticket as a favourite.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Session is established Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Write favourites
Linked Use Cases (Realises)	
Base Activity	
Inputs	Product parameters : ProductParameters Infotext : Infotext Product ID : ProductId
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Save electronic ticket as favourite

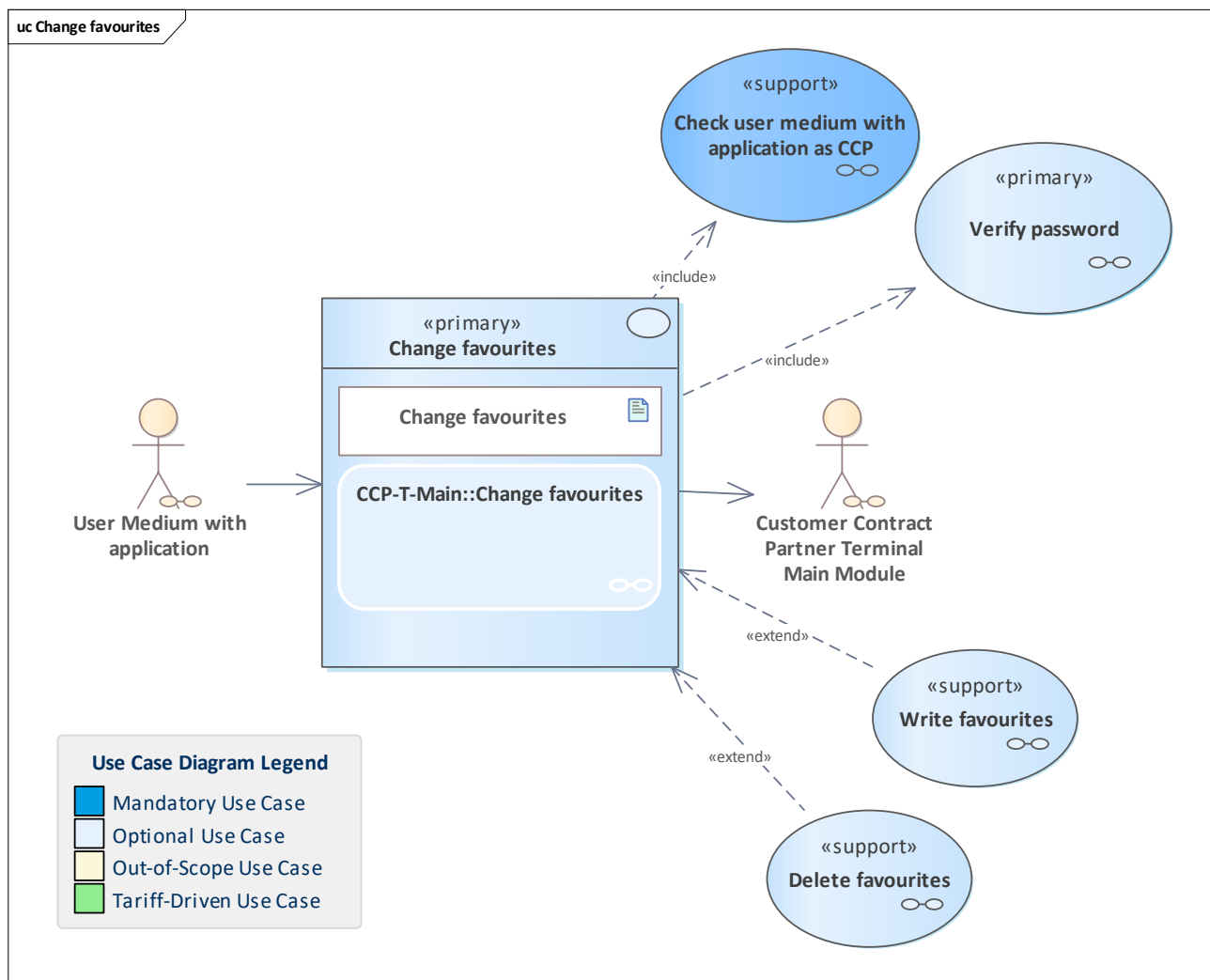


14.2.9 Optional: Write favourites



Use Case	Write favourites
Description	Write the favourites data object to the user medium with an application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Favourites : Favourites
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Write favourites

14.2.10 Optional: Change favourites

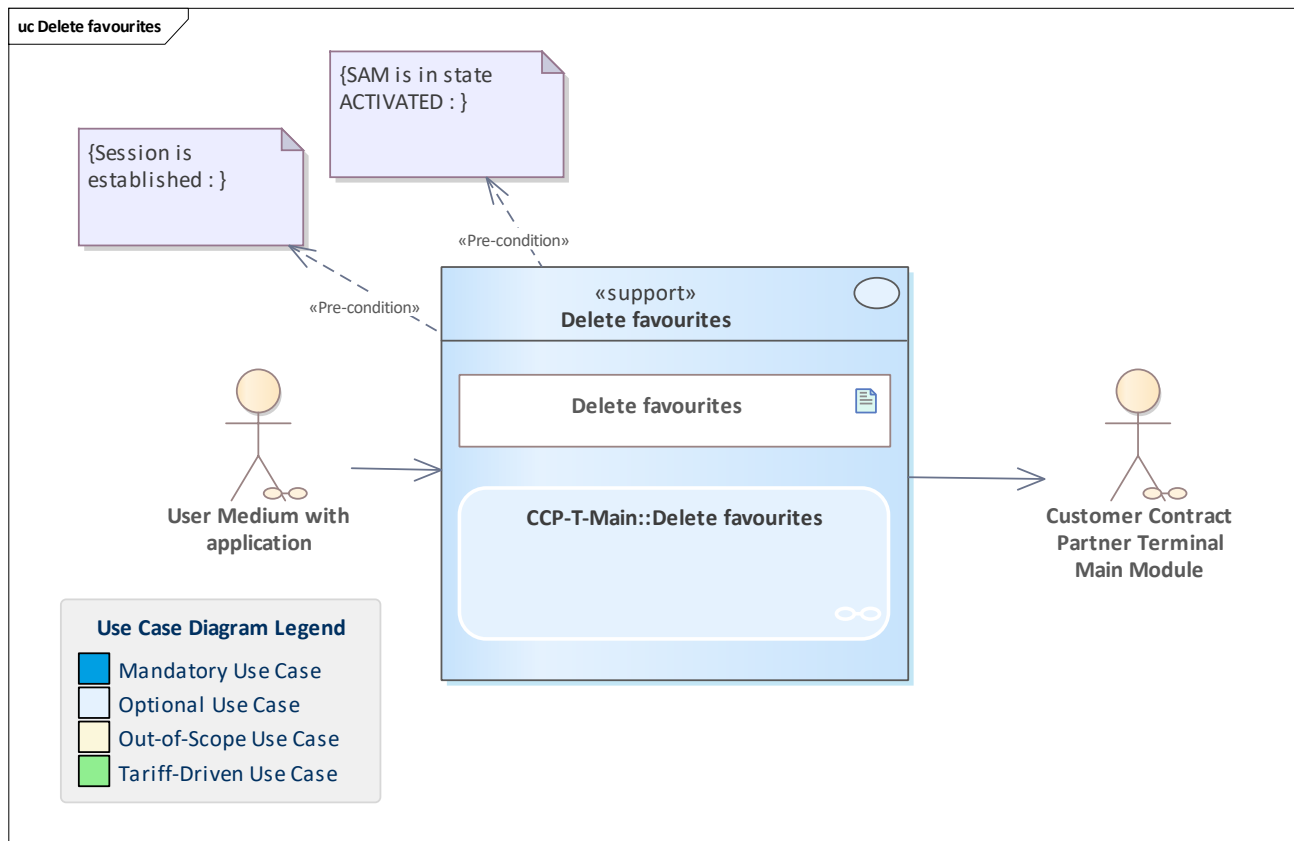


Use Case	Change favourites
Description	The customer wants to change his favourites aided by a CCP-T. The customer must enter his password/PIN to change the favourites.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Write favourites / Delete favourites
Linked Use Cases (Includes)	Verify password / Check user medium with application as CCP
Linked Use Cases (Realises)	
Base Activity	
Inputs	



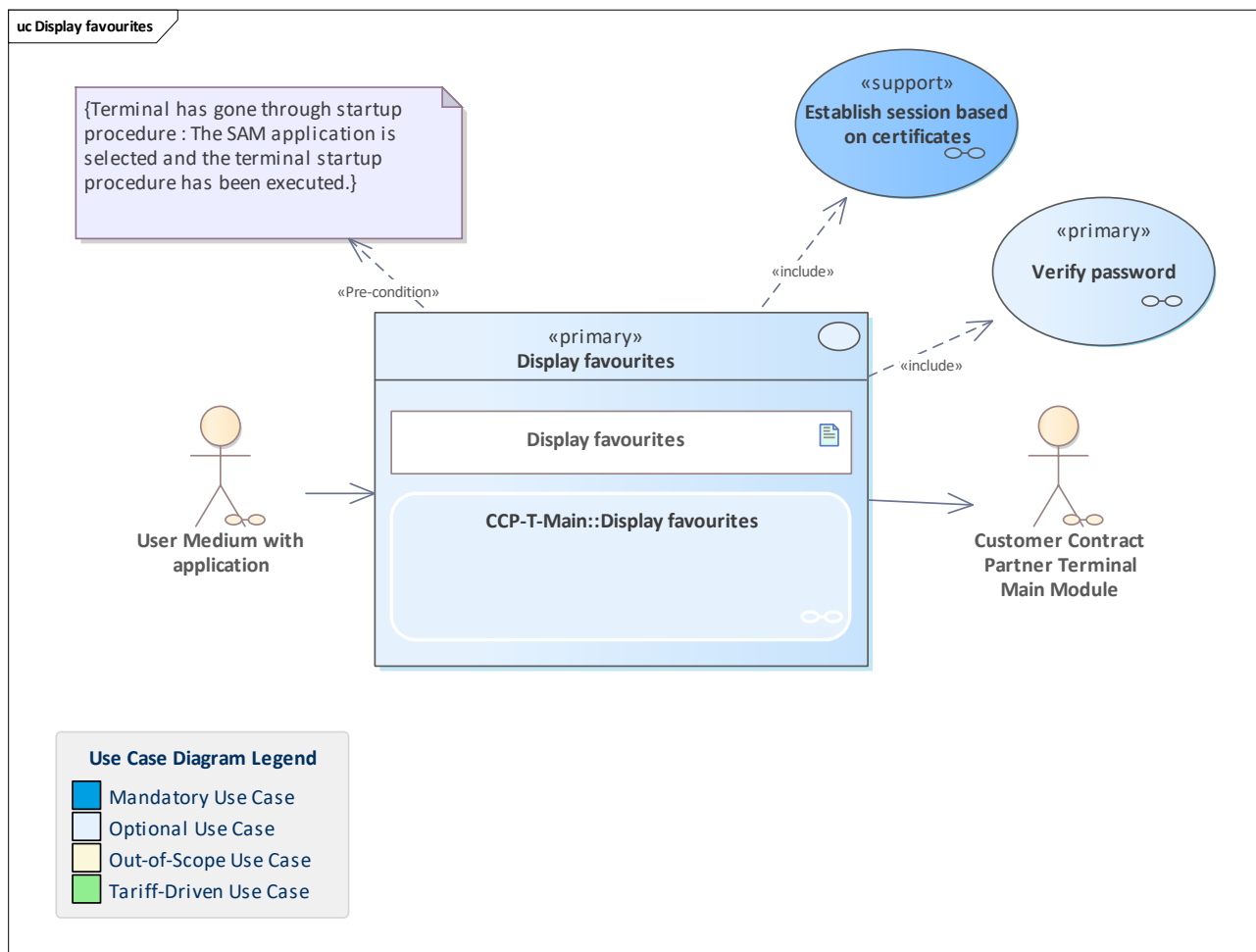
Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Change favourites</u>

14.2.11 Optional: Delete favourites



Use Case	Delete favourites
Description	Delete the favourites data object on the user medium with an application.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Delete favourites

14.2.12 Optional: Display favourites

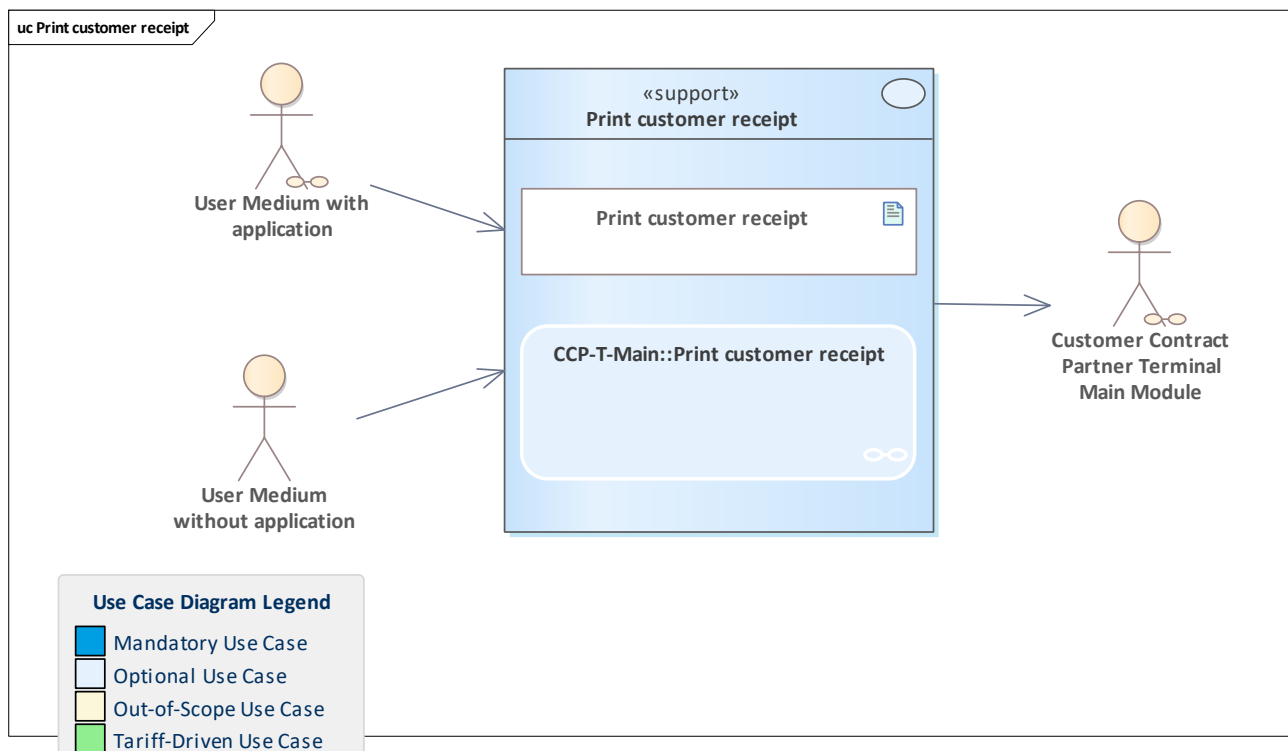


Use Case	<u>Display favourites</u>
Description	The customer wants to display his user medium-located favourites information on a CCP-T. The customer must enter his password/PIN to show the favourites information.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>Terminal has gone through startup procedure</u> <u>Terminal has gone through startup procedure</u>
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	<u>Verify password</u> / <u>Establish session based on certificates</u>
Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Password</u>
Outputs	



Error Cases	
Activity Diagram	<u>CCP-T-Main::Display favourites</u>

14.2.13 Optional: Print customer receipt



Use Case	Print customer receipt
Description	A customer receipt is printed e.g. after selling an entitlement or having entitlements displayed on a terminal.
Initiating Actor	User Medium with application User Medium without application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	List of entitlements : Entitlement
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Print customer receipt



15 Account-Based Payment Bundle CCP-Terminal

Functionality bundle that covers the use cases for a CCP terminal forming the basis for using the account-based payment method.

15.1 Overview

Issue entitlement

Perform entitlement issuance and notify

Take back entitlement

Perform entitlement termination and notify

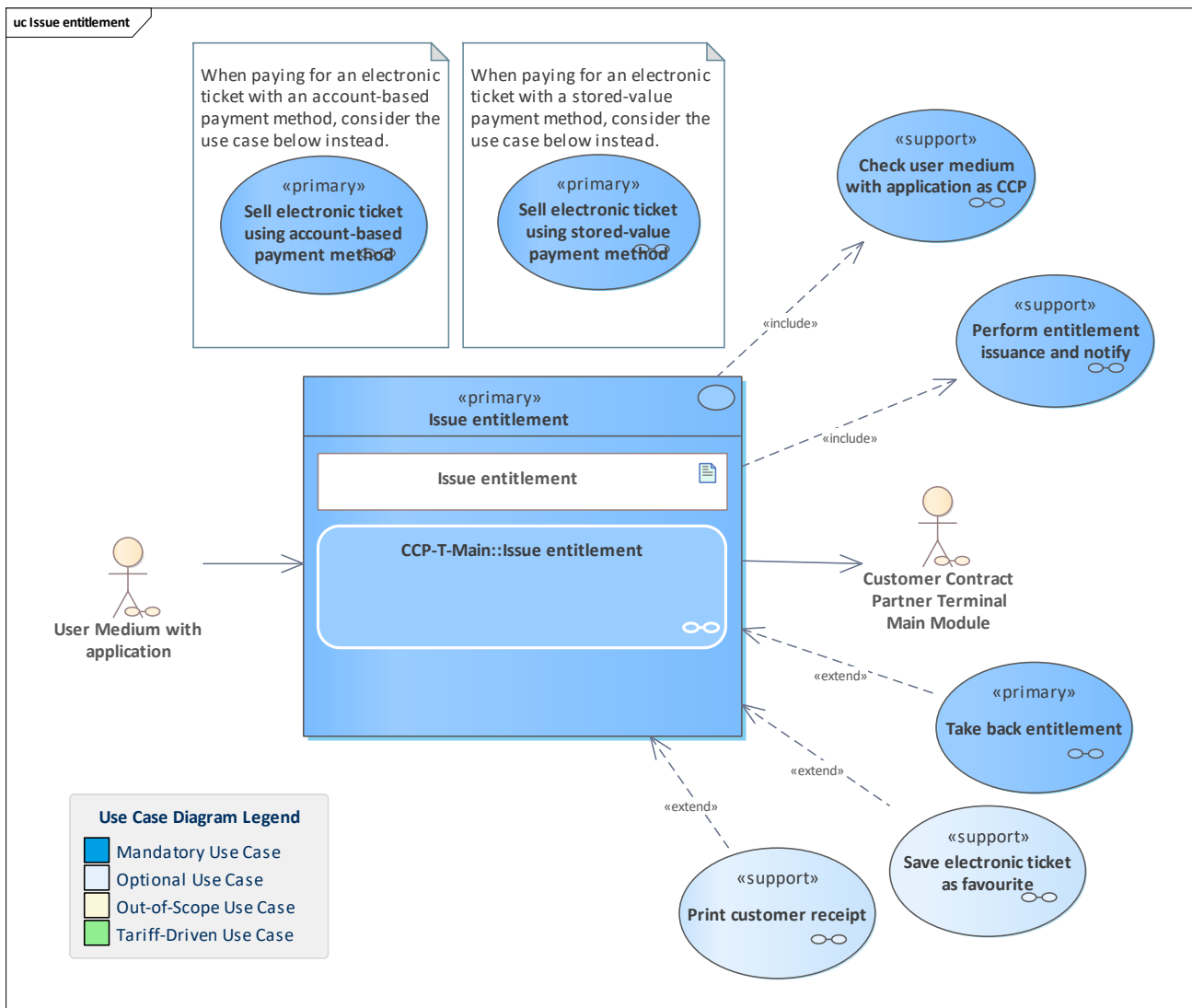
Change entitlement

Display entitlement

Optional: Reimburse and terminate account-based payment method

15.2 Use Cases

15.2.1 Issue entitlement

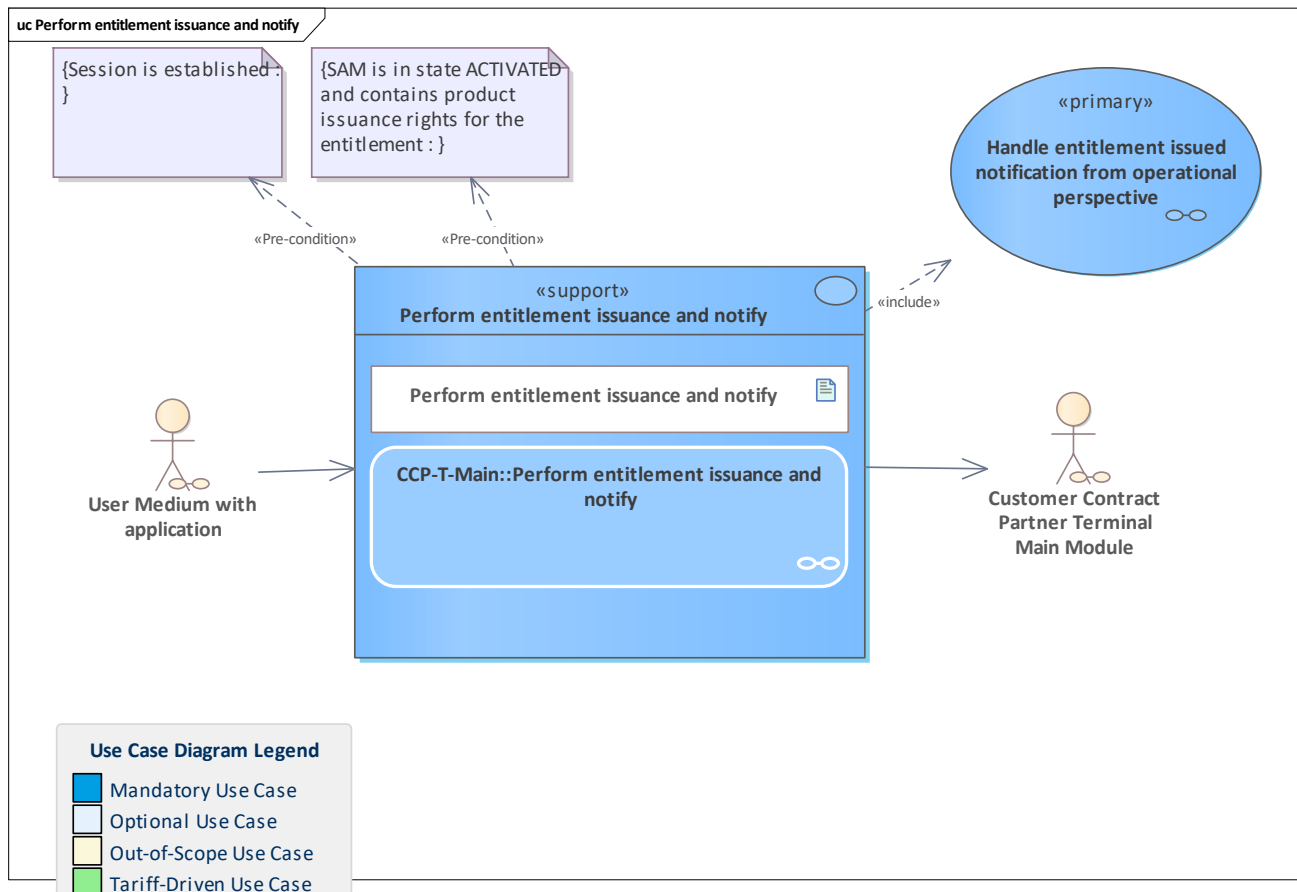


Use Case	<u>Issue entitlement</u>
Description	An entitlement is issued to a user medium. This can be an electronic ticket or a payment method. Note: if a stored-value payment method is issued and the current balance is not equal to zero, the payment means involved in the implicit recharging must match the ones given in the product parameters and may additionally be given as part of the entitlement issued metadata.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>SAM is in state ACTIVATED and contains product issuance rights for the entitlement</u>
Postconditions	
Linked Use Cases (Extended By)	<u>Take back entitlement</u> / <u>Take back entitlement</u> / <u>Print customer receipt</u> / <u>Save electronic ticket as favourite</u> / <u>Print customer receipt</u> / <u>Save electronic ticket as favourite</u> / <u>Save electronic ticket as favourite</u> / <u>Print customer receipt</u>
Linked Use Cases (Includes)	<u>Check user medium with application as CCP</u> / <u>Perform entitlement issuance and notify</u> / <u>Check user medium with application as CCP</u> / <u>Issue entitlement</u> / <u>Issue entitlement</u>



Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Replaced entitlement : EntitlementId</u> <u>Payment means</u> <u>Payment type : PaymentMeansCode</u> <u>Product billing type : AccountingProcedureTypeCode</u> <u>Infotext : Infotext</u> <u>Product parameters : ProductParameters</u> <u>Stored-value autload parameters : StoredValueAutoloadParameters</u> <u>Stored-value parameters : StoredValueParameters</u> <u>CCP organisation ID : OrganisationId</u> <u>Product ID : ProductId</u> <u>Entitlement effective time : EntitlementEffectiveTime</u> <u>Entitlement expiration time : EntitlementExpirationTime</u>
Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Issue entitlement</u>

15.2.2 Perform entitlement issuance and notify

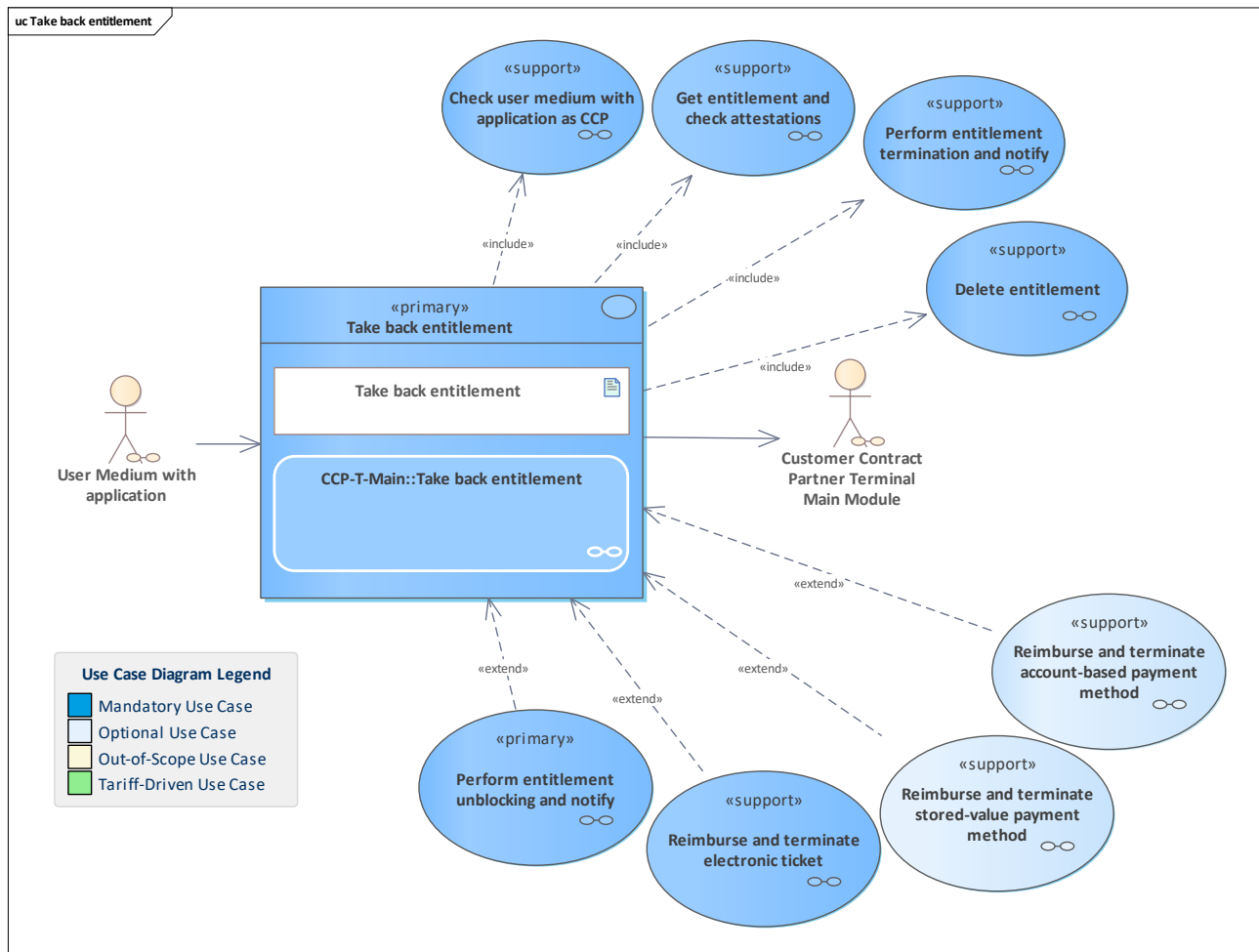


Use Case	<u>Perform entitlement issuance and notify</u>
Description	The CCP terminal performs a transaction to issue an entitlement to a user medium with application. The CCP back-office system is notified about the issuance. If the transaction is aborted, the CCP back-office system is also informed.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>Session is established</u> <u>SAM is in state ACTIVATED and contains product issuance rights for the entitlement</u>
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	<u>Handle entitlement issued notification from operational perspective</u>
Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Stored-value autoloading parameters :</u> <u>StoredValueAutoloadParameters</u> <u>Stored-value parameters : StoredValueParameters</u> <u>Replaced entitlement : EntitlementId</u> <u>Payment means</u>



	Payment type : PaymentMeansCode Product billing type : AccountingProcedureTypeCode Infotext : Infotext Product parameters : ProductParameters Entitlement expiration time : EntitlementExpirationTime Entitlement effective time : EntitlementEffectiveTime CCP organisation ID : OrganisationId Product ID : ProductId Entry ID : EntryId
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform entitlement issuance and notify

15.2.3 Take back entitlement

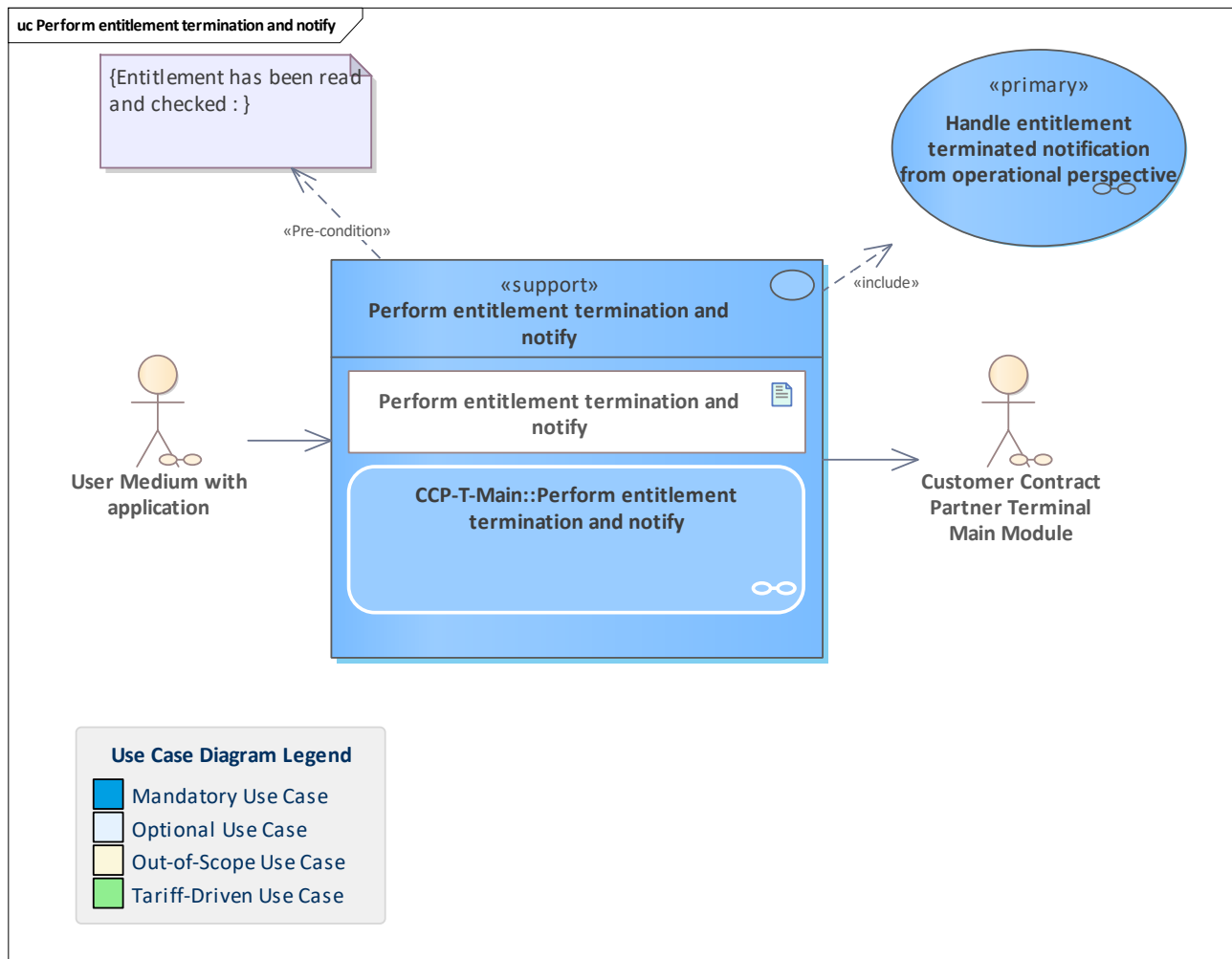


Use Case	<u>Take back entitlement</u>
Description	An entitlement is marked as terminated to prevent any further usage of the entitlement. Usually, the entitlement is deleted from the user medium application directly afterwards. This process may optionally involve reimbursement towards the customer in case it is executed at a terminal of the entitlement owner (pCCP).
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	<u>Reimburse and terminate account-based payment method</u> / <u>Reimburse and terminate stored-value payment method</u> / <u>Reimburse and terminate electronic ticket</u> / <u>Get entitlement and check attestations</u> / <u>Delete entitlement</u> / <u>Perform entitlement unblocking and notify</u>
Linked Use Cases (Includes)	<u>Perform entitlement termination and notify</u> / <u>Get entitlement and check attestations</u> / <u>Check user medium with application as CCP</u> / <u>Perform entitlement termination and notify</u> / <u>Perform entitlement termination and notify</u> / <u>Delete entitlement</u>



Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Take back entitlement

15.2.4 Perform entitlement termination and notify

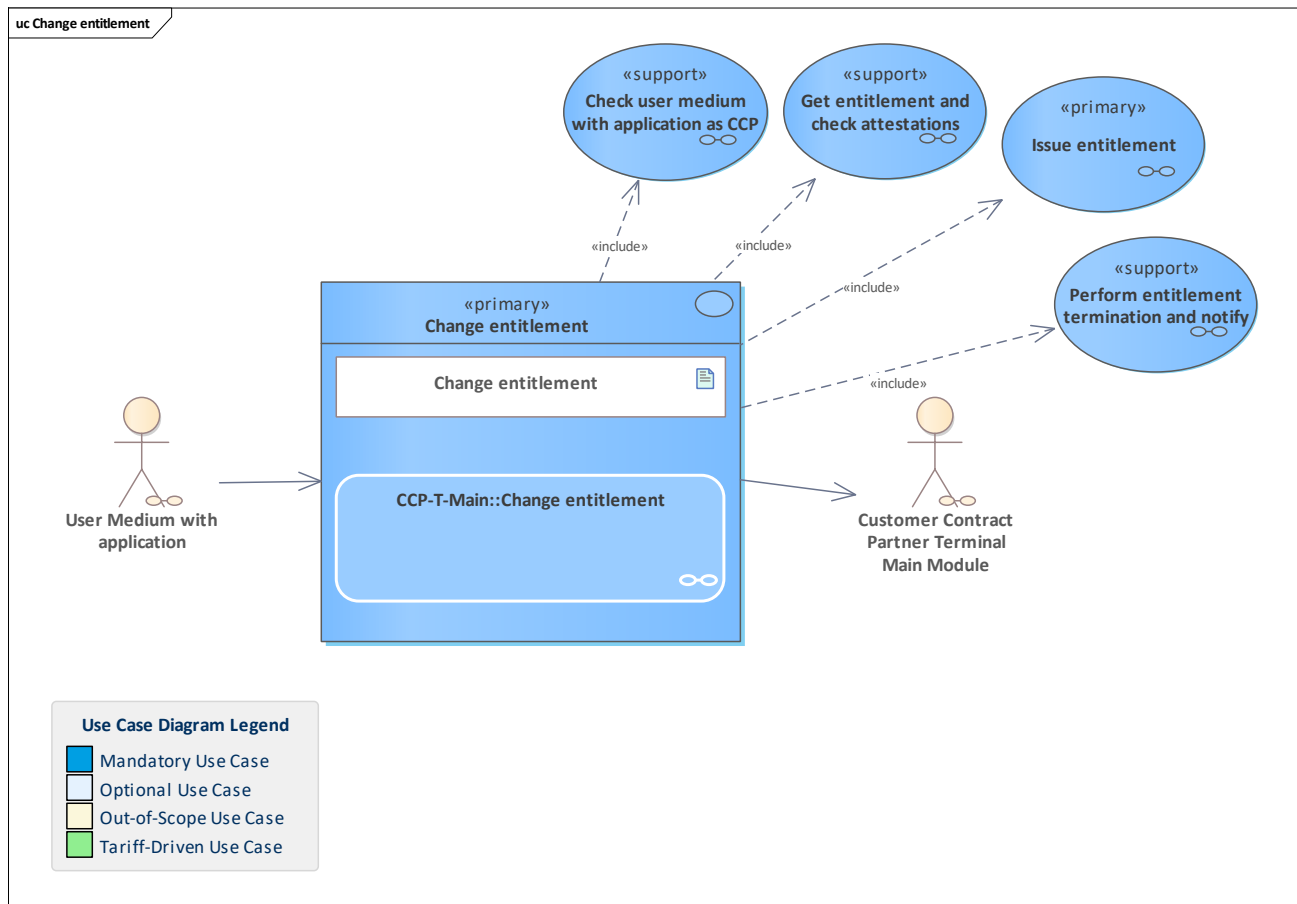


Use Case	Perform entitlement termination and notify
Description	Perform the transaction to terminate an entitlement in the CCP terminal and notify the responsible CCP back-office system (same CCP as for the terminal). If the transaction is aborted, the back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle entitlement terminated notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement directory entry : EntitlementDirectoryEntry



Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Perform entitlement termination and notify</u>

15.2.5 Change entitlement

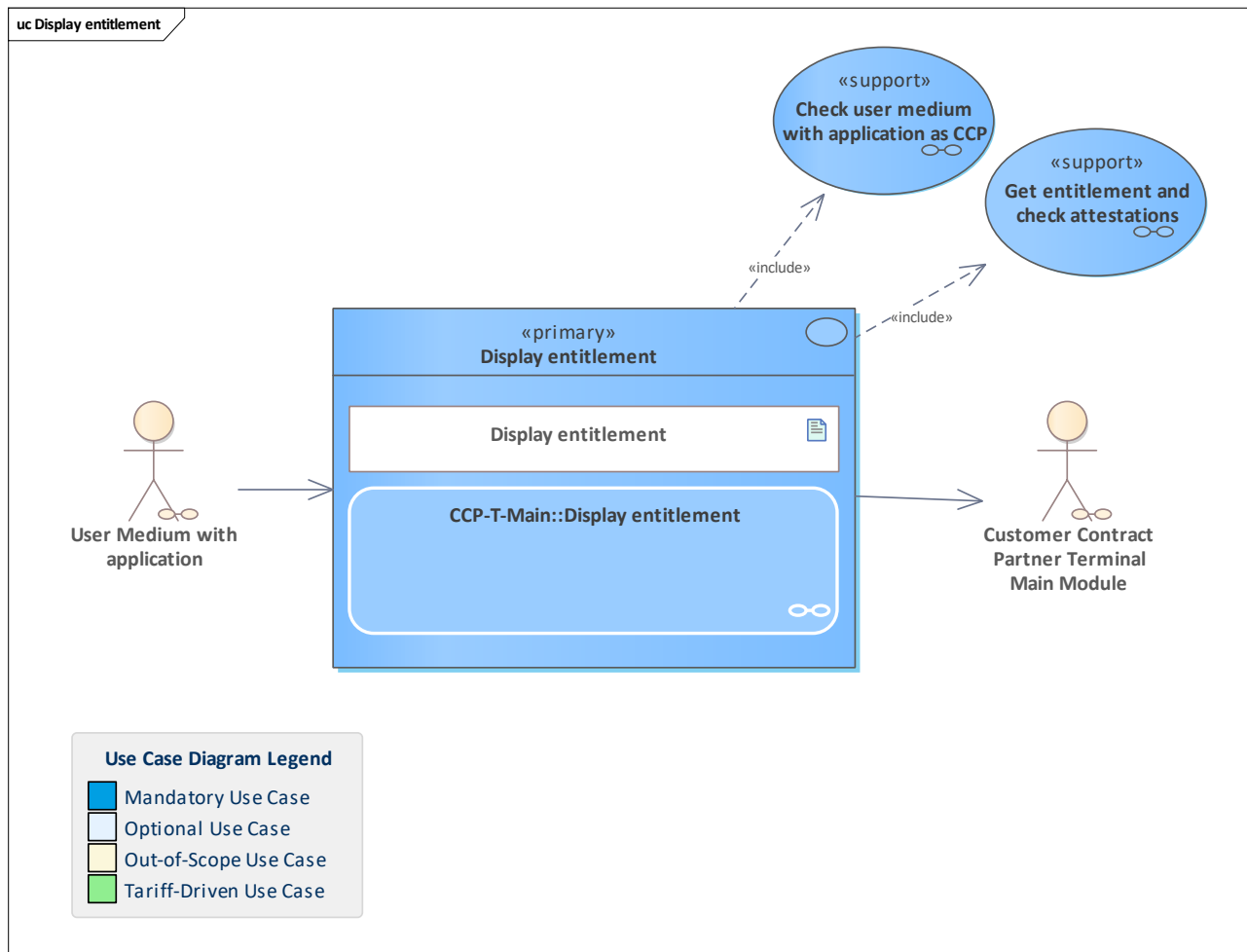


Use Case	Change entitlement
Description	<p>An entitlement needs to be replaced with a new one, for example, due to changed product parameters.</p> <p>Please note that it is assumed that there is no need for any payment transaction.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Issue entitlement / Check user medium with application as CCP / Perform entitlement termination and notify / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	



Error Cases	
Activity Diagram	<u>CCP-T-Main::Change entitlement</u>

15.2.6 Display entitlement

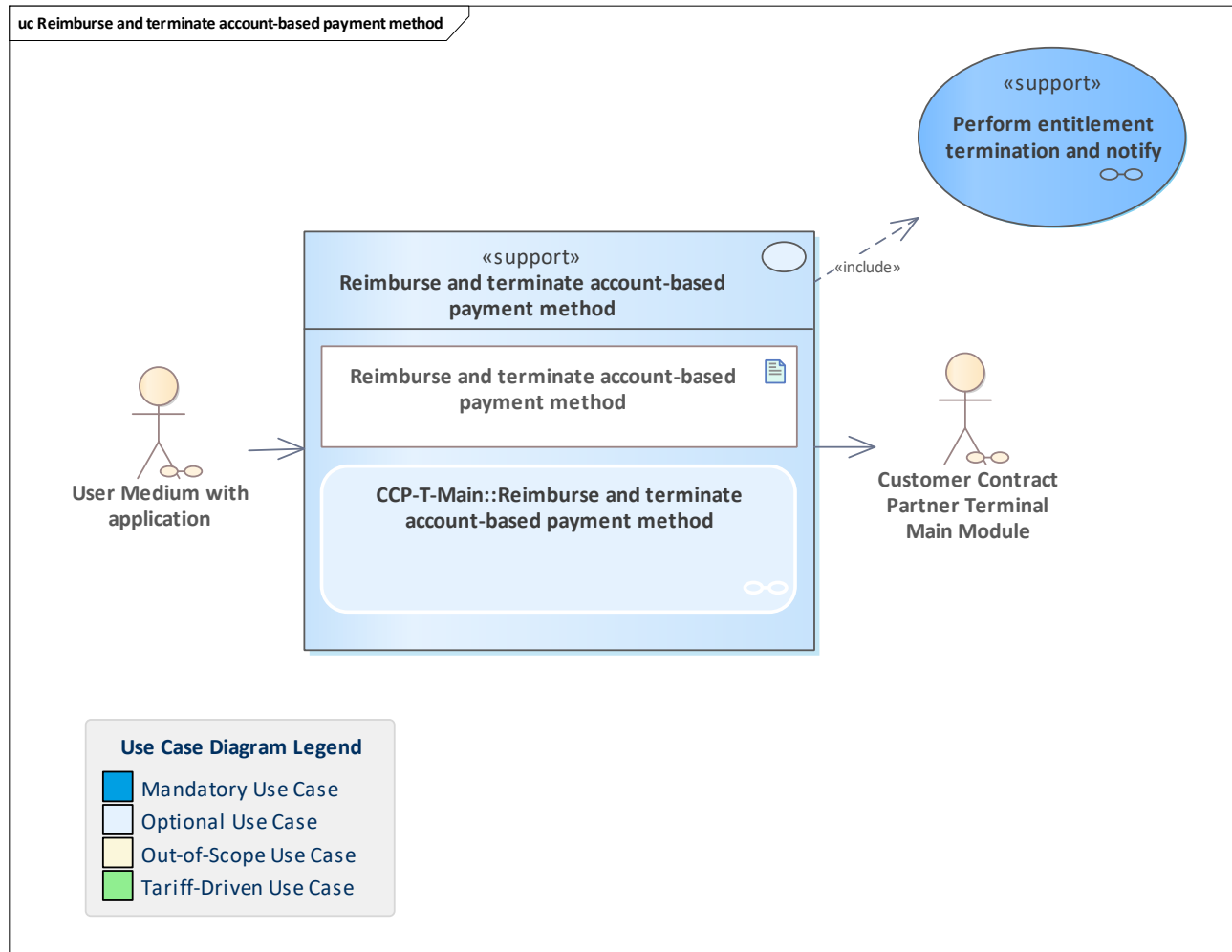


Use Case	Display entitlement
Description	Display an entitlement on a user medium with an application, independent of its status and validity period.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	



Activity Diagram	CCP-T-Main::Display_entitlement
-------------------------	---

15.2.7 Optional: Reimburse and terminate account-based payment method



Use Case	Reimburse and terminate account-based payment method
Description	<p>Reimburse and terminate an account-based payment method as a supporting use case. The primary use case is Take back entitlement.</p> <p>Performed by the CCP terminal. The technical part is the termination of the account-based payment method which has to be done in the terminal.</p> <p>A reimbursement becomes necessary if the account-based payment method is based on a prepaid account with a remaining balance, or if a subscription was cancelled in the middle of the term.</p> <p>The termination is done and notified to the responsible CCP back-office system.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases	



(Extended By)	
Linked Use Cases (Includes)	Perform entitlement termination and notify
Linked Use Cases (Realises)	
Base Activity	
Inputs	Account-based payment method entry : EntitlementDirectoryEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Reimburse and terminate account-based payment method



16 Stored-Value Payment Bundle CCP-Terminal

Functionality bundle that covers the use cases for a CCP terminal forming the basis for using the account-based payment method.

16.1 Overview

Recharge stored-value payment method

Perform stored-value payment method recharging and notify

Optional: Autoload stored-value payment method

Reimburse stored-value payment method

Perform stored-value payment method reimbursing and notify

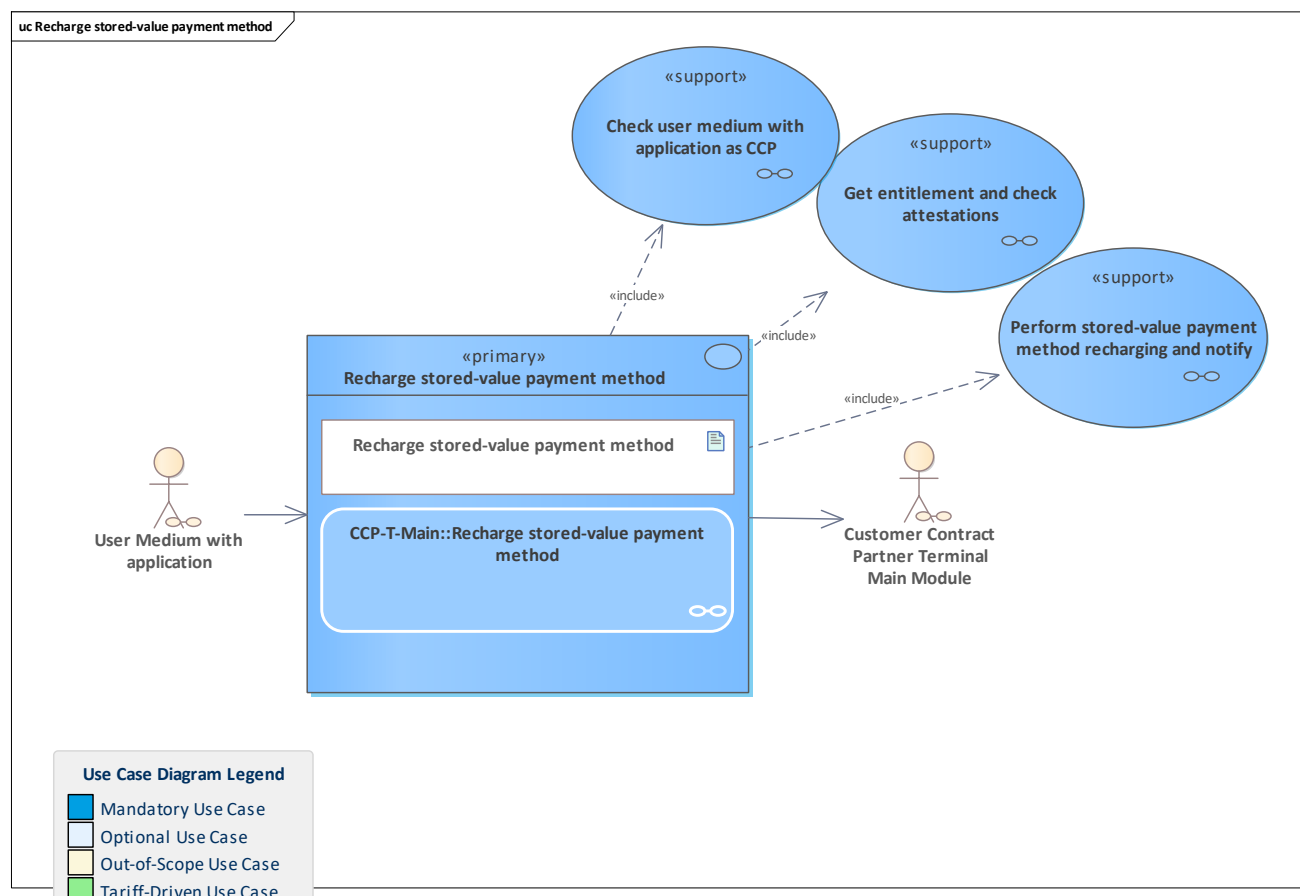
Reimburse and terminate stored-value payment method

Change entitlement

Display entitlement

16.2 Use Cases

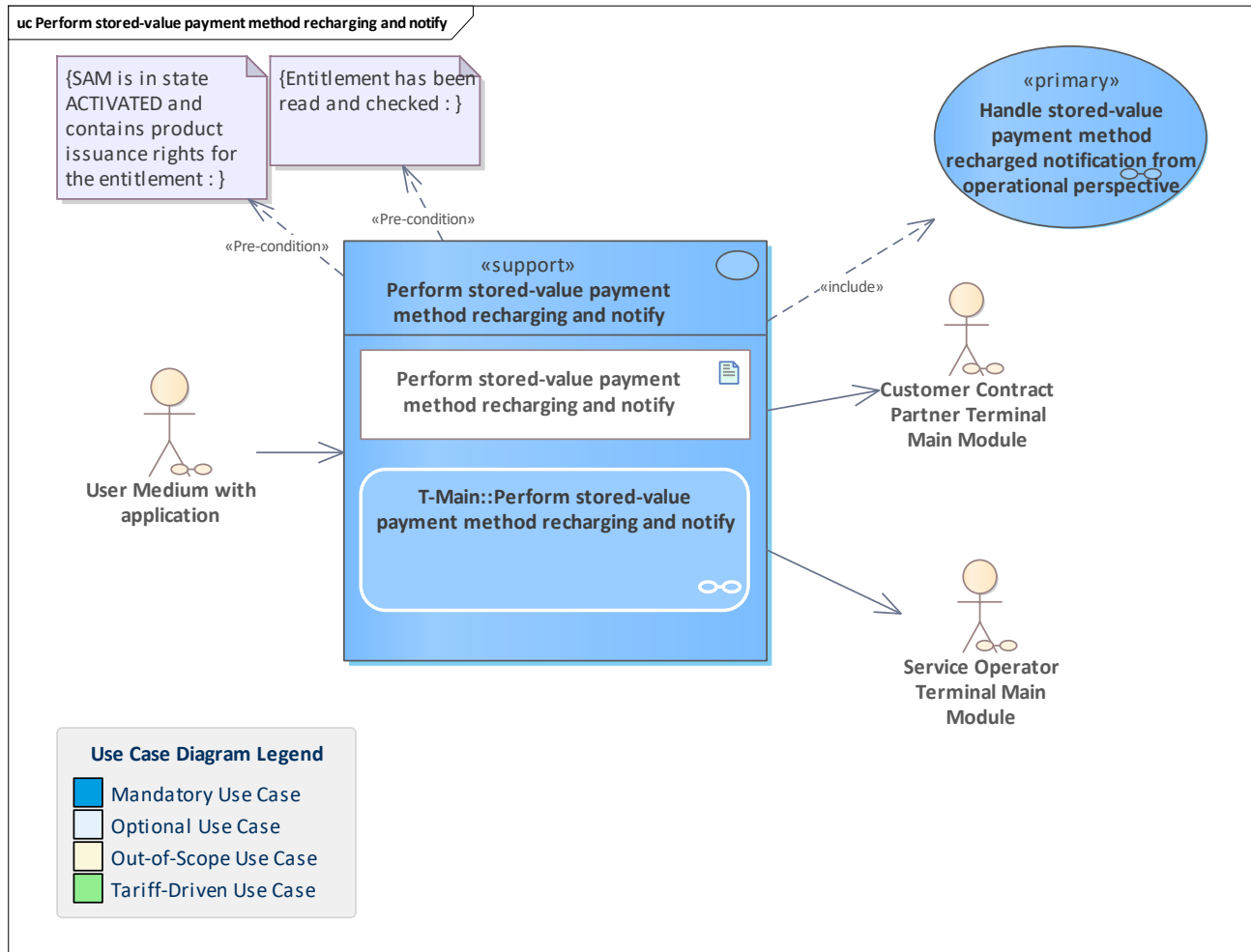
16.2.1 Recharge stored-value payment method





Use Case	Recharge stored-value payment method
Description	<p>The CCP terminal lets the customer recharge a chosen amount onto his stored-value payment method.</p> <p>The use case checks the stored-value payment method and ensures, that the maximum balance in the stored-value account is not exceeded for the intended recharge transaction.</p> <p>Finally, the recharge is performed and notified to the CCP back-office system (same CCP as the terminal operator of the current terminal).</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the entitlement
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform stored-value payment method recharging and notify / Check user medium with application as CCP / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Recharge stored-value payment method

16.2.2 Perform stored-value payment method recharging and notify

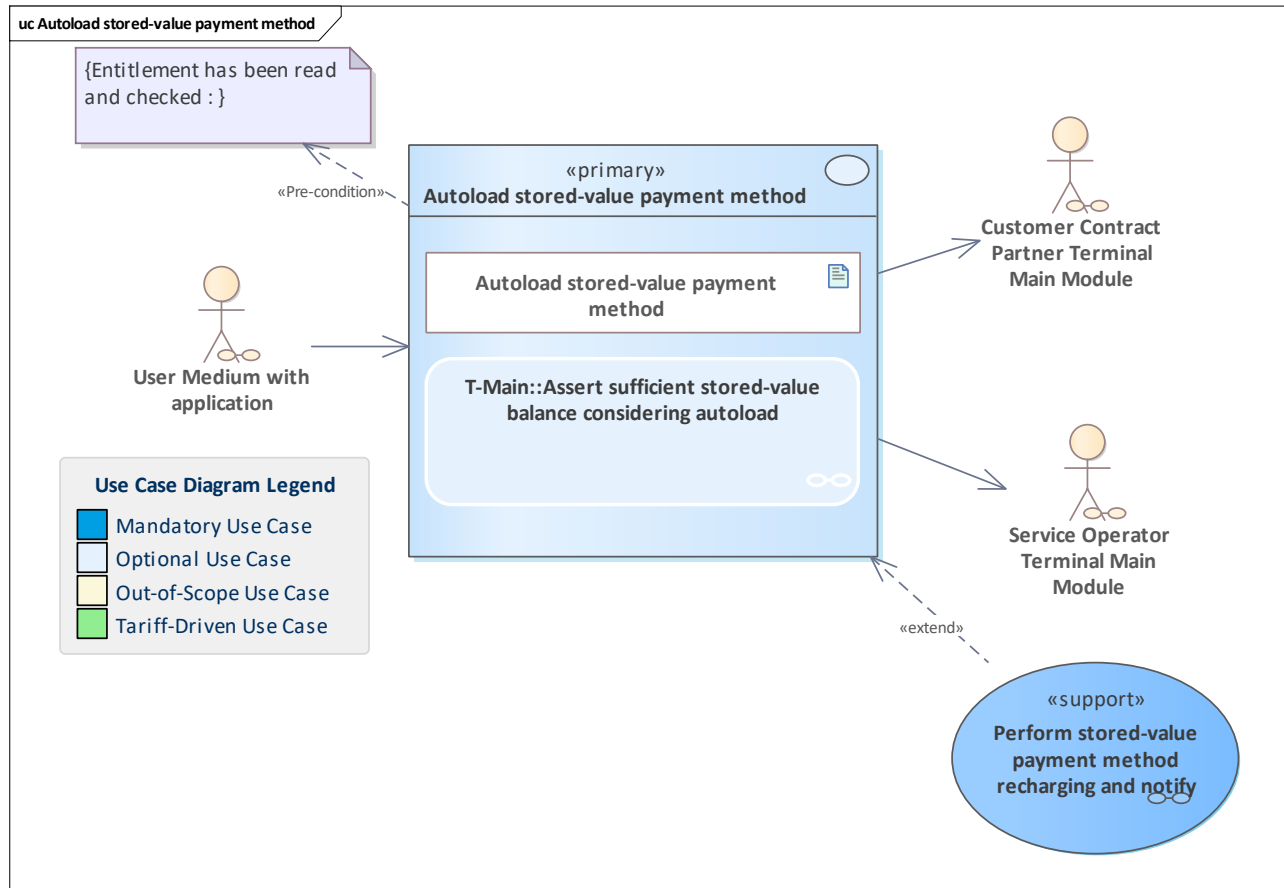


Use Case	Perform stored-value payment method recharging and notify
Description	<p>The terminal performs the transaction to recharge a stored-value payment method and notifies the corresponding back-office system about it.</p> <p>In case of a performed autoloading-triggered recharge in a CICO terminal, the notification is sent to the responsible SO back-office system.</p> <p>Otherwise, the notification is sent to the related CCP back-office system.</p> <p>If the transaction is aborted, the responsible back-office system is also notified.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module Service Operator Terminal Main Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the entitlement Entitlement has been read and checked
Postconditions	
Linked Use Cases	



(Extended By)	
Linked Use Cases (Includes)	Handle stored-value payment method recharged notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Payment method directory entry : EntitlementDirectoryEntry Action payment parameters : SvpMChargingPaymentParameters Action amount (non-negative)
Outputs	
Error Cases	
Activity Diagram	T-Main::Perform stored-value payment method recharging and notify

16.2.3 Optional: Autoload stored-value payment method

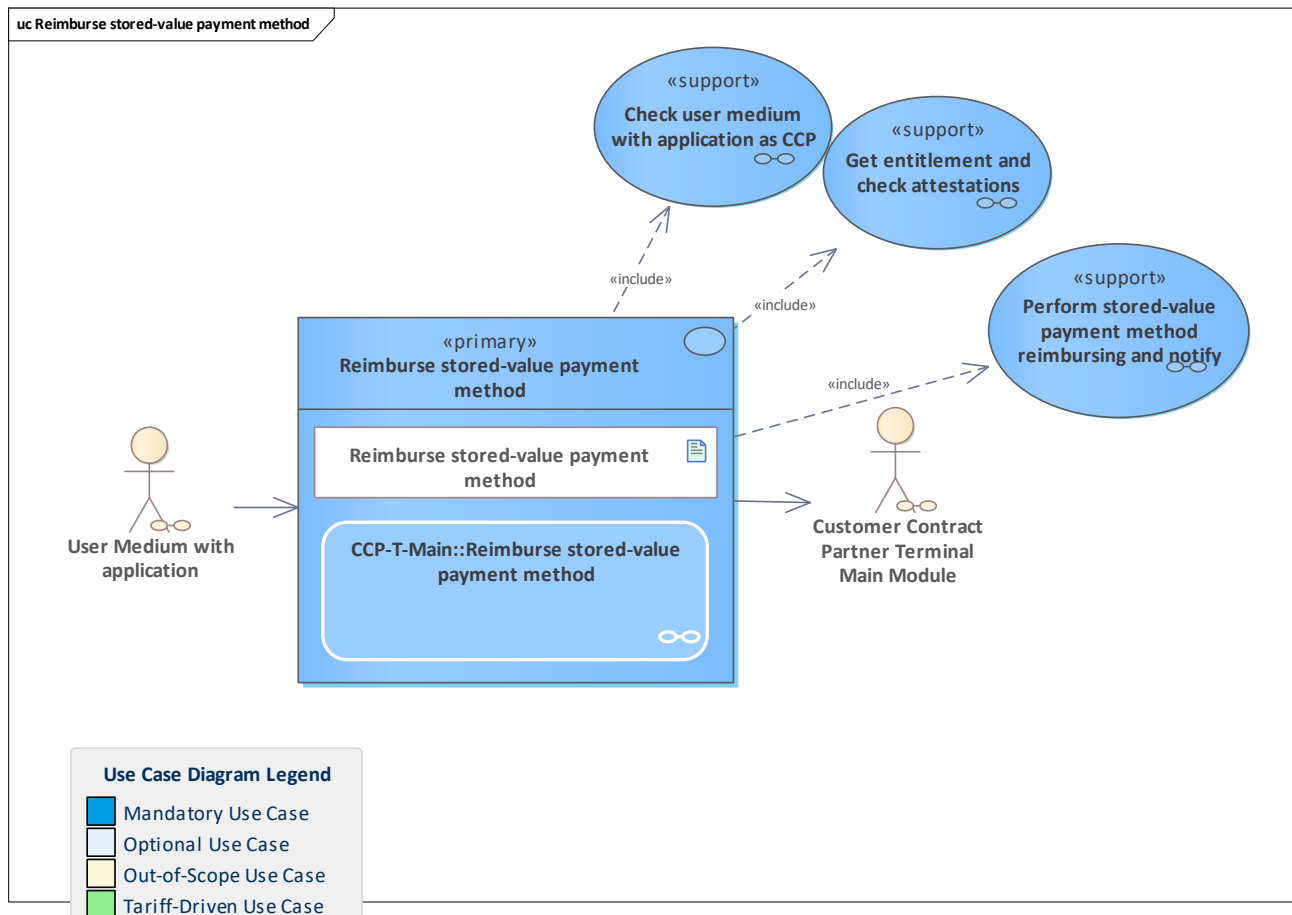


Use Case	<u>Autoload stored-value payment method</u>
Description	The terminal checks if sufficient stored-value funds are available and, if necessary, automatically triggers a recharge process if autoload was contractually agreed upon. In a CICO environment, the autoload option can also be available in SO terminals. In this case, the terminal must have a SAM with sufficient rights.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u> <u>Service Operator Terminal Main Module</u>
Preconditions	<u>SAM is in state ACTIVATED and contains product issuance rights for the entitlement</u> <u>Entitlement has been read and checked</u> <u>Entitlement has been read and checked</u>
Postconditions	
Linked Use Cases (Extended By)	<u>Perform stored-value payment method recharging and notify</u>
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	



Base Activity	
Inputs	Stored-value payment method : Entitlement Payment method directory entry : EntitlementDirectoryEntry Action amount
Outputs	
Error Cases	Insufficient funds
Activity Diagram	T-Main::Assert sufficient stored-value balance considering autoload

16.2.4 Reimburse stored-value payment method

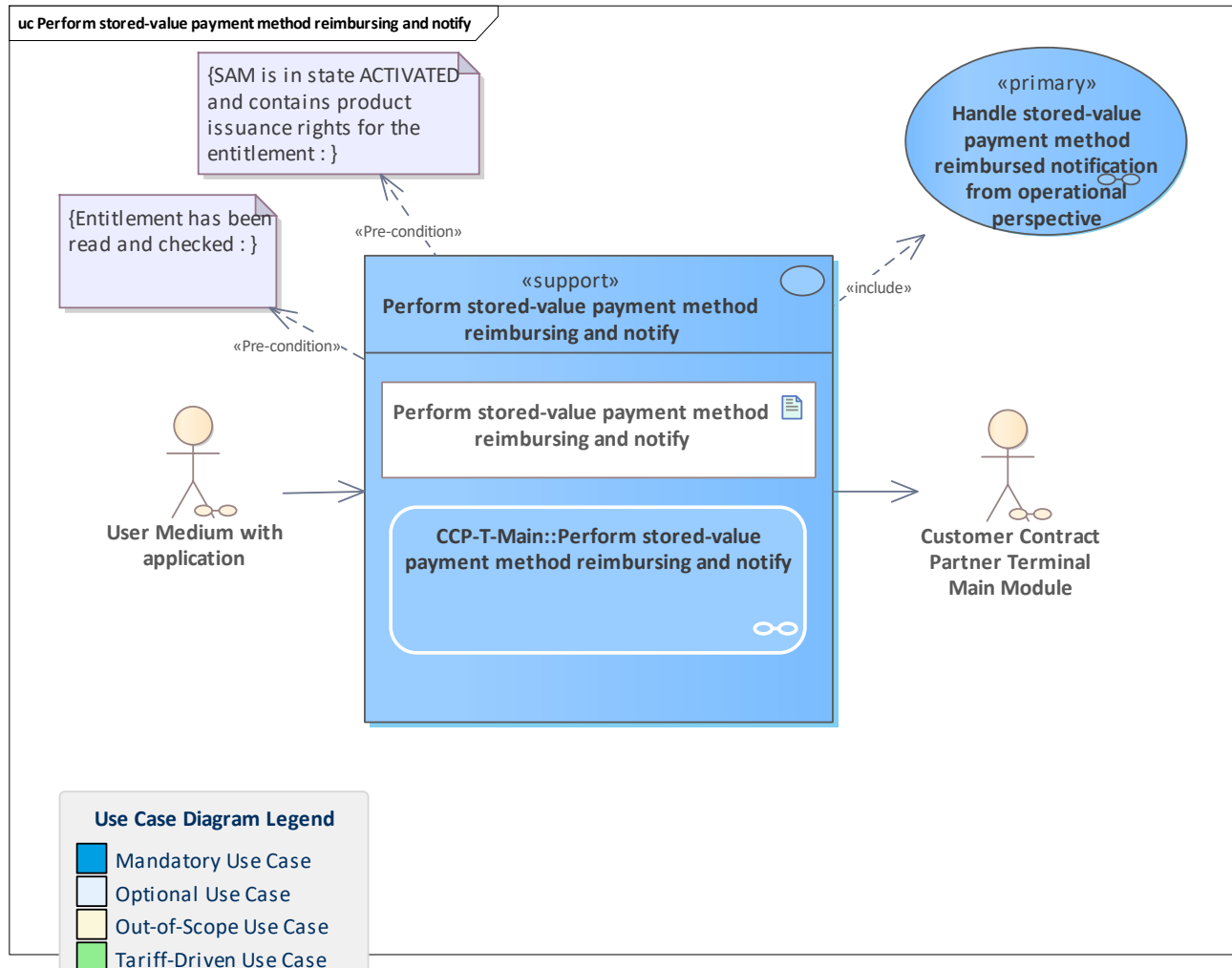


Use Case	Reimburse stored-value payment method
Description	Reimburse stored-values from a stored-value payment method. Performed by the CCP terminal. This process performs the reimbursement (if the current balance of the stored-value payment method is > 0), including a transaction and the triggering of the notification chain. Note: the stored-value payment method is not terminated.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform stored-value payment method reimbursing and notify / Check user medium with application as CCP / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	



Error Cases	
Activity Diagram	<u>CCP-T-Main::Reimburse stored-value payment method</u>

16.2.5 Perform stored-value payment method reimbursing and notify

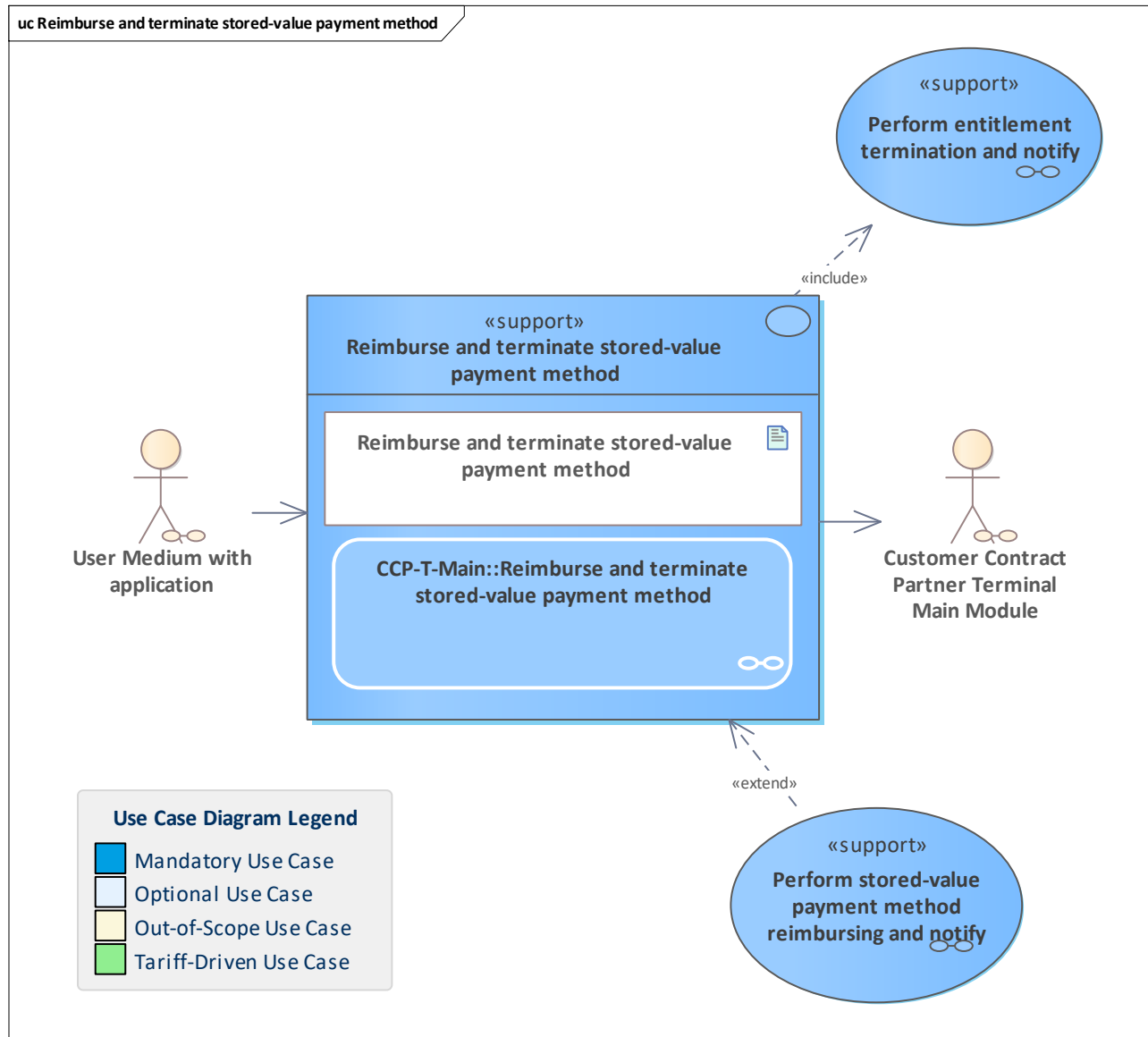


Use Case	Perform stored-value payment method reimbursing and notify
Description	The CCP terminal performs stored-value payment method reimbursement and notifies the responsible CCP back-office system. If the transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the entitlement Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle stored-value payment method reimbursed notification from operational perspective
Linked Use Cases (Realises)	



Base Activity	
Inputs	Payment method directory entry : EntitlementDirectoryEntry Action payment parameters : SvpMChargingPaymentParameters Action amount (non-positive)
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform stored-value payment method reimbursing and notify

16.2.6 Reimburse and terminate stored-value payment method

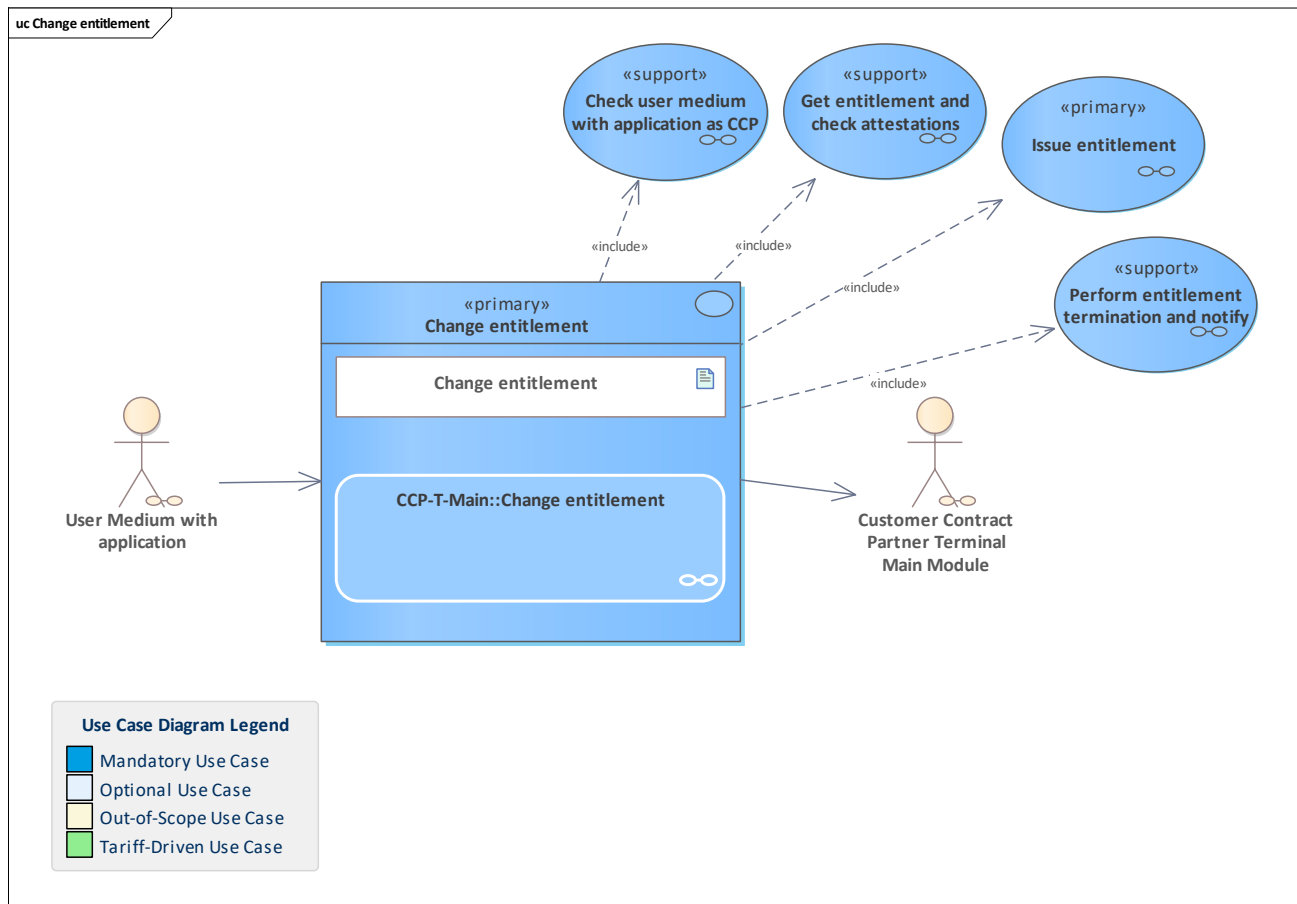


Use Case	<u>Reimburse and terminate stored-value payment method</u>
Description	<p>Reimburse and terminate a stored-value payment method as a supporting use case. The primary use case is Take back entitlement.</p> <p>Performed by the CCP terminal.</p> <p>This process is divided into two parts:</p> <ul style="list-style-type: none"> the potential reimbursement (if the current balance of the stored-value payment method is > 0), including a transaction and the triggering of the notification chain the termination transaction of the stored-value payment method, including the triggering of the notification chain
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>



Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Perform stored-value payment method reimbursing and notify
Linked Use Cases (Includes)	Perform entitlement termination and notify
Linked Use Cases (Realises)	
Base Activity	
Inputs	Stored-value payment method entry : EntitlementDirectoryEntry Stored-value payment method : Entitlement
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Reimburse and terminate stored-value payment method

16.2.7 Change entitlement

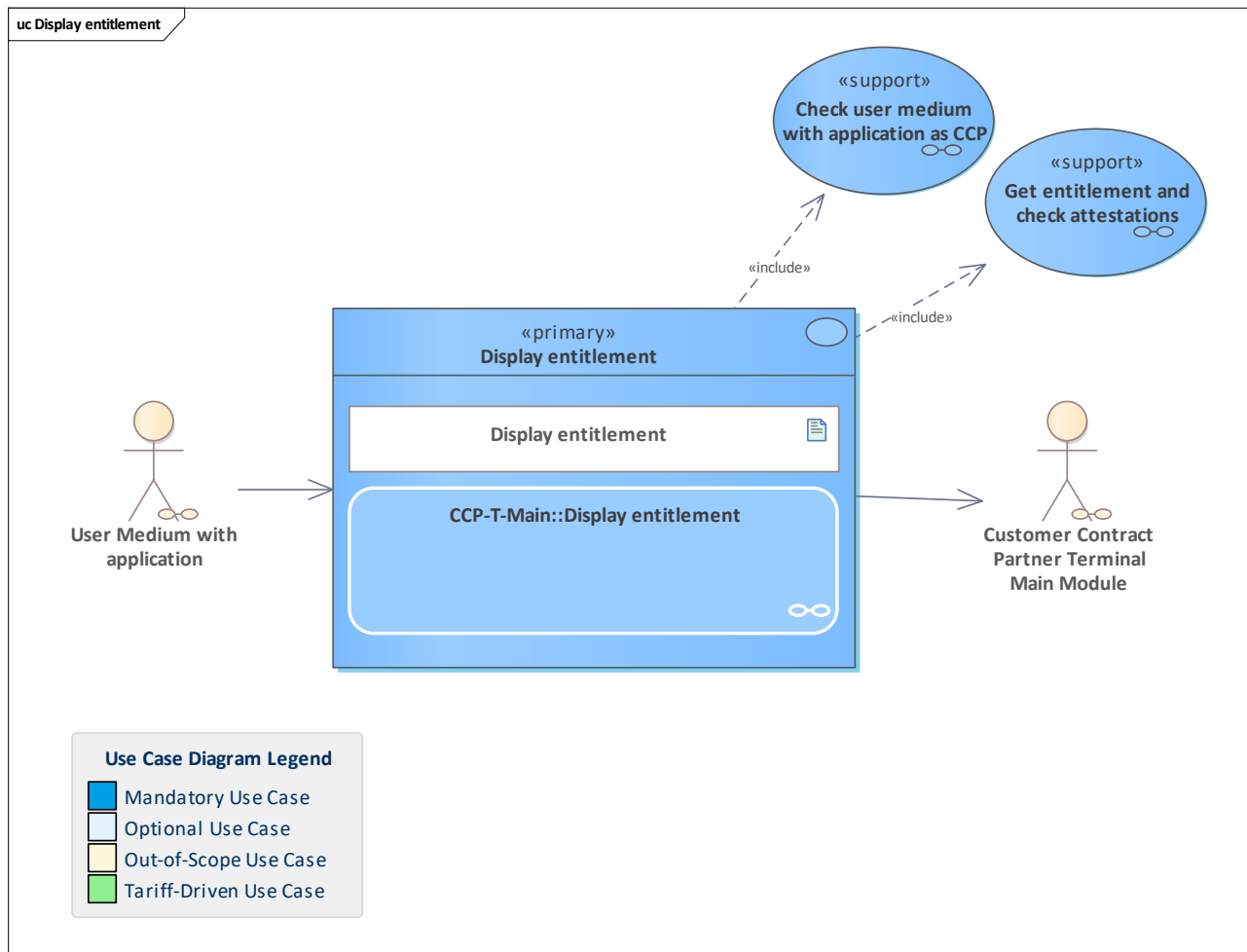


Use Case	Change entitlement
Description	<p>An entitlement needs to be replaced with a new one, for example, due to changed product parameters.</p> <p>Please note that it is assumed that there is no need for any payment transaction.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Issue entitlement / Check user medium with application as CCP / Perform entitlement termination and notify / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	



Error Cases	
Activity Diagram	<u>CCP-T-Main::Change entitlement</u>

16.2.8 Display entitlement



Use Case	Display entitlement
Description	Display an entitlement on a user medium with an application, independent of its status and validity period.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	



Activity Diagram	CCP-T-Main::Display_entitlement
-------------------------	---



17 Sale Electronic Ticket via Account-Based Payment Bundle CCP-Terminal

Functionality bundle that provides CCP terminal use cases for selling or purchasing an electronic ticket to added to the user medium using the account-based payment method, which is also located on the user medium.

17.1 Overview

Sell electronic ticket using account-based payment method

Optional: Sell static entitlement using account-based payment method

Debit account-based payment method

Perform account-based payment method debiting and notify

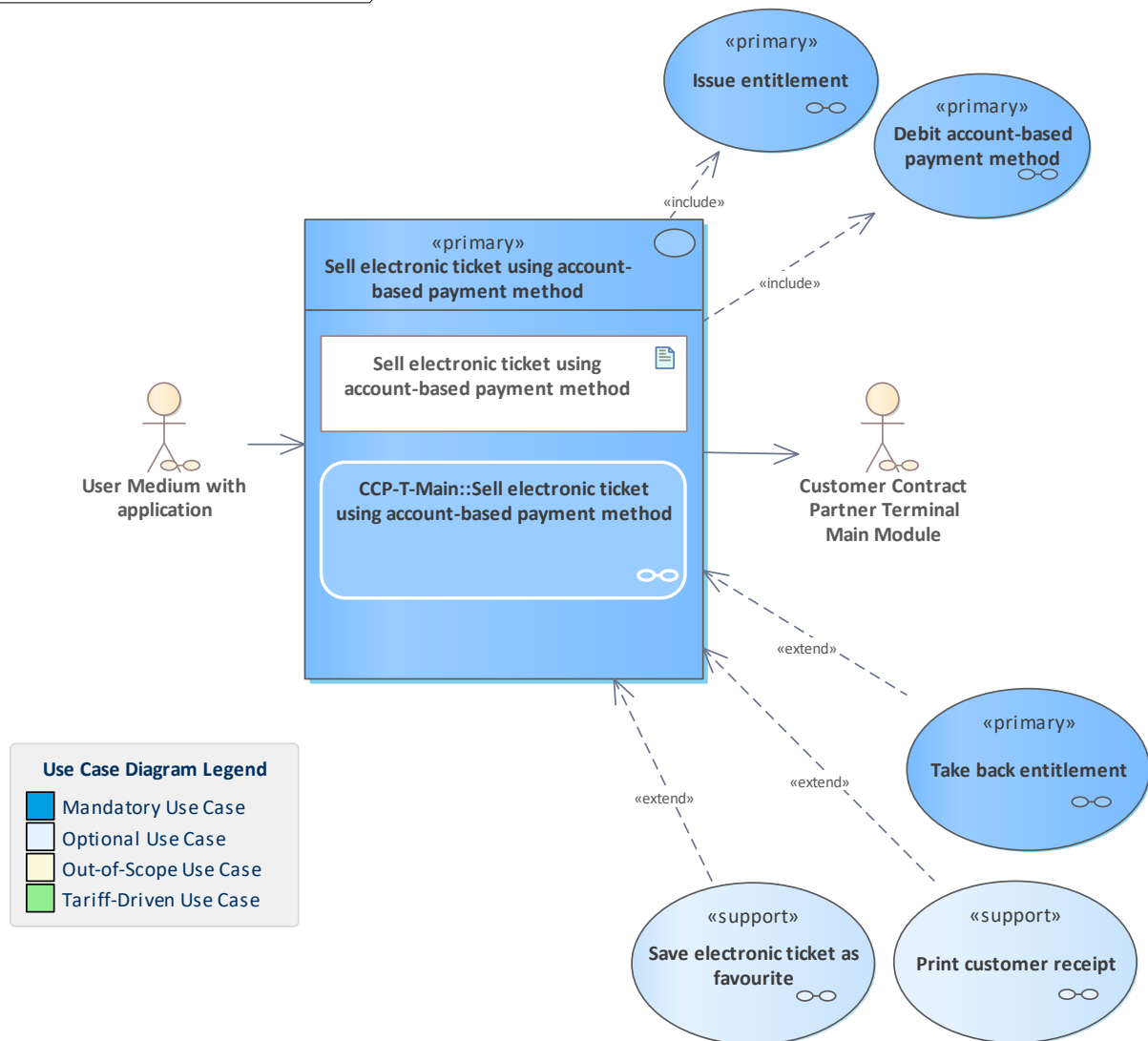
Credit account-based payment method

Perform account-based payment method crediting and notify

17.2 Use Cases

17.2.1 Sell electronic ticket using account-based payment method

uc Sell electronic ticket using account-based payment method

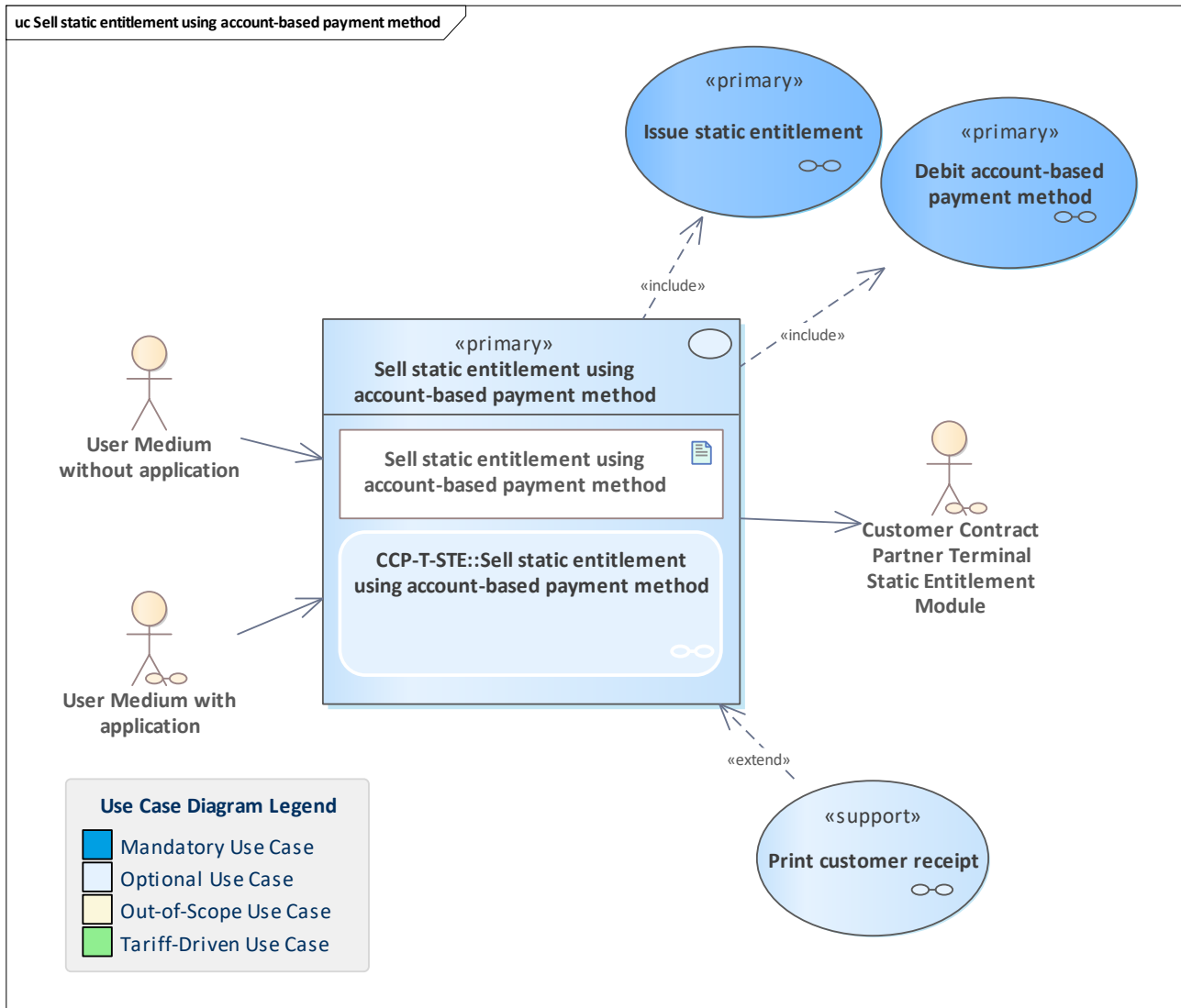


Use Case	<u>Sell electronic ticket using account-based payment method</u>
Description	An electronic ticket is sold and issued to a user medium. Payment is done using an account-based payment method already present on the same user medium. If there is not enough space on the target user medium to issue the electronic ticket, entitlements might be deleted and/or terminated after user/staff confirmation.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>SAM is in state ACTIVATED and contains product issuance rights for the entitlement</u>
Postconditions	
Linked Use Cases (Extended By)	<u>Take back entitlement</u> / <u>Take back entitlement</u> / <u>Print customer receipt</u> / <u>Save electronic ticket as favourite</u> / <u>Save electronic ticket as favourite</u> / <u>Print customer receipt</u>
Linked Use Cases (Includes)	<u>Issue entitlement</u> / <u>Debit account-based payment method</u>
Linked Use Cases (Realises)	



Base Activity	
Inputs	CCP organisation ID : OrganisationId Entitlement effective time : EntitlementEffectiveTime Entitlement expiration time : EntitlementExpirationTime Infotext : Infotext Product billing type : AccountingProcedureTypeCode Product ID : ProductId Product parameters : ProductParameters Replaced entitlement ID : EntitlementId Ticket price VAT rate (in 1/100 percent)
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Sell electronic ticket using account-based payment method

17.2.2 Optional: Sell static entitlement using account-based payment method

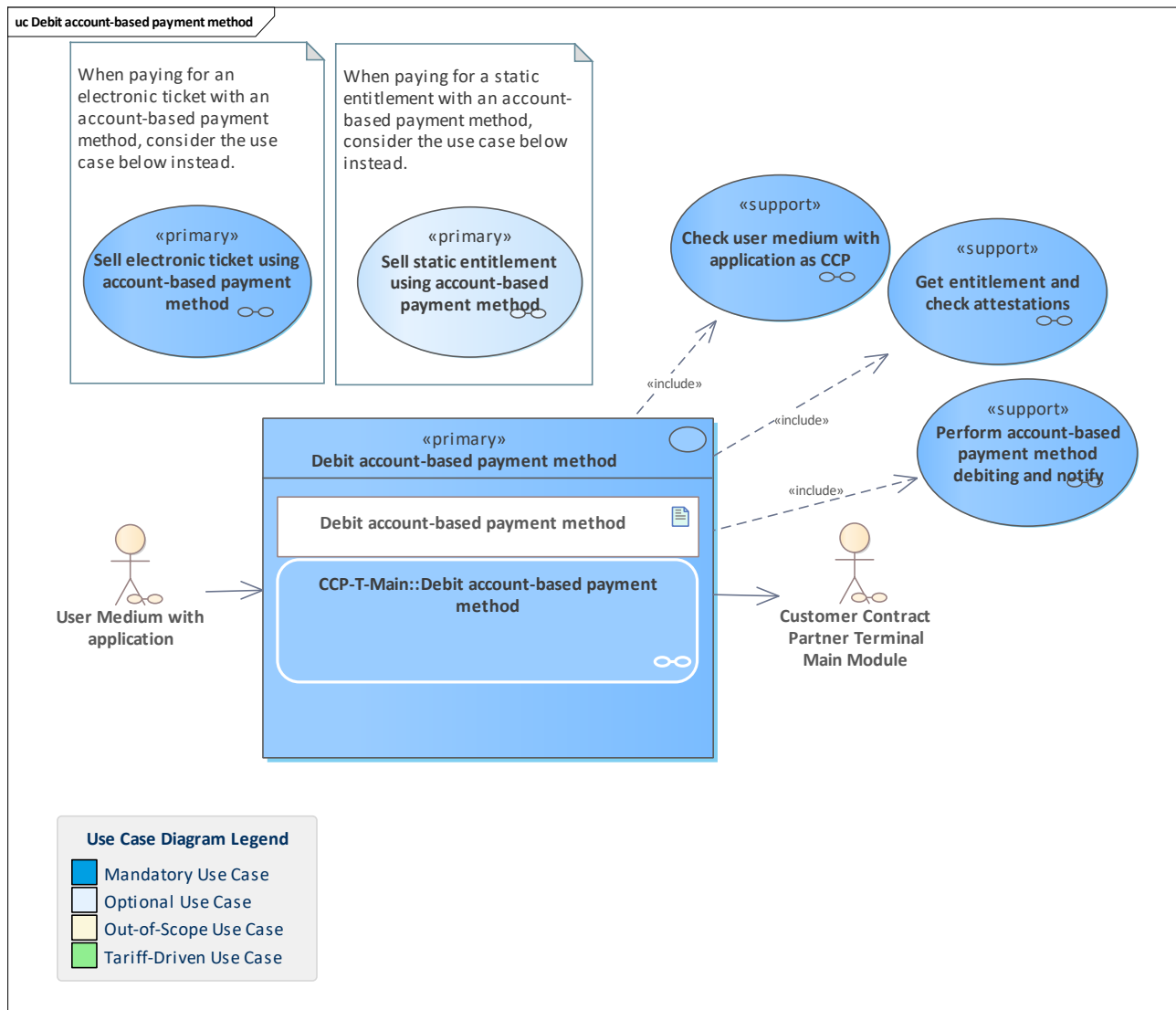


Use Case	<u>Sell static entitlement using account-based payment method</u>
Description	<p>Sell a static entitlement using an account-based payment method. Since this process involves both static and non-static entitlements, error scenarios and error handling mechanisms for both could apply. Thus, we provide additional steps to prevent error handling processes in the back-office systems where possible. For that, we exploit the fact that we can keep the already issued static entitlement from the customer until the payment succeeds. Selling multiple static entitlements in a single process can be implemented in two ways, which both involve one debiting action per issued static entitlement:</p> <ol style="list-style-type: none"> 1. Perform the issuance and debiting process steps (including notifying the back-office systems) multiple times and perform the delivery step only once for a StaticEntitlements structure containing the 1..* StaticEntitlementData for each issued static



	<p>entitlement.</p> <p>2. Perform the whole process multiple times. In the target system, e.g. the customer smartphone, the static entitlements delivered in multiple steps can then be combined into a single StaticEntitlements structure so that it can be inspected efficiently. To do so, take the StaticEntitlementData from all static entitlements and combine them with a single SAM certificate to form a single StaticEntitlements structure. Of course, the issuing SAM needs to be identical for all entitlements for this to work.</p>
Initiating Actor	User Medium without application User Medium with application
Reacting Actor	Customer Contract Partner Terminal Static Entitlement Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Autoload stored-value payment method / Print customer receipt / Print customer receipt
Linked Use Cases (Includes)	Issue static entitlement / Issue static entitlement / Debit account-based payment method
Linked Use Cases (Realises)	
Base Activity	
Inputs	CCP organisation ID : OrganisationId Entitlement effective time : EntitlementEffectiveTime Entitlement expiration time : EntitlementExpirationTime Infotext : Infotext Product ID : ProductId Product parameters : ProductParameters Replaced entitlement ID : EntitlementId SCE ID : AppInstanceId Ticket price VAT rate (in 1/100 percent)
Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Sell static entitlement using account-based payment method

17.2.3 Debit account-based payment method

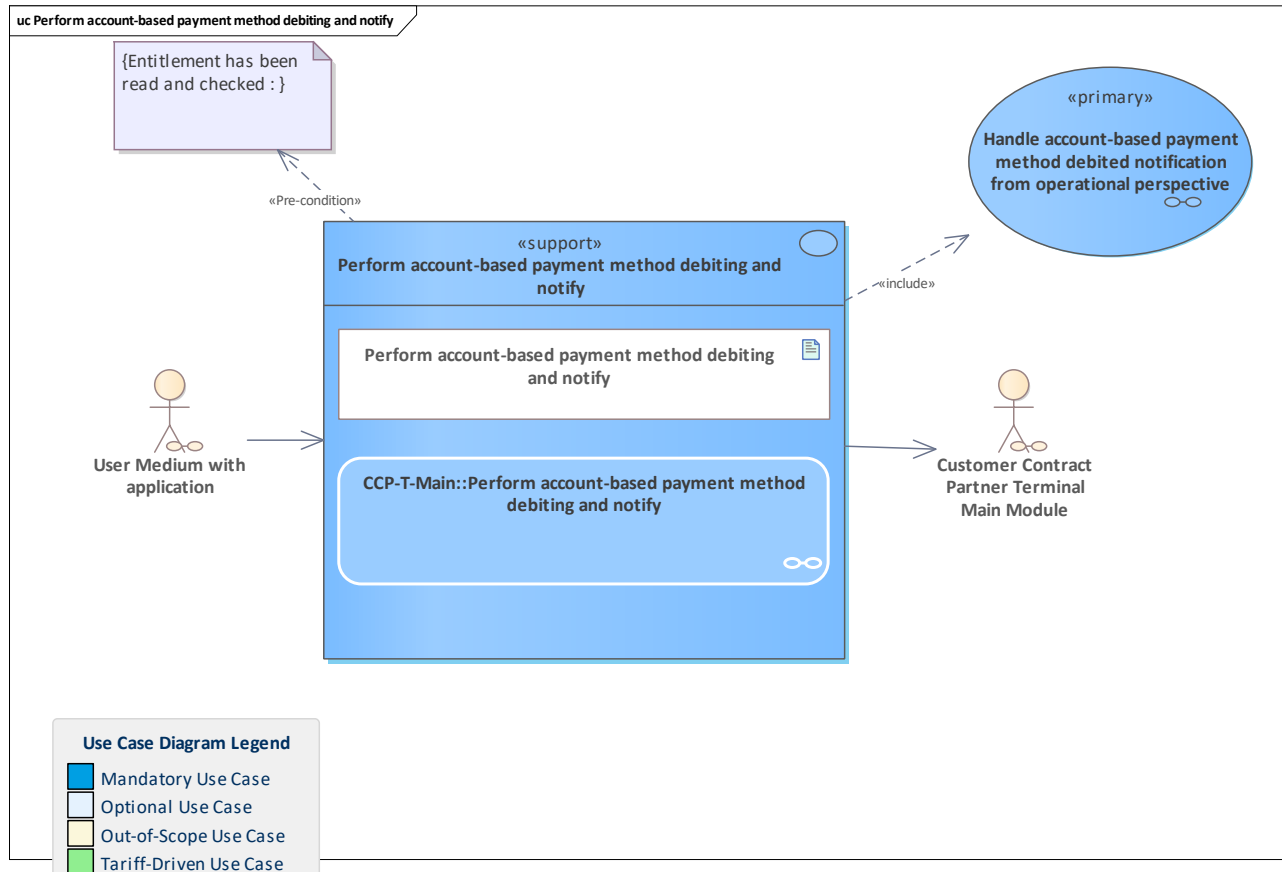


Use Case	<u>Debit account-based payment method</u>
Description	Debit an account-based payment method by a CCP terminal. The use case initially checks the account-based payment method. Finally, the debit is performed and notified to the CCP back-office system (same CCP as the terminal operator of the current terminal). Normally, this use case is combined with a purchase transaction of an electronic ticket.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	<u>Perform account-based payment method debiting and notify</u> / <u>Debit account-based payment method</u> / <u>Check user medium with application as CCP</u> / <u>Get entitlement and check attestations</u> / <u>Debit</u>



	account-based payment method
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action debiting parameters : CreditDebitPaymentParameters Debiting amount incl VAT VAT rate (in 1/100 percent)
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Debit account-based payment method

17.2.4 Perform account-based payment method debiting and notify

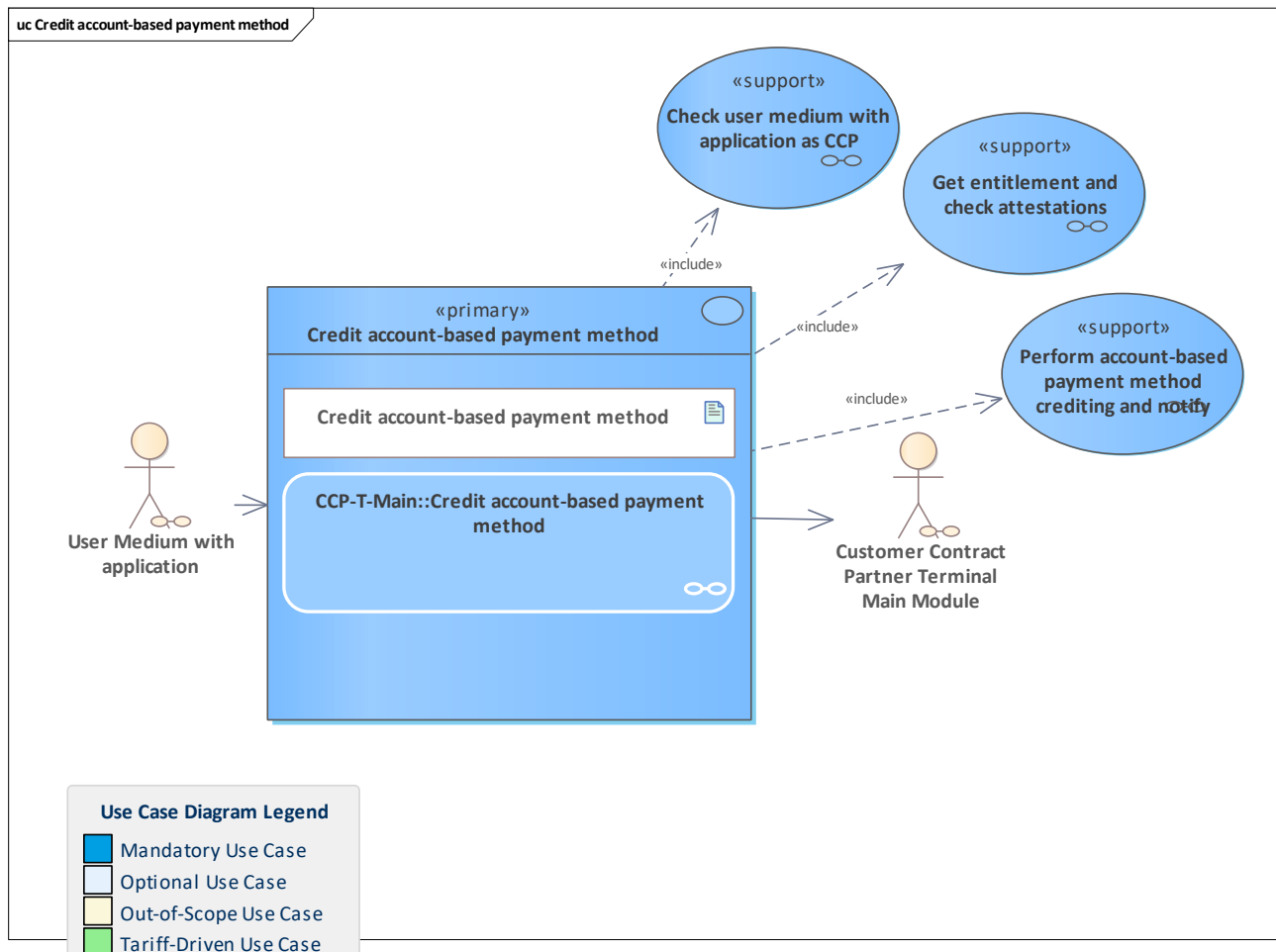


Use Case	Perform account-based payment method debiting and notify
Description	The CCP terminal performs a transaction to debit an account-based payment method on a user medium with an application and notifies the CCP back-office system about the debiting transaction. If a transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle account-based payment method debited notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action amount (non-positive) Action payment parameters : CreditDebitPaymentParameters Entitlement directory entry : EntitlementDirectoryEntry VAT rate (in 1/100 percent)
Outputs	



Error Cases	
Activity Diagram	CCP-T-Main::Perform account-based payment method debiting and notify

17.2.5 Credit account-based payment method

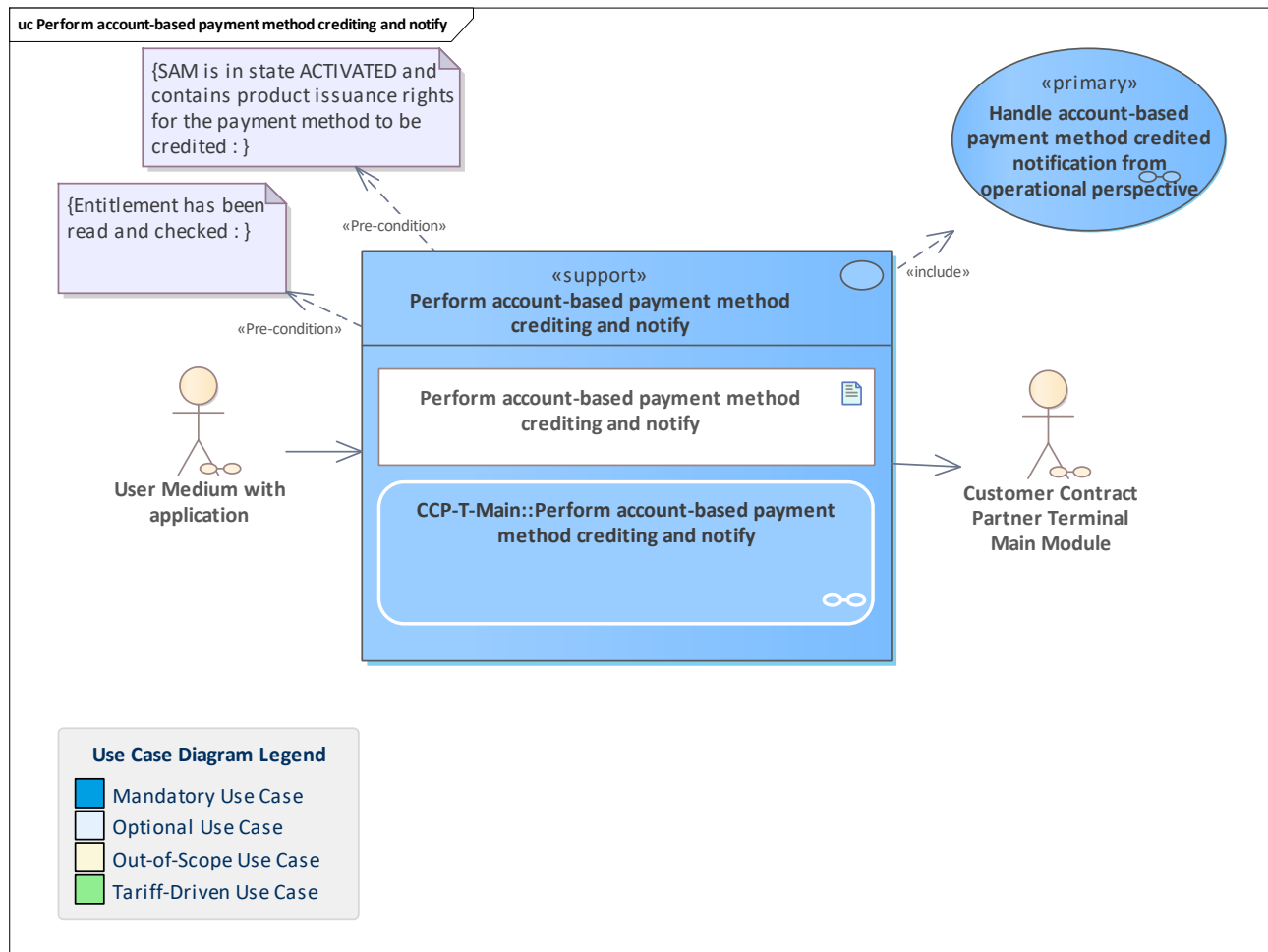


Use Case	Credit account-based payment method
Description	Credit an account-based payment method by a CCP terminal. The use case initially checks the account-based payment method. Finally, the credit is performed and notified to the CCP back-office system (same CCP as the terminal operator of the current terminal). Normally, this use case is combined with a reimbursement action of an electronic ticket.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform account-based payment method crediting and notify / Get entitlement and check attestations / Check user medium with application as CCP
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action payment parameters : CreditDebitPaymentParameters



	<u>VAT rate (in 1/100 percent)</u> <u>Crediting amount incl VAT (non-negative)</u>
Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Credit account-based payment method</u>

17.2.6 Perform account-based payment method crediting and notify



Use Case	Perform account-based payment method crediting and notify
Description	The CCP terminal performs a transaction to credit an account-based payment method on a user medium with an application and notifies the CCP back-office system about the crediting transaction. If a transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the payment method to be credited Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle account-based payment method credited notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	



Inputs	<u>Crediting amount (non-negative)</u> <u>Action payment parameters : CreditDebitPaymentParameters</u> <u>Entitlement directory entry : EntitlementDirectoryEntry</u> <u>VAT rate (in 1/100 percent)</u>
Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-Main::Perform account-based payment method crediting and notify</u>



18 Sale Electronic Ticket via Stored-Value Payment Bundle CCP-Terminal

Functionality bundle that provides CCP terminal use cases for selling or purchasing an electronic ticket to added to the user medium using the stored-value payment method, which is also located on the user medium.

18.1 Overview

Sell electronic ticket using stored-value payment method

Optional: Sell static entitlement using stored-value payment method

Debit stored-value payment method

Perform stored-value payment method debiting and notify

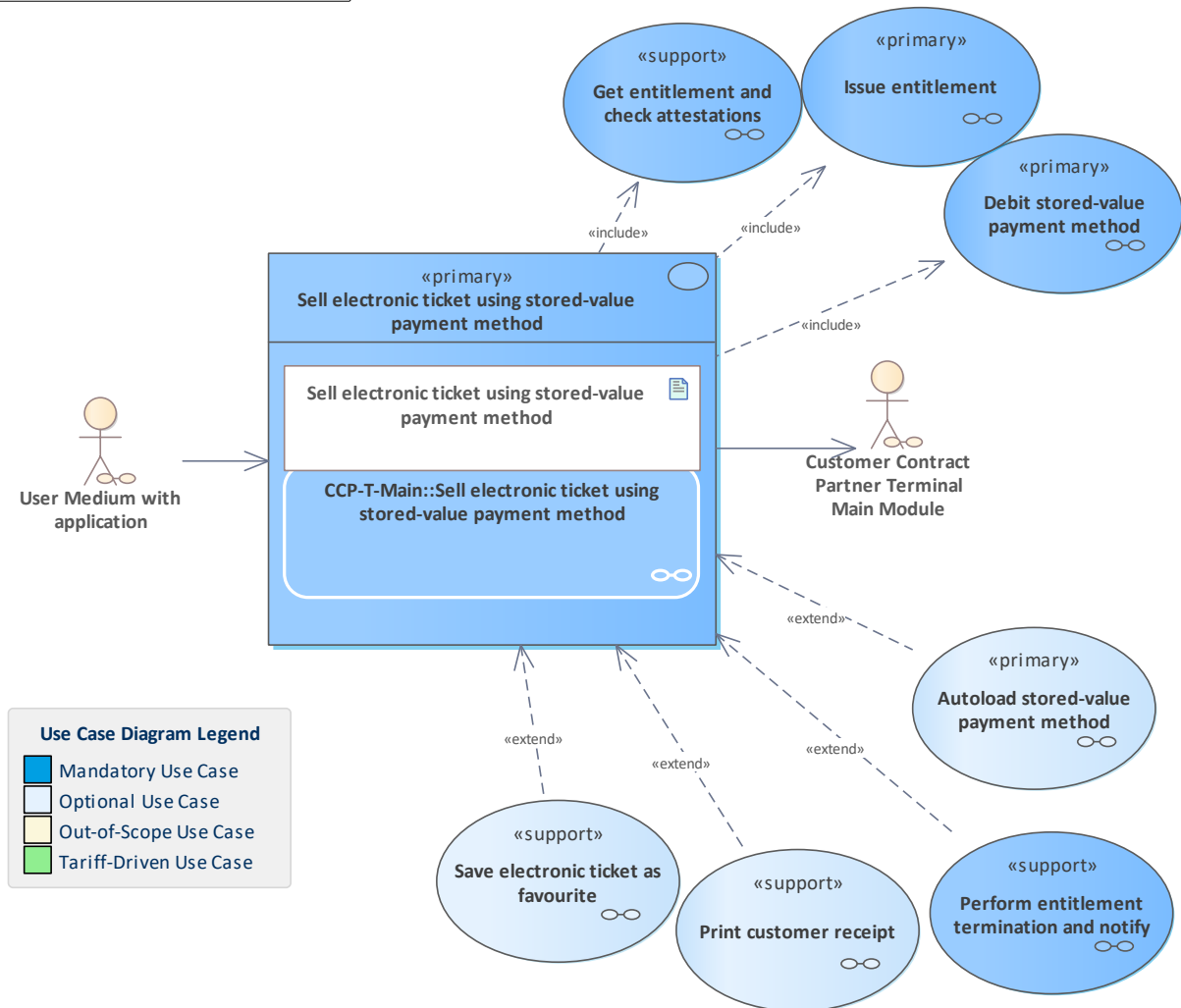
Credit stored-value payment method

Perform stored-value payment method crediting and notify

18.2 Use Cases

18.2.1 Sell electronic ticket using stored-value payment method

uc Sell electronic ticket using stored-value payment method

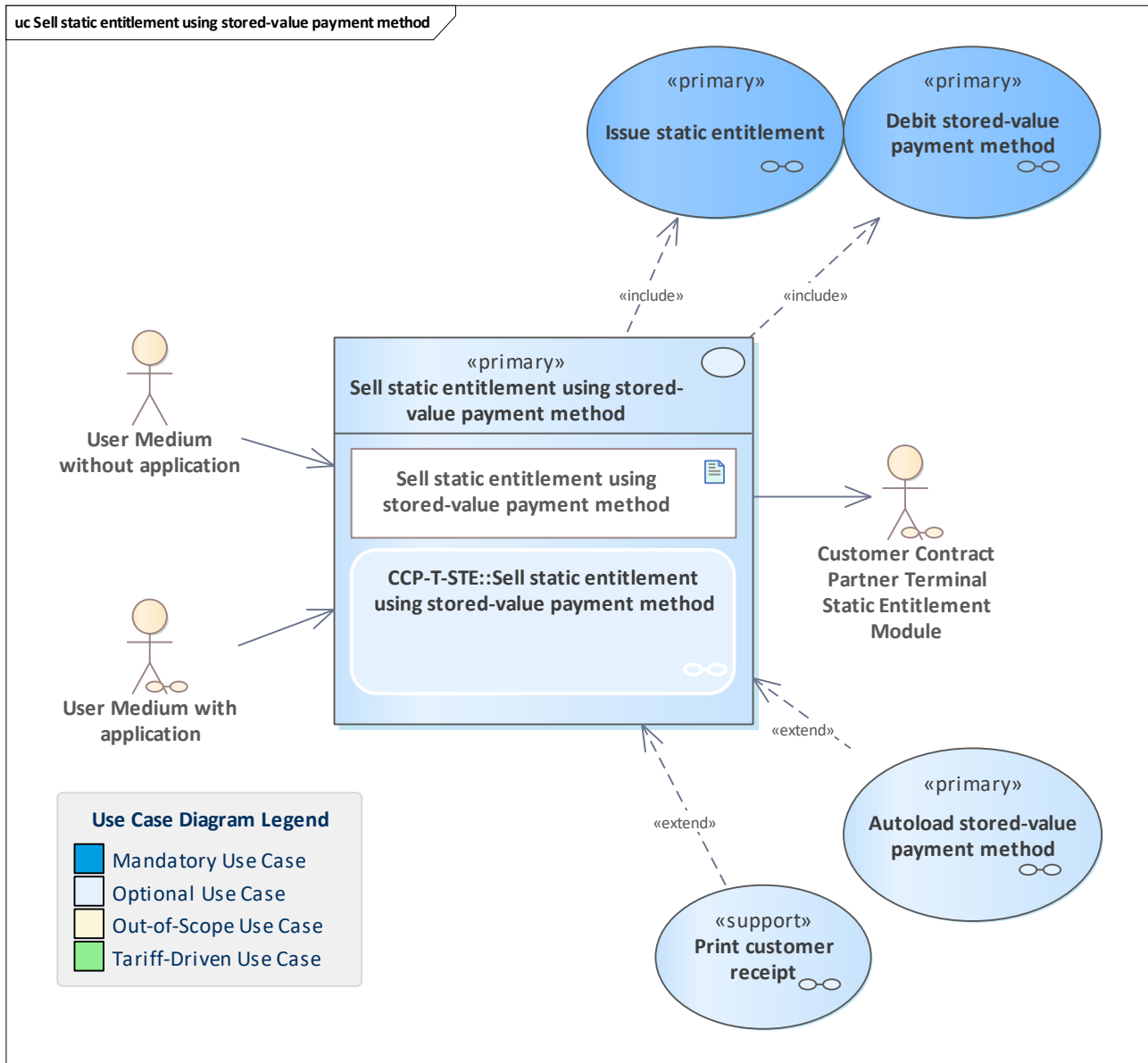


Use Case	Sell electronic ticket using stored-value payment method
Description	<p>An electronic ticket is sold and issued to a user medium. Payment is done using a stored-value payment method already present on the same user medium. To do so, the payment method balance is checked.</p> <p>If the balance does not suffice, the customer can recharge the payment method to be able to complete the sales process. If there is not enough space on the target user medium to issue the electronic ticket, entitlements might be deleted and/or terminated after user/staff confirmation.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the entitlement
Postconditions	
Linked Use Cases (Extended By)	Autoload stored-value payment method / Autoload stored-value payment method / Perform entitlement termination and notify / Print customer receipt / Save electronic ticket as favourite / Print customer receipt / Save electronic ticket as favourite
Linked Use Cases	Get entitlement and check attestations / Get entitlement and check



(Includes)	attestations / Issue entitlement / Debit stored-value payment method
Linked Use Cases (Realises)	
Base Activity	
Inputs	CCP organisation ID : OrganisationId Entitlement effective time : EntitlementEffectiveTime Entitlement expiration time : EntitlementExpirationTime Infotext : Infotext Product billing type : AccountingProcedureTypeCode Product ID : ProductId Product parameters : ProductParameters Replaced entitlement ID : EntitlementId Ticket price VAT rate (in 1/100 percent)
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Sell electronic ticket using stored-value payment method

18.2.2 Optional: Sell static entitlement using stored-value payment method

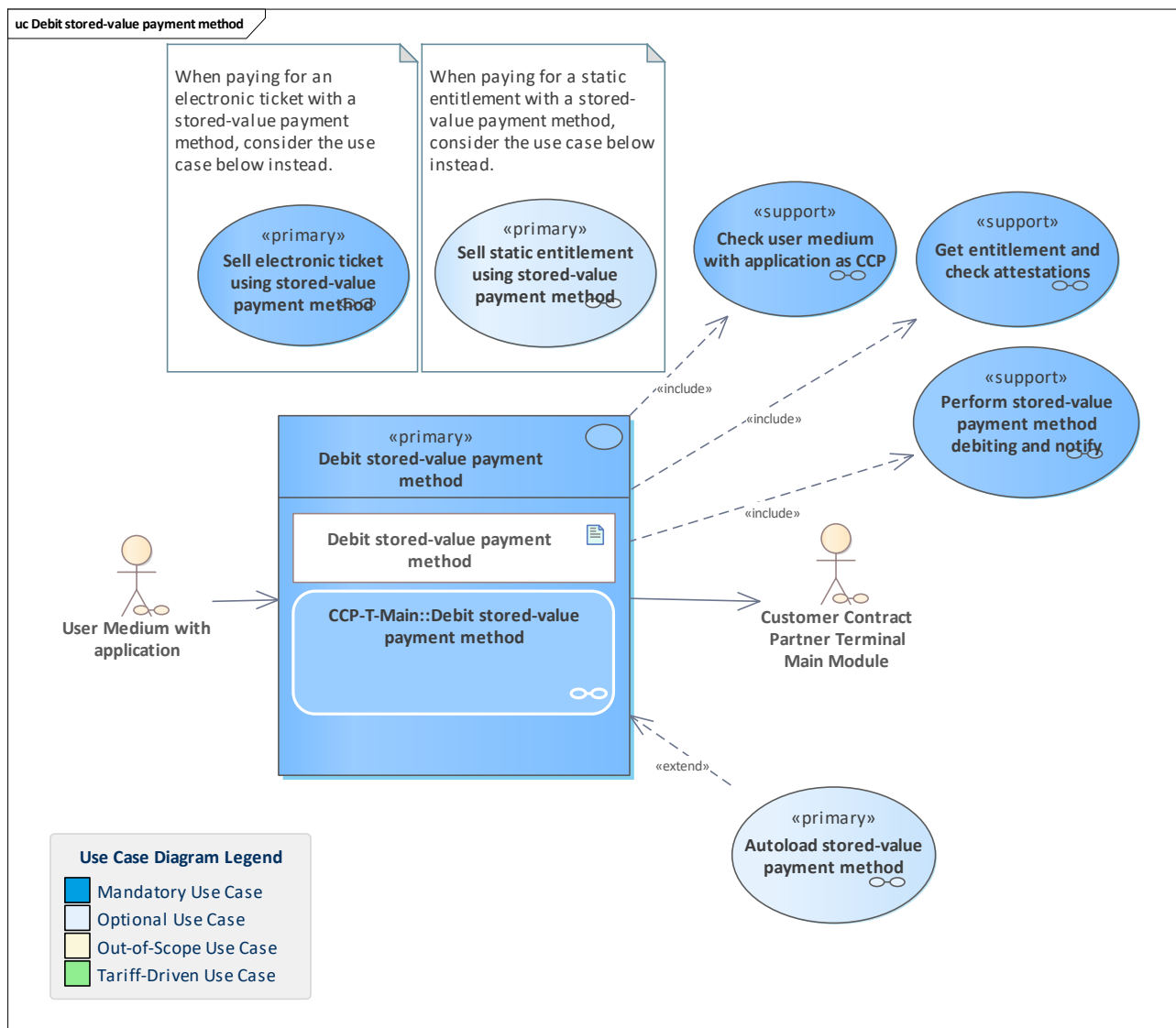


Use Case	Sell static entitlement using stored-value payment method
Description	<p>Sell a static entitlement using a stored-value payment method. Since this process involves both static and non-static entitlements, error scenarios and error handling mechanisms for both could apply. Thus, we provide additional steps to prevent error handling processes in the back-office systems where possible. For that, we exploit the fact that we can keep the already issued static entitlement from the customer until the payment succeeds. Selling multiple static entitlements in a single process can be implemented in two ways, which both involve one debiting action per issued static entitlement:</p> <ol style="list-style-type: none"> 1. Perform the issuance and debiting process steps (including



	<p>notifying the back-office systems) multiple times and perform the delivery step only once for a StaticEntitlements structure containing the 1..* StaticEntitlementData for each issued static entitlement.</p> <p>2. Perform the whole process multiple times. In the target system, e.g. the customer smartphone, the static entitlements delivered in multiple steps can then be combined into a single StaticEntitlements structure so that it can be inspected efficiently. To do so, take the StaticEntitlementData from all static entitlements and combine them with a single SAM certificate to form a single StaticEntitlements structure. Of course, the issuing SAM needs to be identical for all entitlements for this to work.</p>
Initiating Actor	User Medium without application User Medium with application
Reacting Actor	Customer Contract Partner Terminal Static Entitlement Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Autoload stored-value payment method / Autoload stored-value payment method / Print customer receipt / Print customer receipt
Linked Use Cases (Includes)	Issue static entitlement / Debit stored-value payment method
Linked Use Cases (Realises)	
Base Activity	
Inputs	CCP organisation ID : OrganisationId Entitlement effective time : EntitlementEffectiveTime Entitlement expiration time : EntitlementExpirationTime Infotext : Infotext Product ID : ProductId Product parameters : ProductParameters Replaced entitlement ID : EntitlementId SCE ID : AppInstanceId Ticket price VAT rate (in 1/100 percent)
Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Sell static entitlement using stored-value payment method

18.2.3 Debit stored-value payment method

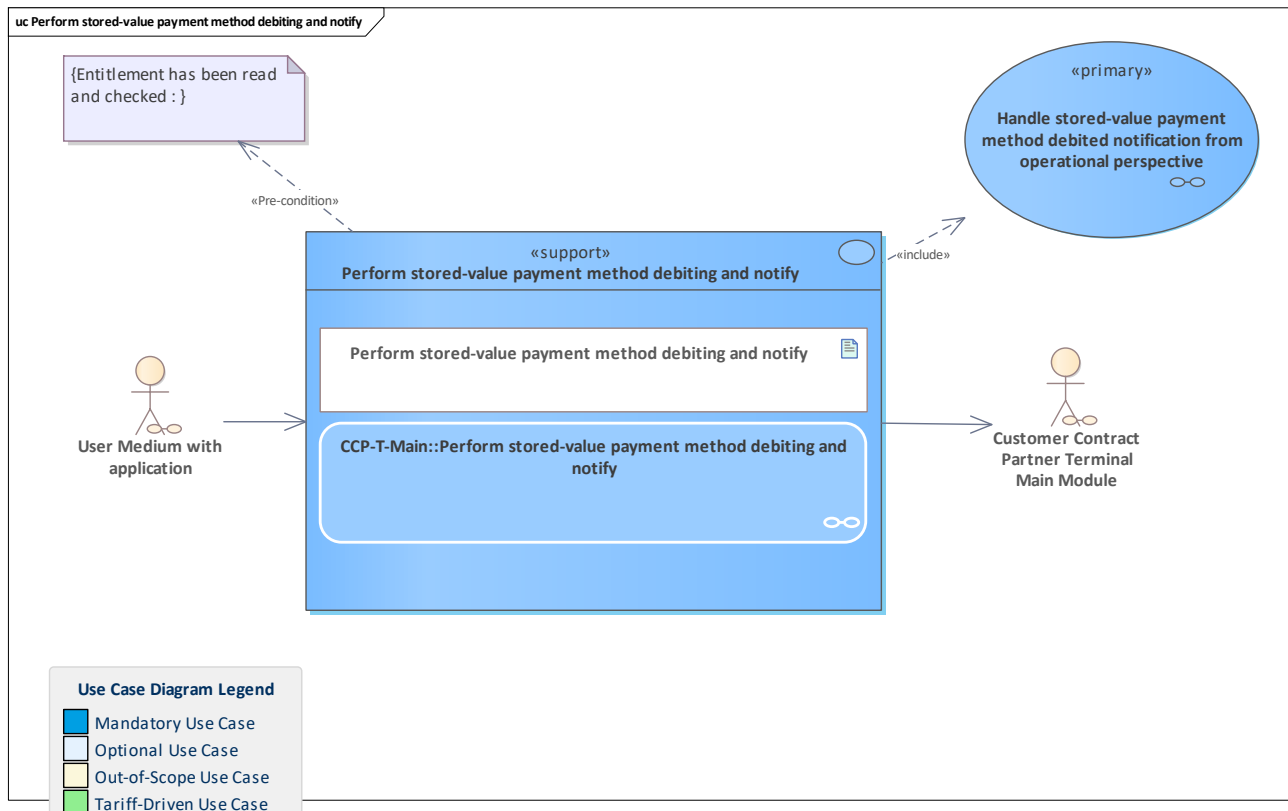


Use Case	Debit stored-value payment method
Description	<p>Debit a stored-value payment method by a CCP terminal. The use case checks the stored-value payment method and ensures, that the current balance in the stored-value account is sufficient for the intended debit action. Finally, the debit is performed and notified to the CCP back-office system (same CCP as the terminal operator of the current terminal).</p> <p>Normally, this use case is combined with a purchase transaction of an electronic ticket.</p> <p>If autoload is configured for the customer, this could come into play if the balance is not sufficient.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	



Linked Use Cases (Extended By)	Autoload stored-value payment method / Autoload stored-value payment method / Autoload stored-value payment method
Linked Use Cases (Includes)	Get entitlement and check attestations / Perform stored-value payment method debiting and notify / Check user medium with application as CCP / Get entitlement and check attestations / Debit stored-value payment method / Debit stored-value payment method
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action debiting parameters : CreditDebitPaymentParameters Debiting amount incl VAT VAT rate (in 1/100 percent)
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Debit stored-value payment method

18.2.4 Perform stored-value payment method debiting and notify

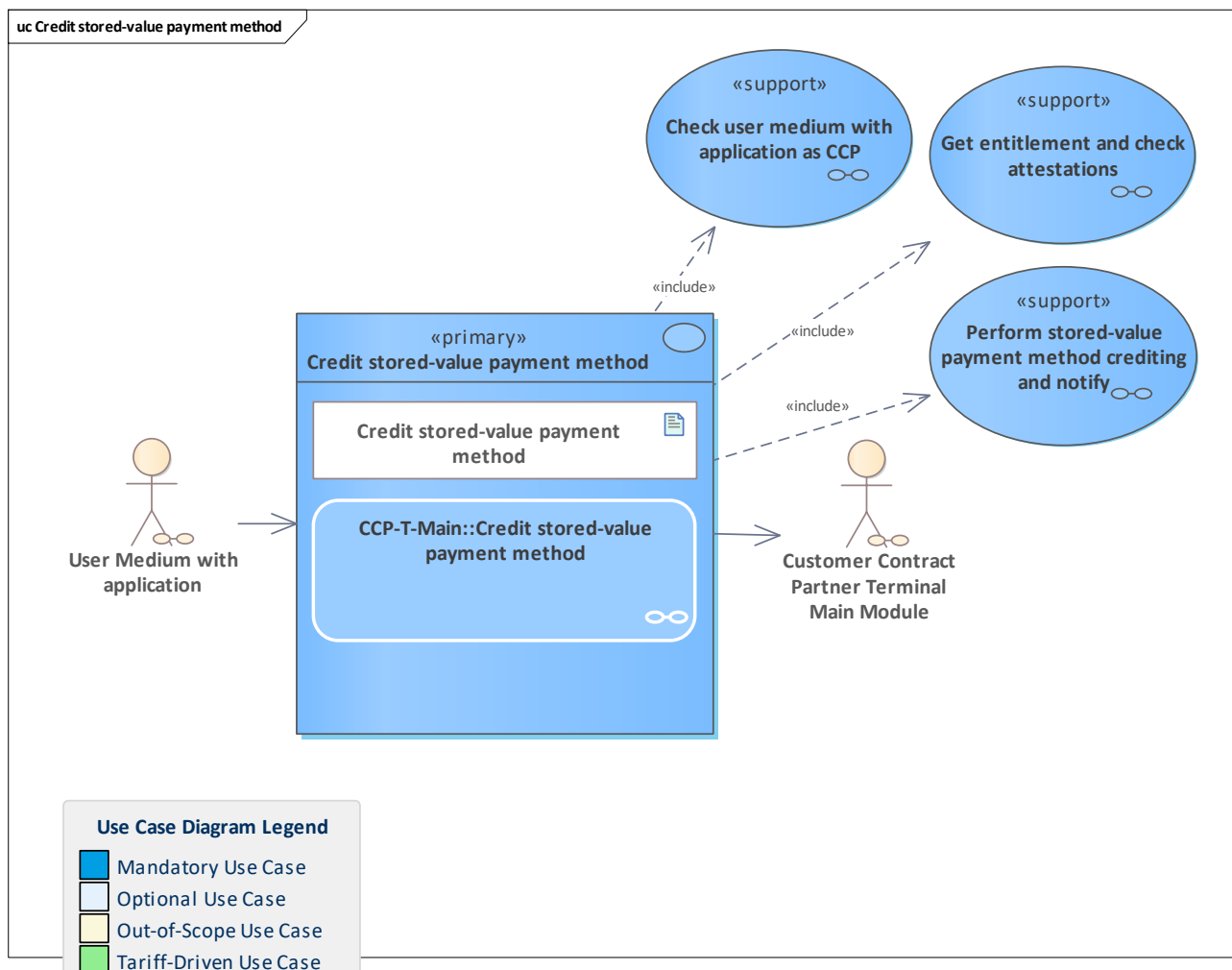


Use Case	Perform stored-value payment method debiting and notify
Description	The CCP terminal performs a transaction to debit a stored-value payment method on a user medium with an application and notifies the CCP back-office system about the debiting transaction. If the transaction is aborted, the CCP back-office system is also notified.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle stored-value payment method debited notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action payment parameters : CreditDebitPaymentParameters Entitlement directory entry : EntitlementDirectoryEntry VAT rate (in 1/100 percent) Action amount (non-positive)
Outputs	
Error Cases	



Activity Diagram	<u>CCP-T-Main::Perform stored-value payment method debiting and notify</u>
-------------------------	--

18.2.5 Credit stored-value payment method

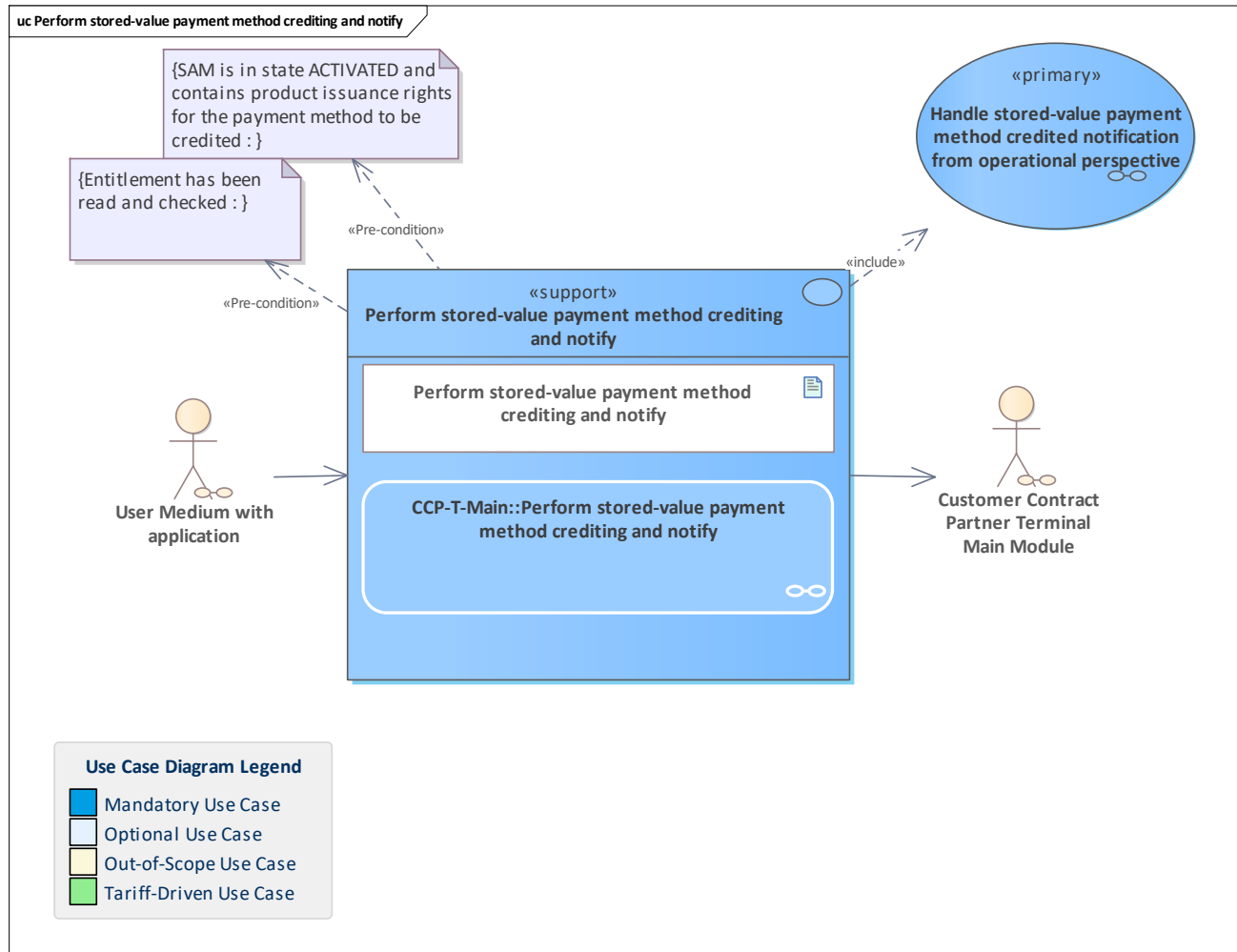


Use Case	Credit stored-value payment method
Description	Credit a stored-value payment method. Done by a CCP terminal e.g. as part of a reimbursement of a purchased electronic ticket. Check the maximum allowed balance and trigger the action and notification.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform stored-value payment method crediting and notify / Get entitlement and check attestations / Check user medium with application as CCP
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action payment parameters : CreditDebitPaymentParameters Crediting amount incl VAT (non-negative) VAT rate (in 1/100 percent)



Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Credit stored-value payment method

18.2.6 Perform stored-value payment method crediting and notify



Use Case	Perform stored-value payment method crediting and notify
Description	The CCP terminal performs the transaction to credit a stored-value payment method and notifies the responsible CCP back-office system about it. A potential transaction abortion is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the payment method to be credited Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle stored-value payment method credited notification from operational perspective
Linked Use Cases (Realises)	



Base Activity	
Inputs	Action payment parameters : CreditDebitPaymentParameters VAT rate (in 1/100 percent) Action amount (non-negative) Payment method directory entry : EntitlementDirectoryEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform stored-value payment method crediting and notify

19 IN-OUT Bundle CCP-Terminal

Functionality bundle that provides CCP terminal use cases for IN-OUT functionality. The term CICO is also used.

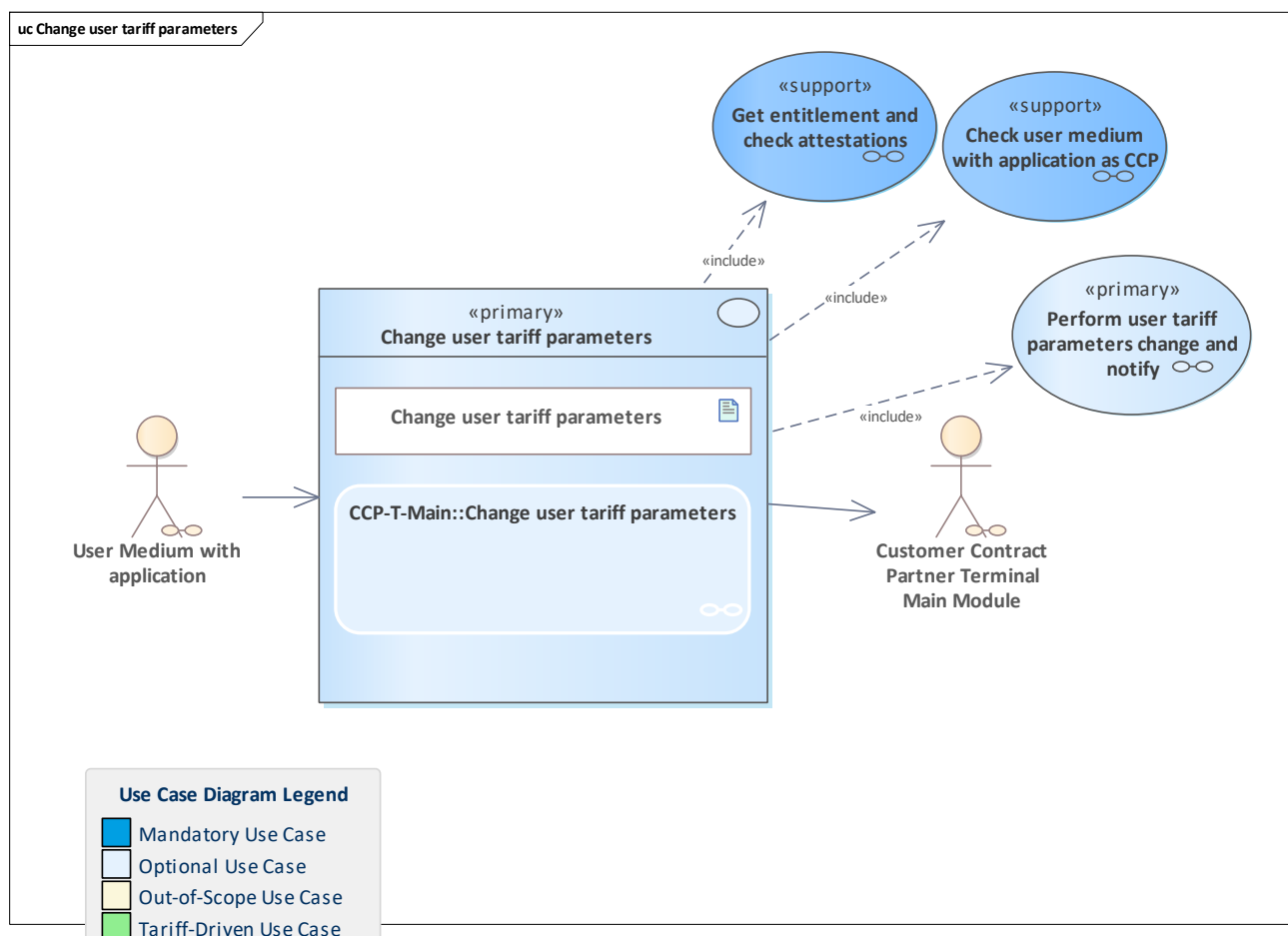
19.1 Overview

Optional: Change user tariff parameters

Optional: Perform user tariff parameters change and notify

19.2 Use Cases

19.2.1 Optional: Change user tariff parameters

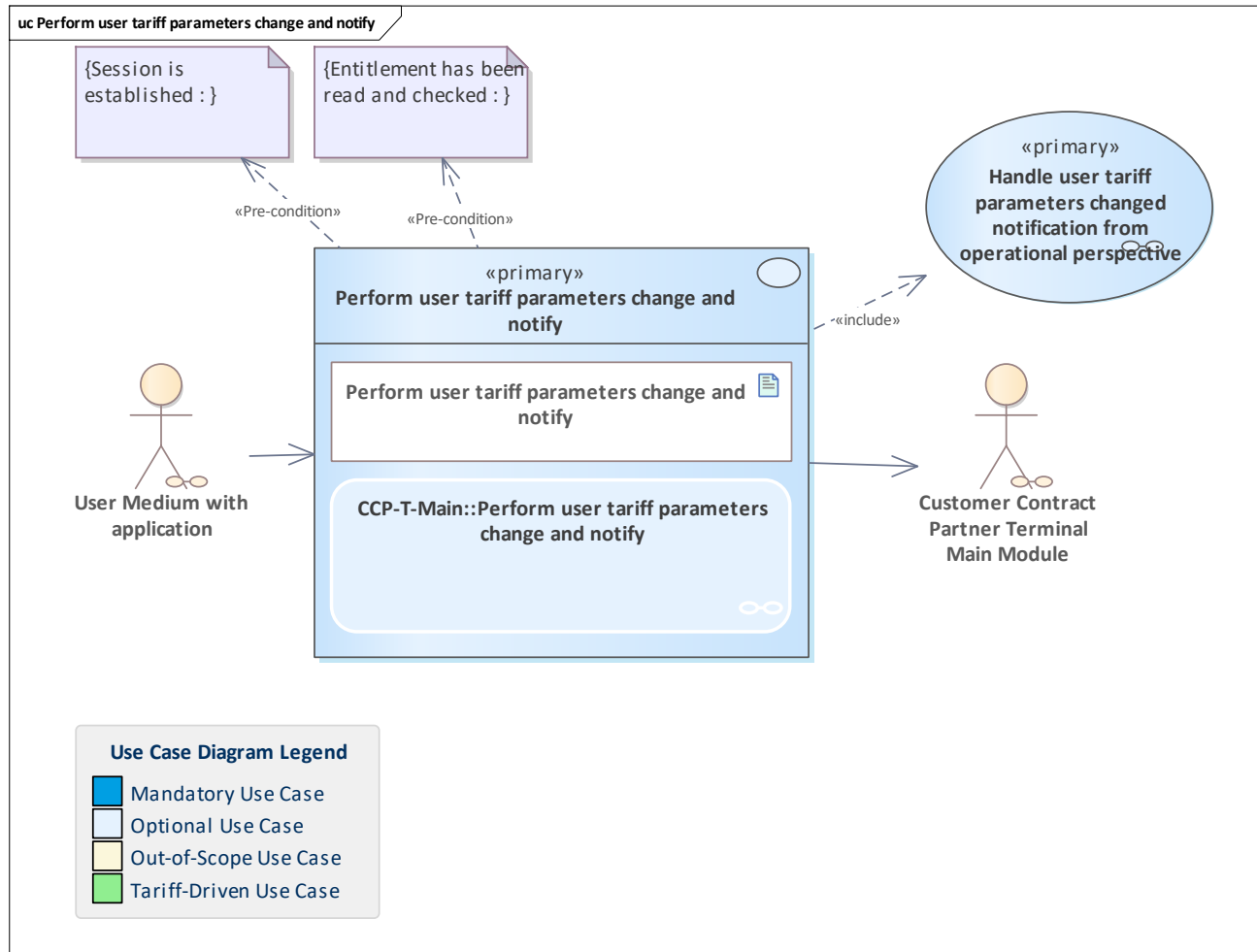


Use Case	Change user tariff parameters
Description	<p>A customer can change user tariff parameters (UTP) such as transport rules or service class for a certain entitlement. This use case is applicable only for the CICO process.</p> <p>Please note that, UTP should be changed before checking in. If UTP are allowed to be changed during the journey from a tariff</p>



	<p>perspective, the terminal must check out, change UTP and check in again.</p> <p>There is no validity period for changed user tariff parameters. As long as the UTP remain in the changed version, they are passed through the terminals and copied to the next recording transaction. A customer must roll back the changes at a CCP-T if requested.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Check user medium with application as CCP / Perform user tariff parameters change and notify / Get entitlement and check attestations
Linked Use Cases (Realises)	
Base Activity	
Inputs	
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Change user tariff parameters

19.2.2 Optional: Perform user tariff parameters change and notify



Use Case	Perform user tariff parameters change and notify
Description	The CCP terminal performs a transaction to change user tariff parameters and notifies the responsible CCP back-office system about the change. A detected transaction abortion is also sent to the CCP system for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked Session is established Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle user tariff parameters changed notification from operational perspective
Linked Use Cases (Realises)	



Base Activity	
Inputs	Action tariff parameters : ActionTariffParameters Entitlement directory entry : EntitlementDirectoryEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Perform user tariff parameters change and notify



20 Ordered Action Management Bundle Executing CCP-Terminal

Functionality bundle that covers use cases to implement CCP terminal functionality for executing (ordered) actions located in a distributed action list.

20.1 Overview

Execute action list entries

Issue entitlement triggered by action order

Perform ordered entitlement issuance and notify

Unblock entitlement triggered by action order

Perform ordered entitlement unblocking and notify

Terminate entitlement triggered by action order

Perform ordered entitlement termination and notify

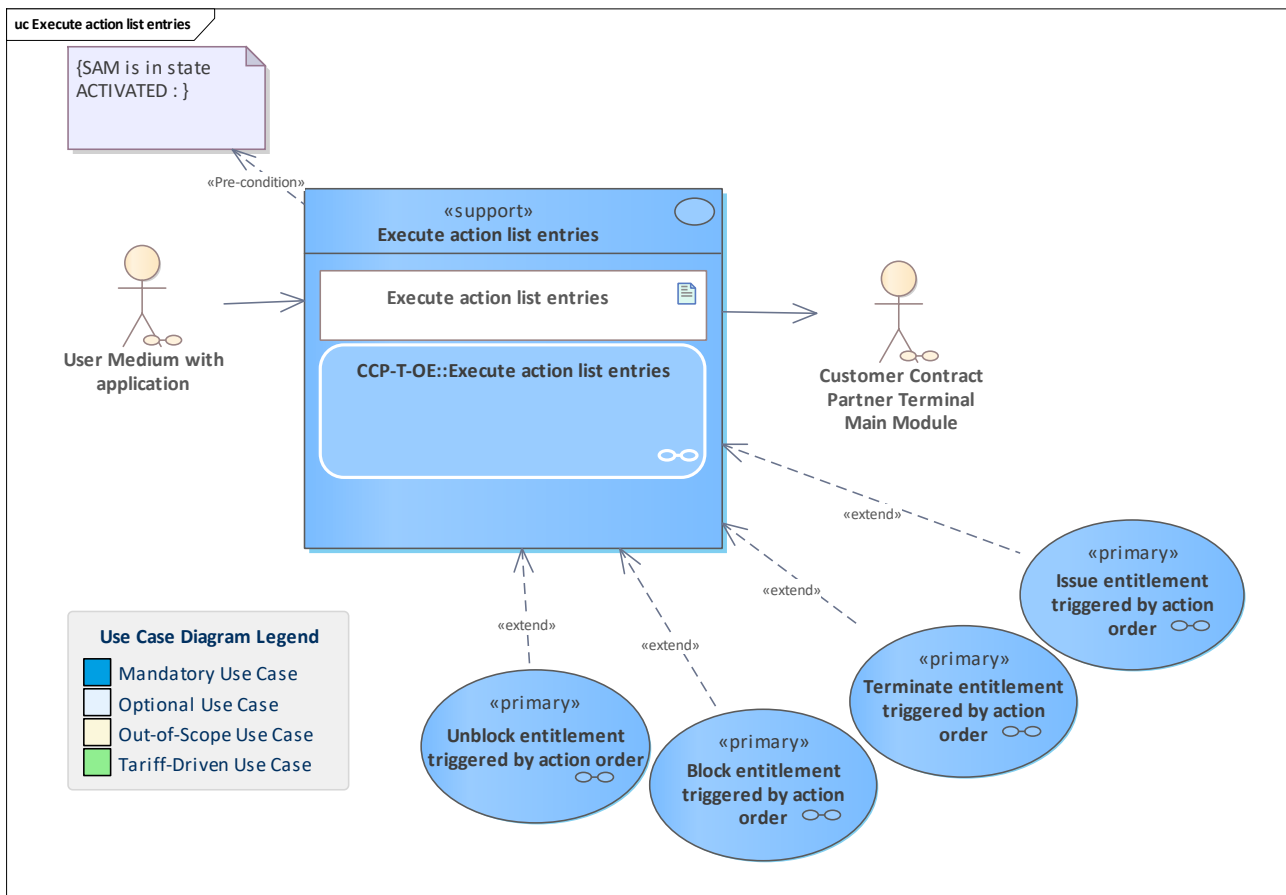
Block entitlement triggered by action order

Perform ordered entitlement blocking and notify

Update terminal action list

20.2 Use Cases

20.2.1 Execute action list entries

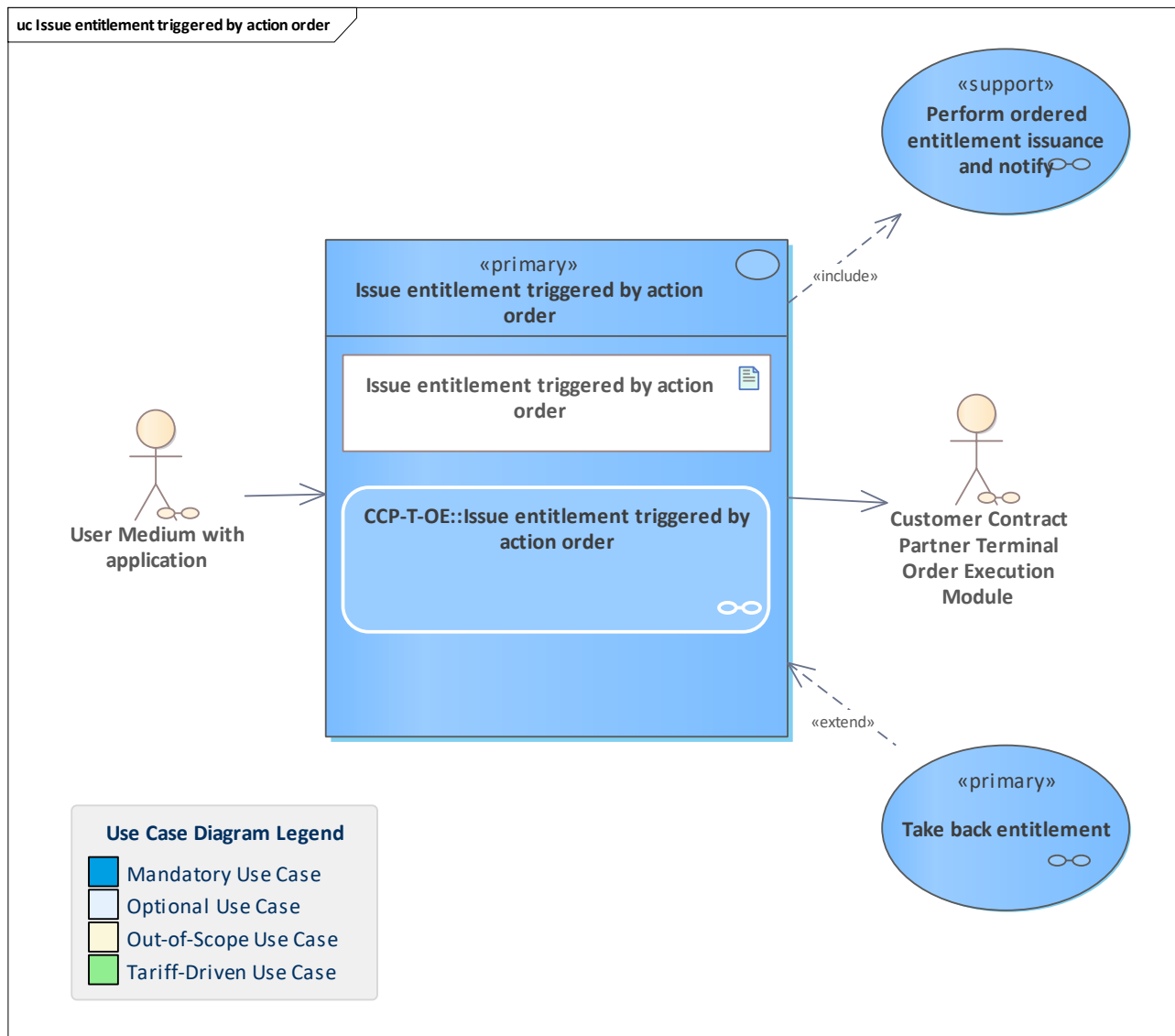


Use Case	<u>Execute action list entries</u>
Description	Check for action list entries and, if necessary, execute them. Triggered by the general processes of <u>CCP-T-Main::Check user medium with application</u> if an action management extension exists. The action list is searched for possible actions for the user medium which is currently presented to the CCP terminal. If one or more action entries exist for the application instance ID of the current user medium, it is executed. Note that this process may only be run if the SAM is activated to guarantee that no termination is executed for which a linked (replacement) issuance cannot be executed.
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Main Module</u>
Preconditions	<u>SAM is in state ACTIVATED</u>
Postconditions	
Linked Use Cases (Extended By)	<u>Unblock entitlement triggered by action order</u> / <u>Block entitlement triggered by action order</u> / <u>Terminate entitlement triggered by action order</u> / <u>Issue entitlement triggered by action order</u>
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	



Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Execute action list entries

20.2.2 Issue entitlement triggered by action order

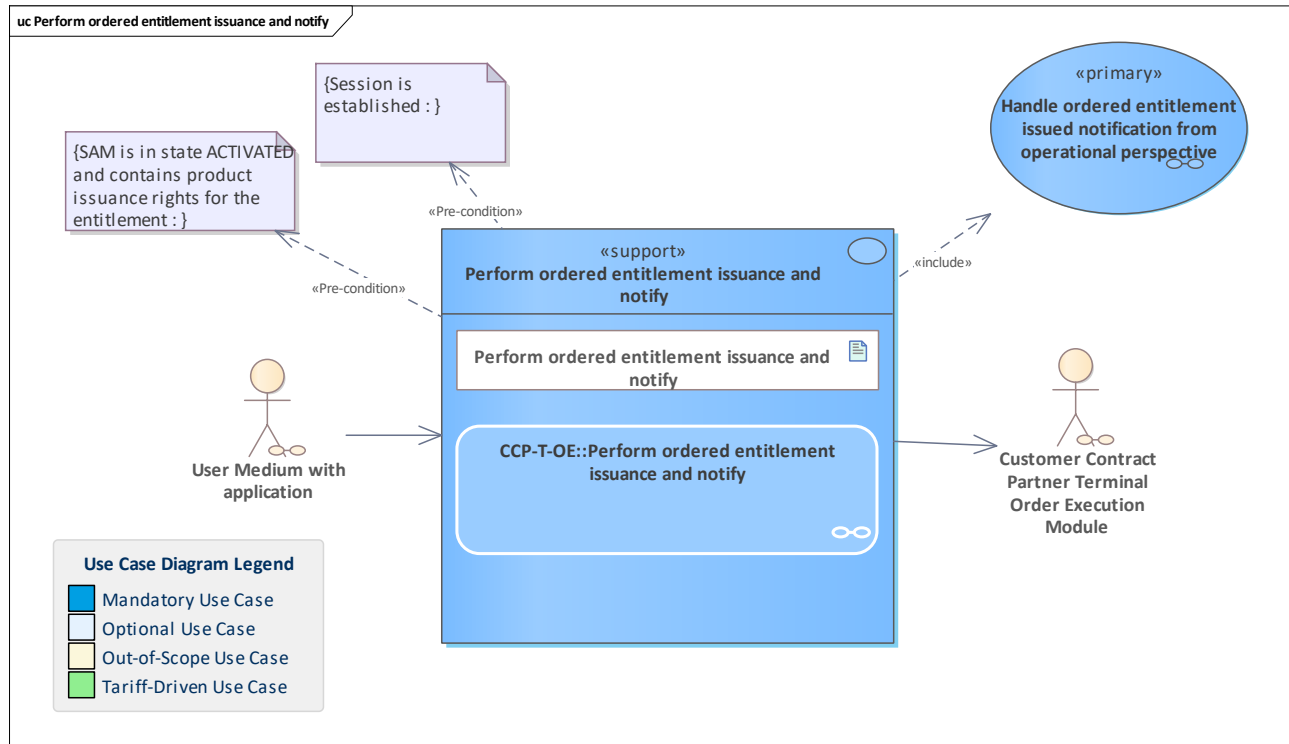


Use Case	<u>Issue entitlement triggered by action order</u>
Description	<p>An entitlement issuance order potentially relevant for a given user medium is checked regarding the need to execute it and is executed, if necessary. The issuance process is similar to the regular issuance process without action management. Additionally, the order ID is written into the issued entitlement to avoid further issuance attempts in other terminals.</p> <p>The terminal does a lookup in the action list for the application instance ID. To avoid a duplicate issuance, the user medium has to be examined, if an entitlement with the same order ID already exists (which was written previously into the entitlement).</p>
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Order Execution Module</u>
Preconditions	
Postconditions	
Linked Use Cases	<u>Take back entitlement</u>



(Extended By)	
Linked Use Cases (Includes)	Perform ordered entitlement issuance and notify
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement issuance action list entry : EntitlementIssuanceActionListEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Issue entitlement triggered by action order

20.2.3 Perform ordered entitlement issuance and notify

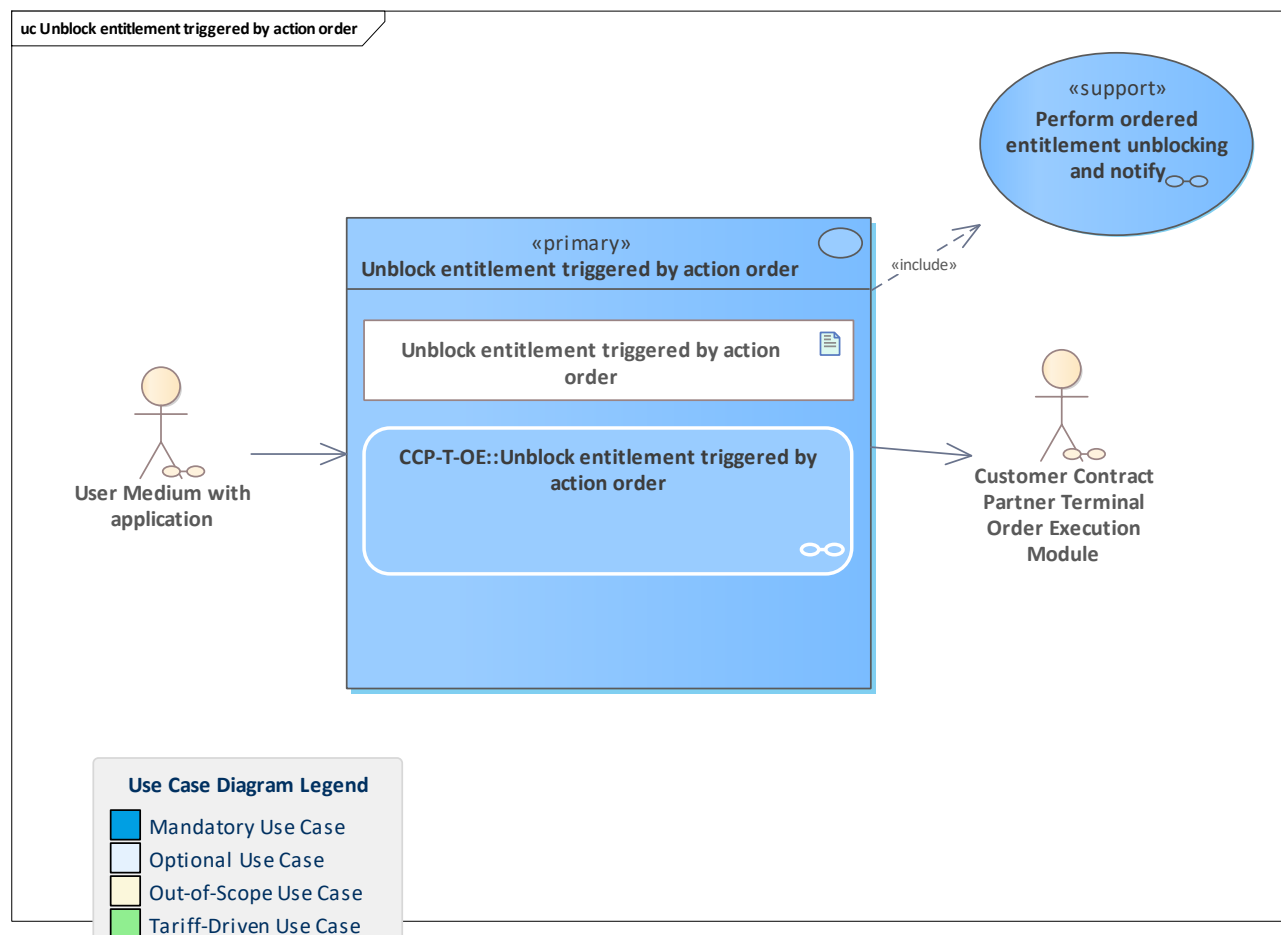


Use Case	Perform ordered entitlement issuance and notify
Description	The CCP terminal with an action management extension executes an entitlement issuance order and notifies the own back-office system about it. If the transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	Session is established SAM is in state ACTIVATED and contains product issuance rights for the entitlement
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle ordered entitlement issued notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action operator : OrganisationId CCP organisation ID : OrganisationId Entitlement effective time : EntitlementEffectiveTime Entry ID : EntryId Infotext : Infotext Order number : OrderNumber Entitlement expiration time : EntitlementExpirationTime



	Product ID : ProductId Product parameters : ProductParameters Stored-value autoloader parameters : StoredValueAutoloadParameters Stored-value parameters : StoredValueParameters
Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Perform ordered entitlement issuance and notify

20.2.4 Unblock entitlement triggered by action order

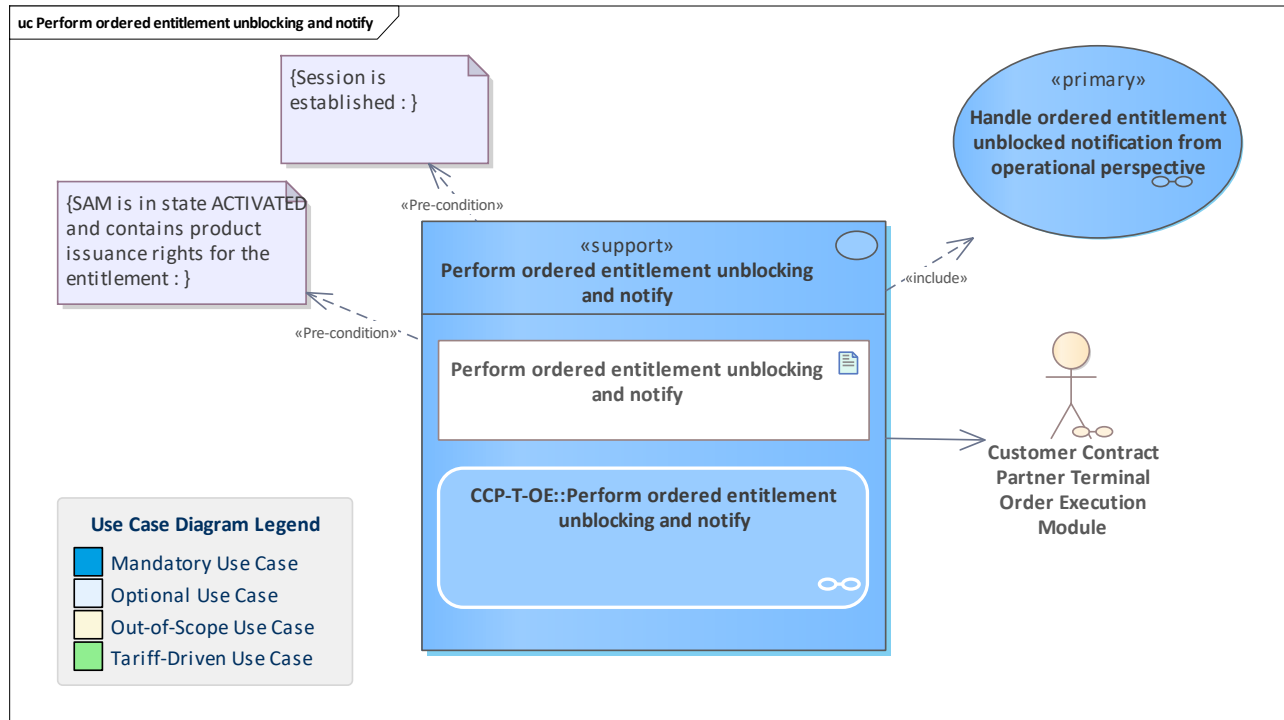


Use Case	Unblock entitlement triggered by action order
Description	<p>An entitlement unblocking order that is potentially relevant for a given user medium is checked regarding the need to execute it and, if necessary, is executed.</p> <p>To achieve this, the terminal does a lookup in the action list for the application instance ID. If the ID matches, the entitlements are examined. If the entitlement ID of one entitlement matches with the entry in the action list, the entitlement in question is unblocked.</p> <p>To avoid a duplicate unblocking attempt, the user medium has to be examined to check if the entitlement in question is already in the regular state.</p>
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the entitlement
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform ordered entitlement unblocking and notify



Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement unblocking action list entry : EntitlementUnblockingActionListEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Unblock entitlement triggered by action order

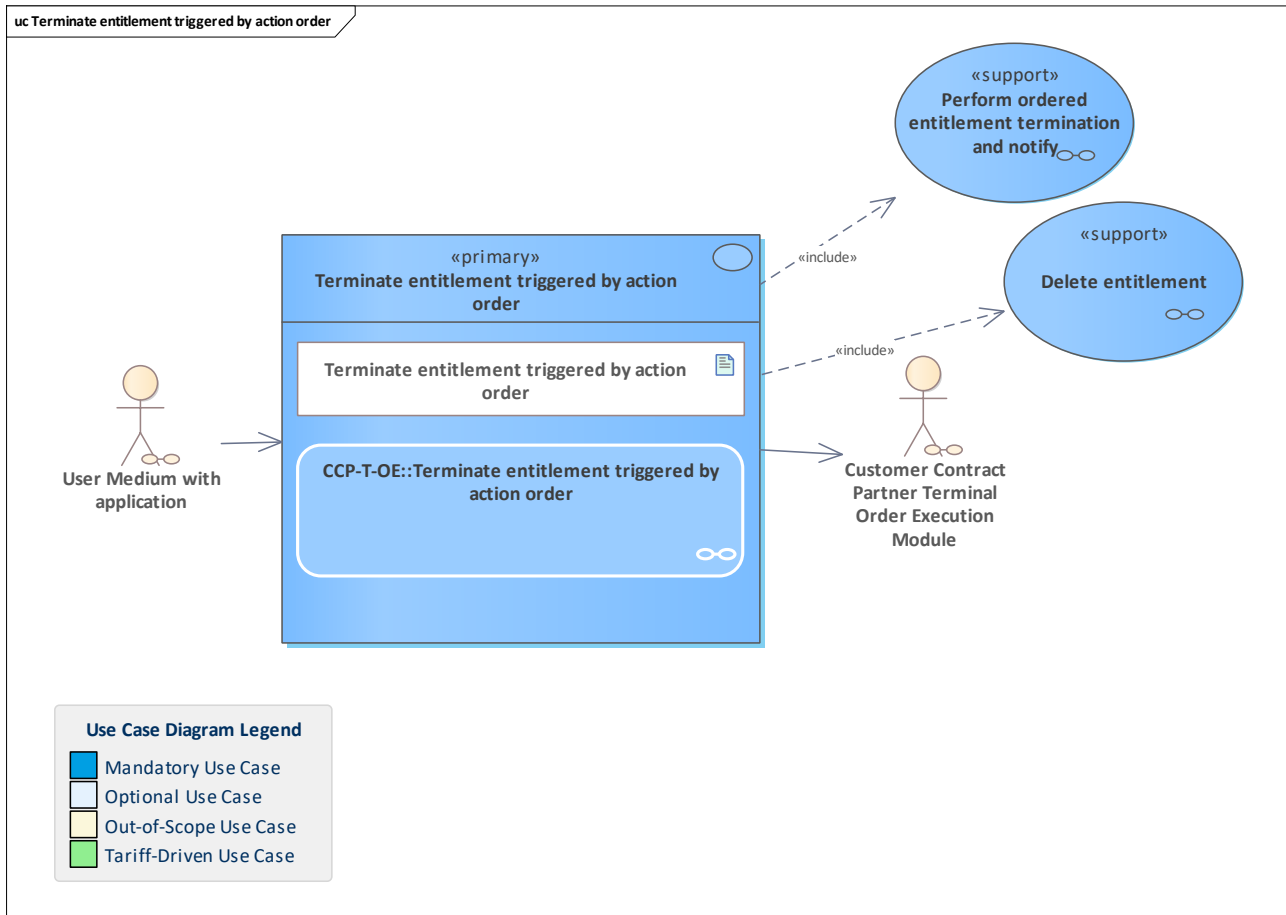
20.2.5 Perform ordered entitlement unblocking and notify



Use Case	Perform ordered entitlement unblocking and notify
Description	The CCP terminal executes an entitlement unblocking order and notifies the own back-office system about it. If the transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	Session is established SAM is in state ACTIVATED and contains product issuance rights for the entitlement
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle ordered entitlement unblocked notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action operator : OrganisationId Entitlement directory entry : EntitlementDirectoryEntry Order ID : OrderId
Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Perform ordered entitlement unblocking and notify



20.2.6 Terminate entitlement triggered by action order

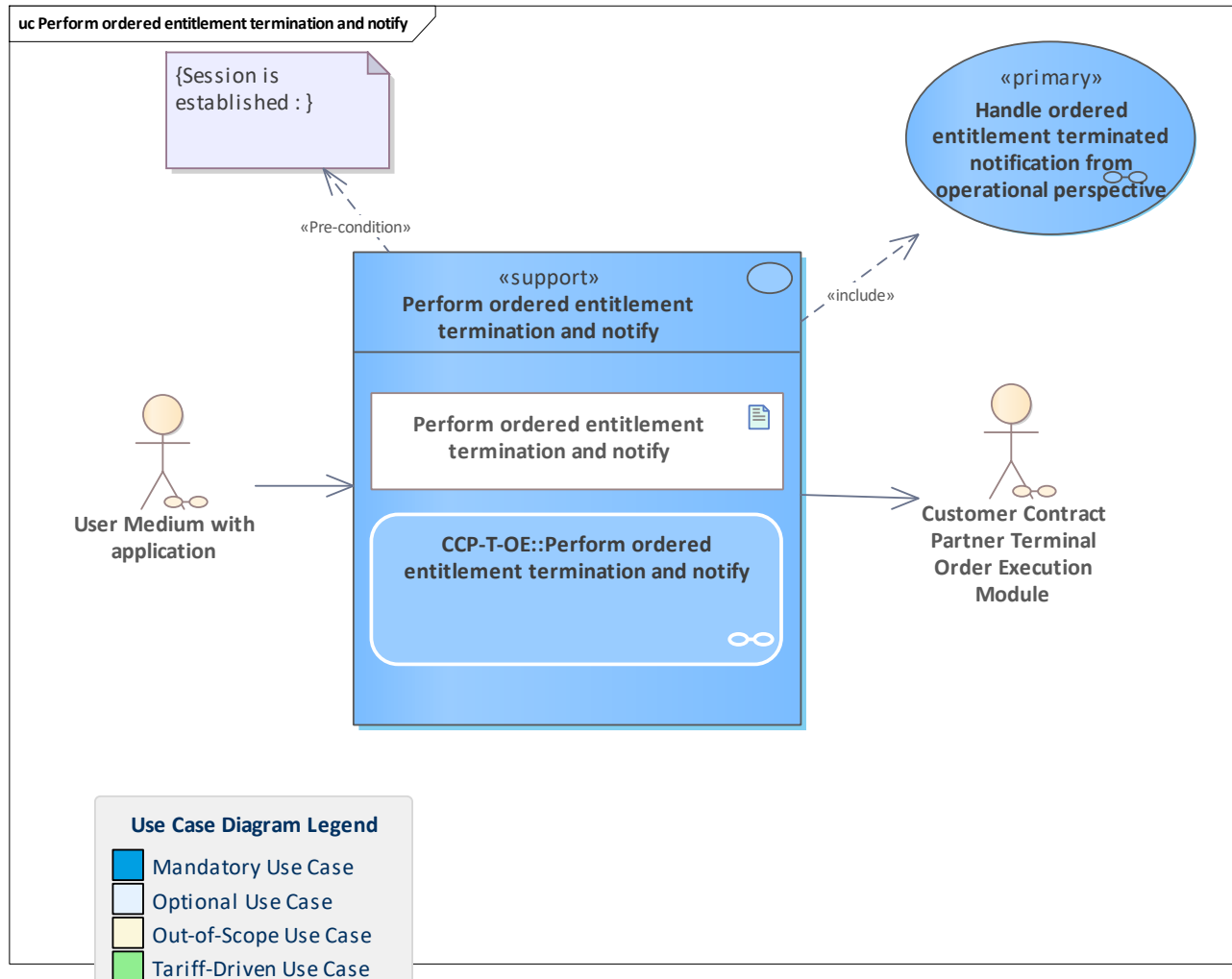


Use Case	Terminate entitlement triggered by action order
Description	An entitlement termination order that is potentially relevant for a given user medium is checked regarding the need to execute it and is executed if necessary. To achieve this, the terminal does a lookup in the action list for the application instance ID. To avoid a duplicate termination attempt, the user medium has to be examined to check if the entitlement in question is already in the state Entitlement terminated .
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform ordered entitlement termination and notify / Delete entitlement
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement termination action list entry : EntitlementTerminationActionListEntry



Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Terminate entitlement triggered by action order

20.2.7 Perform ordered entitlement termination and notify

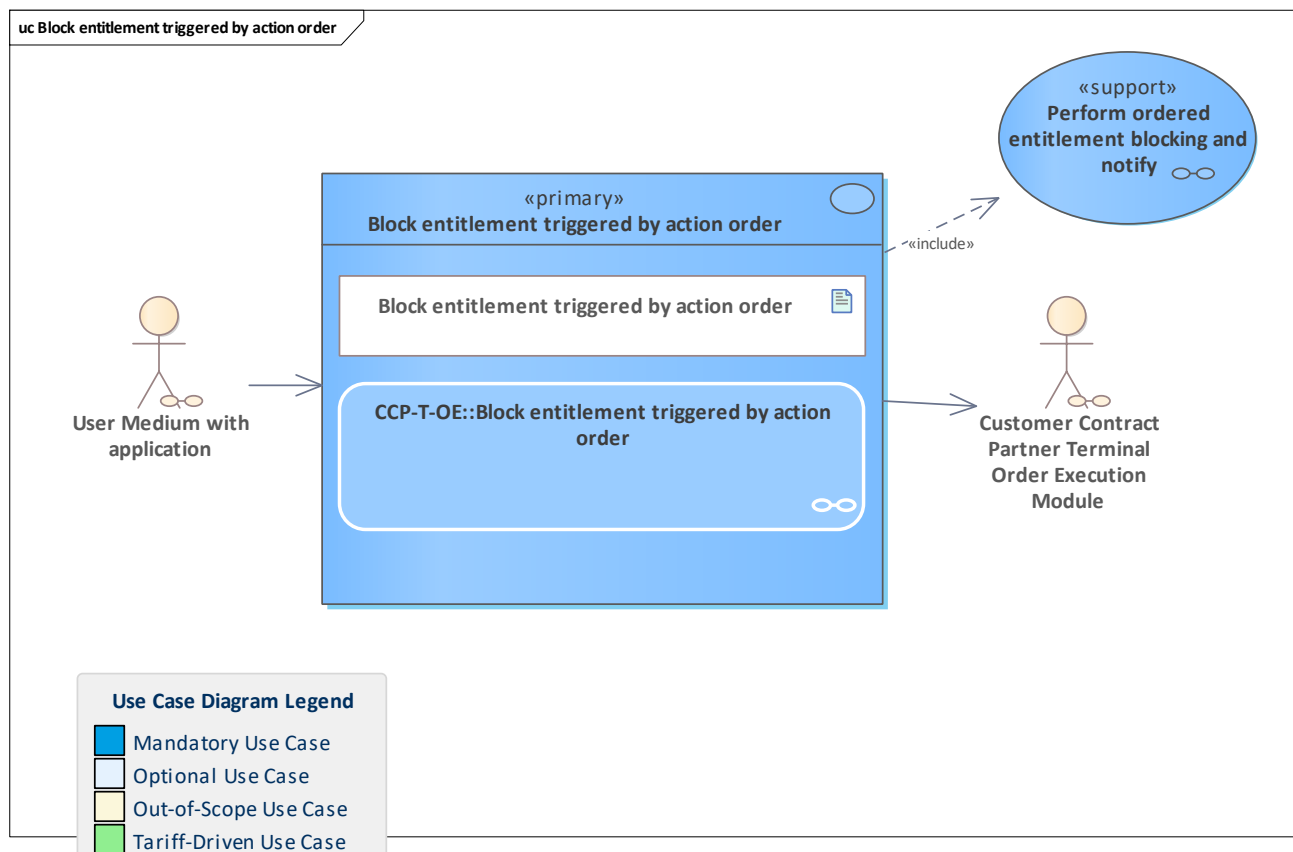


Use Case	Perform ordered entitlement termination and notify
Description	Execute an entitlement termination order in the CCP terminal with an action management extension and notify the own back-office system about it. If the transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	User Medium with application
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle ordered entitlement terminated notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	



Inputs	<u>Action operator : OrganisationId</u> <u>Order ID : OrderId</u> <u>Entitlement directory entry : EntitlementDirectoryEntry</u>
Outputs	
Error Cases	
Activity Diagram	<u>CCP-T-OE::Perform ordered entitlement termination and notify</u>

20.2.8 Block entitlement triggered by action order

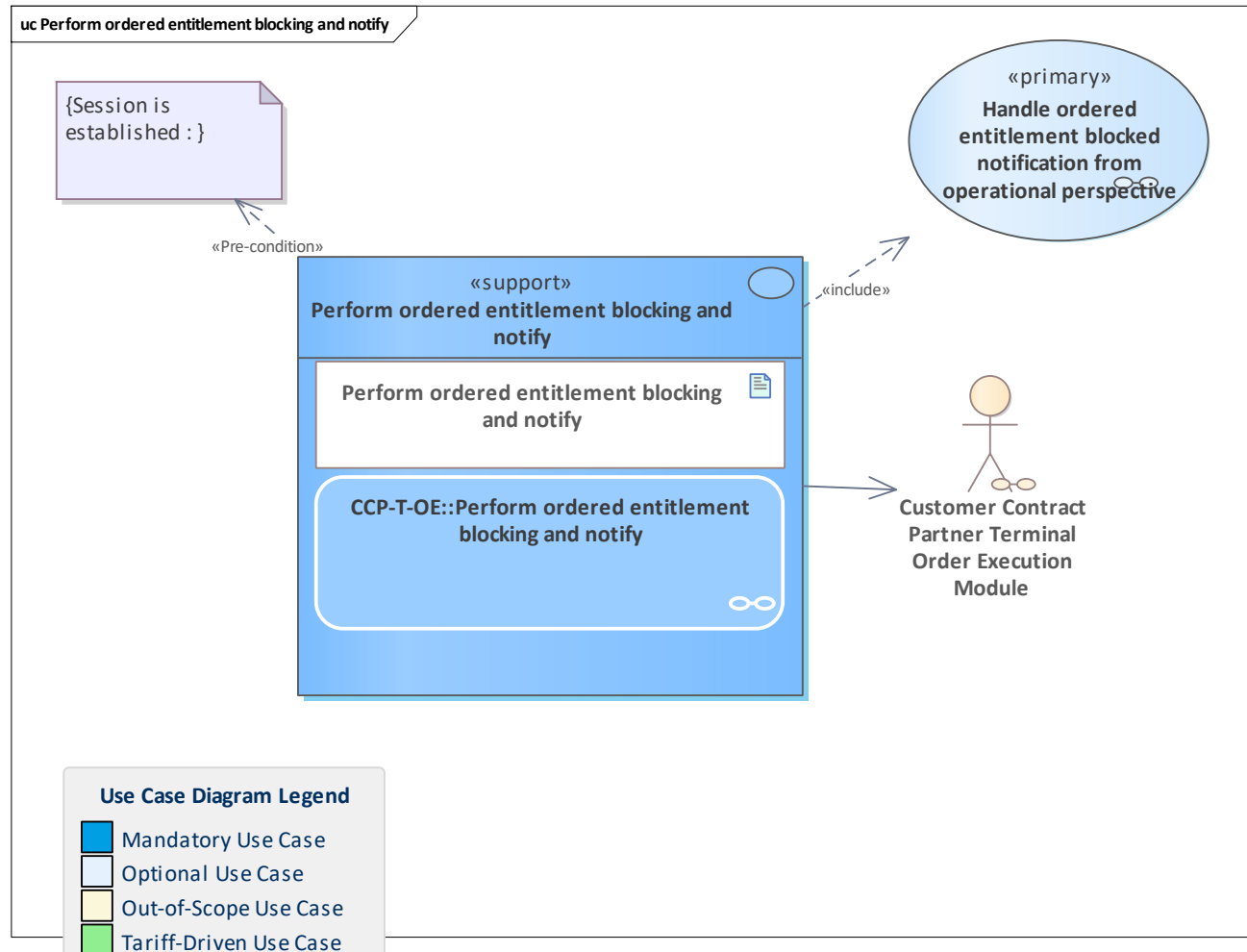


Use Case	<u>Block entitlement triggered by action order</u>
Description	<p>An potentially relevant entitlement blocking order for a certain user medium is checked regarding the need to execute it and, if necessary, is executed.</p> <p>To achieve this, the terminal does a lookup in the action list for the application instance ID. If a match is found, the entitlements are examined concerning their potential order ID. If the order ID matches with the ordered blocking action in the action list and the entitlement is not in the state <u>Entitlement blocked</u>, the blocking is performed.</p> <p>This use case aims at blocking an entitlement issued via action management which an unknown ID. Thus, a regular hotlist entry in the hotlist service is not possible.</p> <p>Note: when an entitlement is blocked due to an action list entry, no extended logging regarding its status should be created during that same interaction. Extended logging regarding entitlement status should only be created for entitlements that were already invalid per status before the interaction with a terminal.</p> <p>Note that this is not relevant for entitlements terminated via action list entries since these are also removed from the user medium application.</p>
Initiating Actor	<u>User Medium with application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Order Execution Module</u>
Preconditions	



Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Perform ordered entitlement blocking and notify
Linked Use Cases (Realises)	
Base Activity	
Inputs	Entitlement blocking action list entry : EntitlementBlockingActionListEntry
Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Block entitlement triggered by action order

20.2.9 Perform ordered entitlement blocking and notify

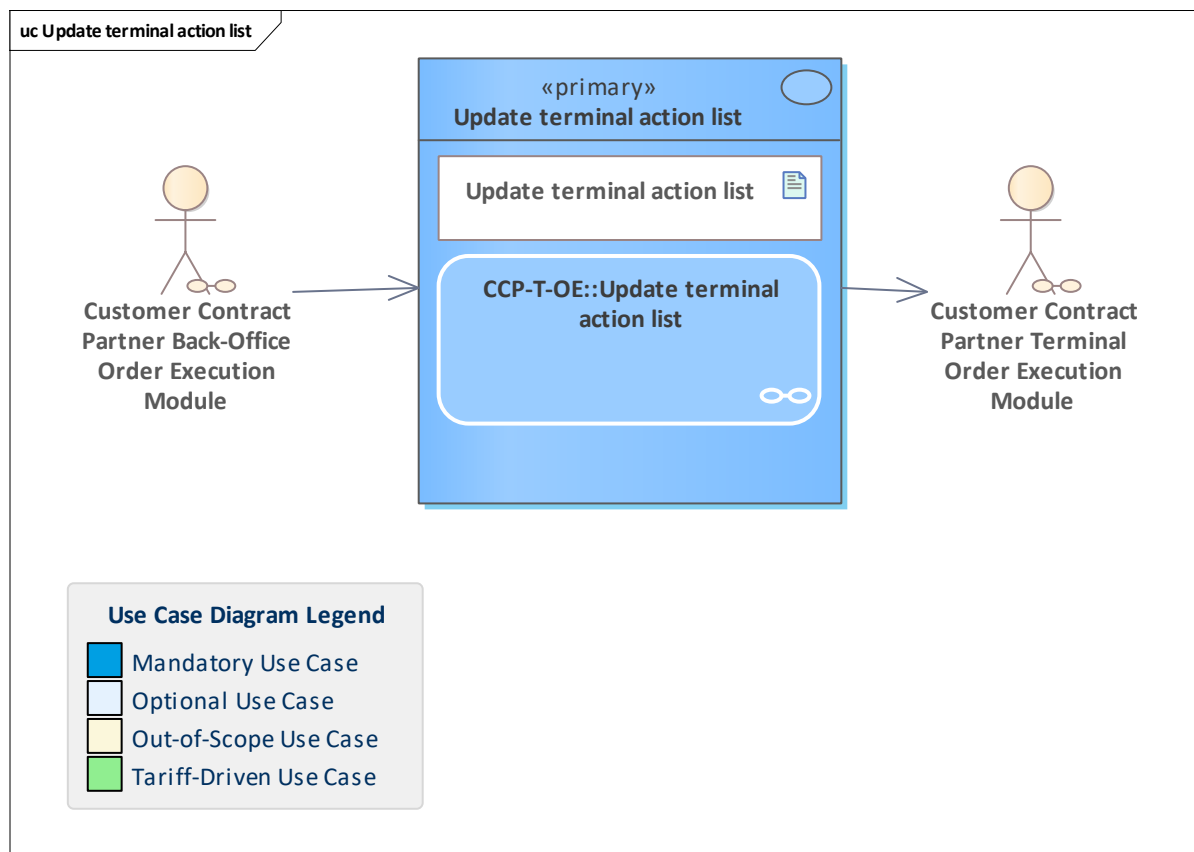


Use Case	Perform ordered entitlement blocking and notify
Description	Execute an entitlement blocking order and notify the own back-office system about it. If the transaction is aborted, the CCP back-office system is also notified for consistent monitoring data.
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	Session is established
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle ordered entitlement blocked notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Action operator : OrganisationId Entitlement directory entry : EntitlementDirectoryEntry Order ID : OrderId



Outputs	
Error Cases	
Activity Diagram	CCP-T-OE::Perform ordered entitlement blocking and notify

20.2.10 Update terminal action list



Use Case	Update terminal action list
Description	The executing CCP back-office system stores the latest action list in the terminal.
Initiating Actor	Customer Contract Partner Back-Office Order Execution Module
Reacting Actor	Customer Contract Partner Terminal Order Execution Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	Terminal action list : tActionList
Outputs	Terminal action list reponse : tActionListResponse
Error Cases	Terminal action list exception : tActionListException
Activity Diagram	CCP-T-OE::Update terminal action list



21 Static Entitlements Bundle CCP-Terminal

Functionality bundle that covers CCP terminal use cases for working with static entitlements.

21.1 Overview

Check user medium without application

Issue static entitlement

Authorise static entitlement

Take back static entitlement

Reimburse and terminate static entitlement

Notify static entitlement terminated

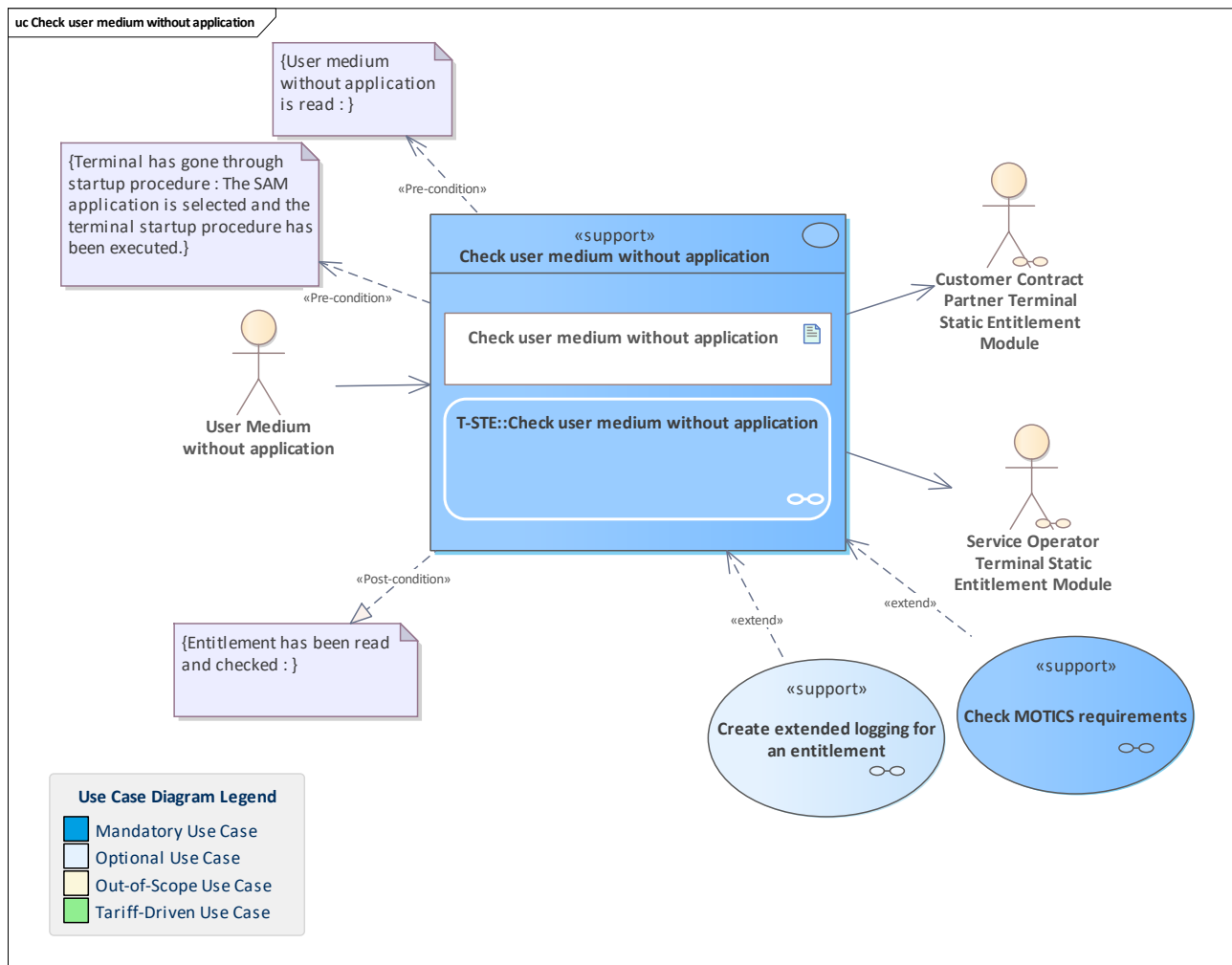
Change static entitlement

Display static entitlement

Optional: Print customer receipt

21.2 Use Cases

21.2.1 Check user medium without application

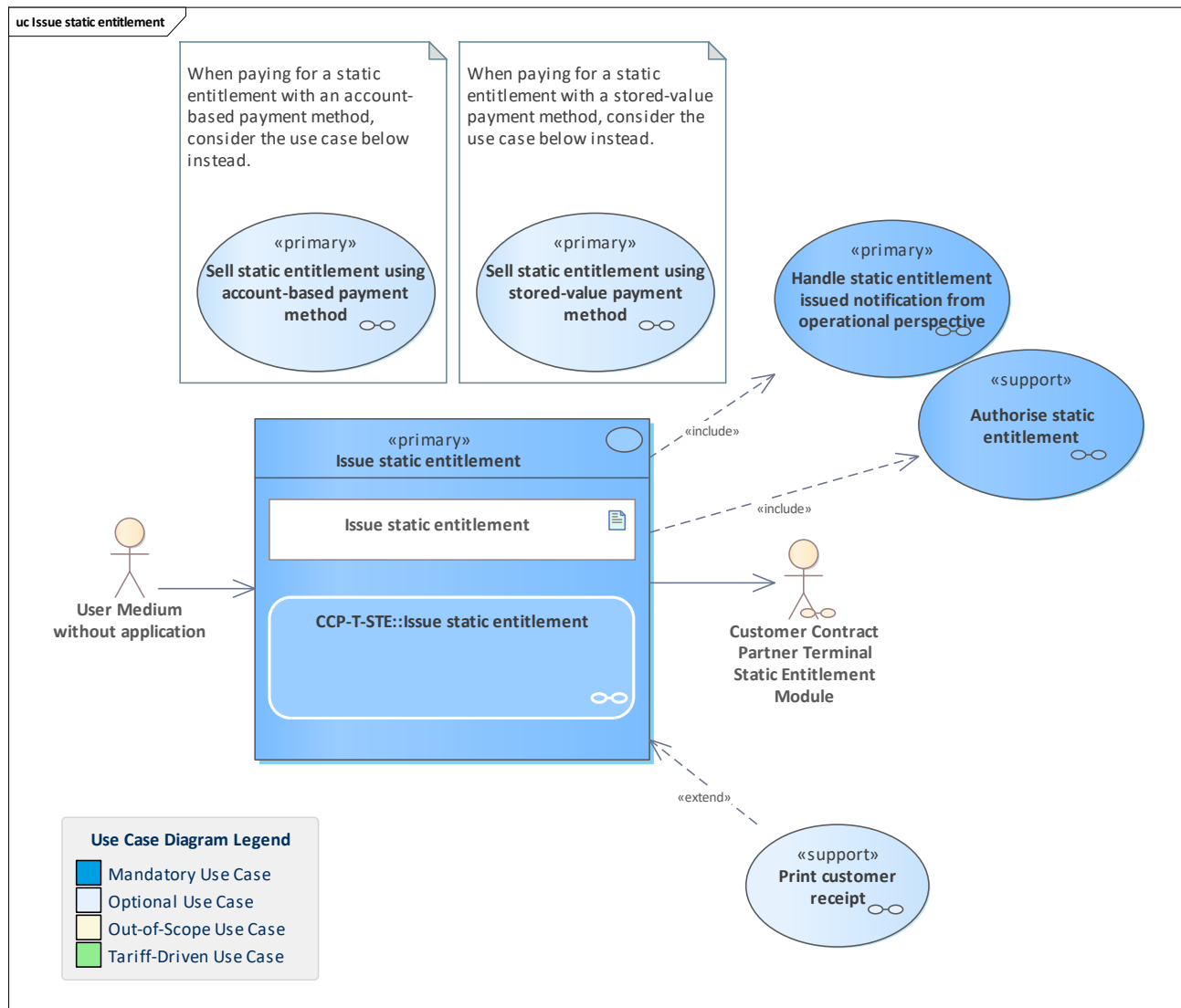


Use Case	Check user medium without application
Description	This use case has the following steps: <ul style="list-style-type: none"> Checking MOTICS requirements if applicable Checking static entitlements
Initiating Actor	User Medium without application
Reacting Actor	Service Operator Terminal Static Entitlement Module Customer Contract Partner Terminal Static Entitlement Module
Preconditions	Terminal has gone through startup procedure User medium without application is read User medium without application is read Terminal has gone through startup procedure
Postconditions	Entitlement has been read and checked
Linked Use Cases (Extended By)	Check MOTICS requirements / Create extended logging for an entitlement
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	Entitlement has been read and checked
Base Activity	
Inputs	Binary static ticket data



Outputs	List of valid static entitlement data : StaticEntitlementData
Error Cases	
Activity Diagram	T-STE::Check user medium without application

21.2.2 Issue static entitlement

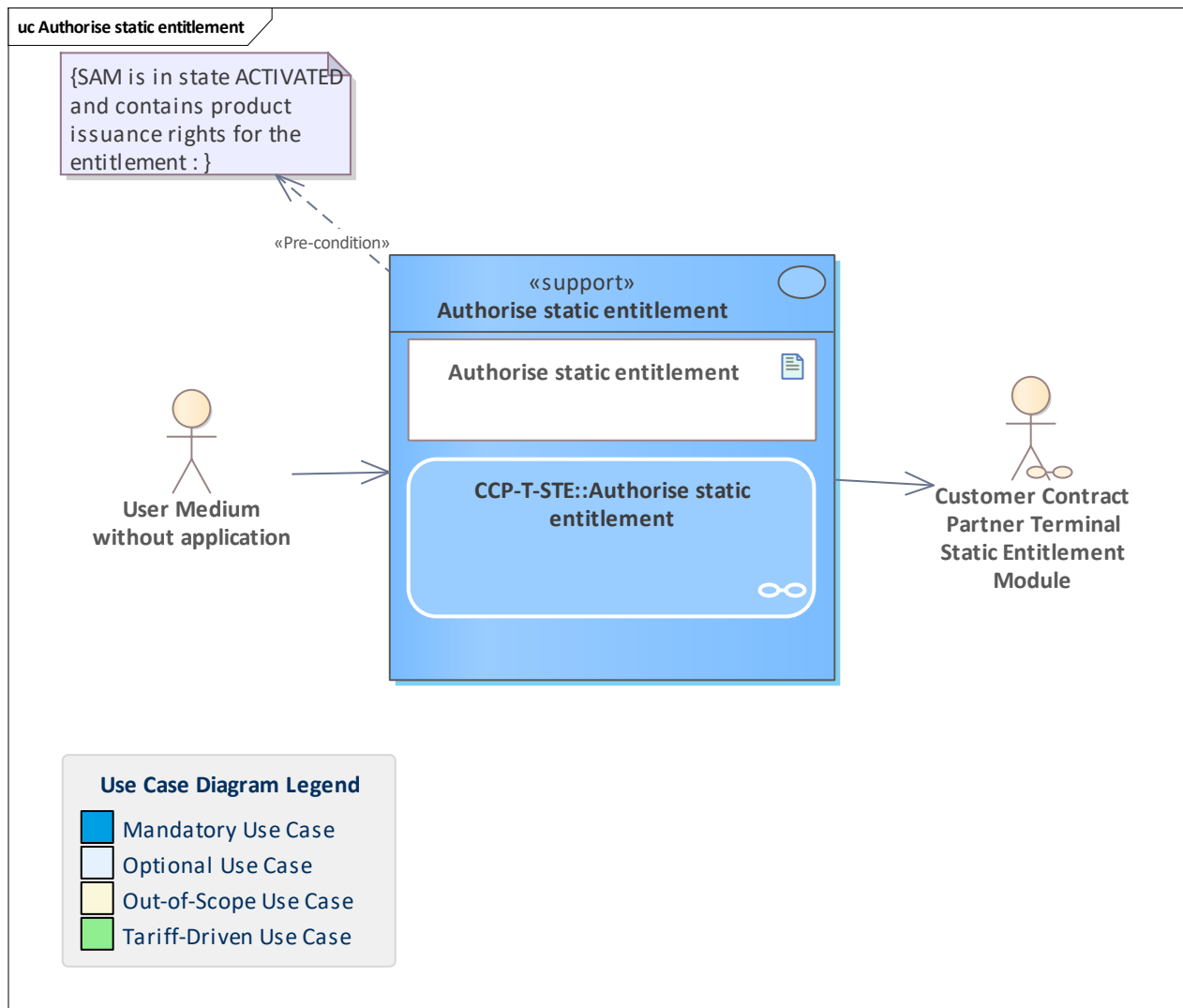


Use Case	Issue static entitlement
Description	Issue a static entitlement and notify the back-office system. Note: this use case only issues a single static electronic ticket. In case multiple such issuances should be performed in the same context, the StaticEntitlements structures can be merged by combining the contained StaticEntitlementData objects, thus deduplicating the SAM certificate. This merging can either happen before delivering the data objects to the target (e.g. if it is a printer) or afterwards (e.g. if it is a smartphone). The back-office systems receive their notifications for every single static electronic ticket (i.e. StaticEntitlementData object), however.
Initiating Actor	User Medium without application
Reacting Actor	Customer Contract Partner Terminal Static Entitlement Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Print customer receipt / Print customer receipt / Print customer receipt



Linked Use Cases (Includes)	Handle static entitlement issued notification from operational perspective / Issue static entitlement / Issue static entitlement / Authorise static entitlement
Linked Use Cases (Realises)	
Base Activity	
Inputs	Replaced entitlement : EntitlementId SCE ID : AppInstanceId Infotext : Infotext Product parameters : ProductParameters Entitlement expiration time : EntitlementExpirationTime Entitlement effective time : EntitlementEffectiveTime CCP organisation ID : OrganisationId Product ID : ProductId
Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Issue static entitlement

21.2.3 Authorise static entitlement

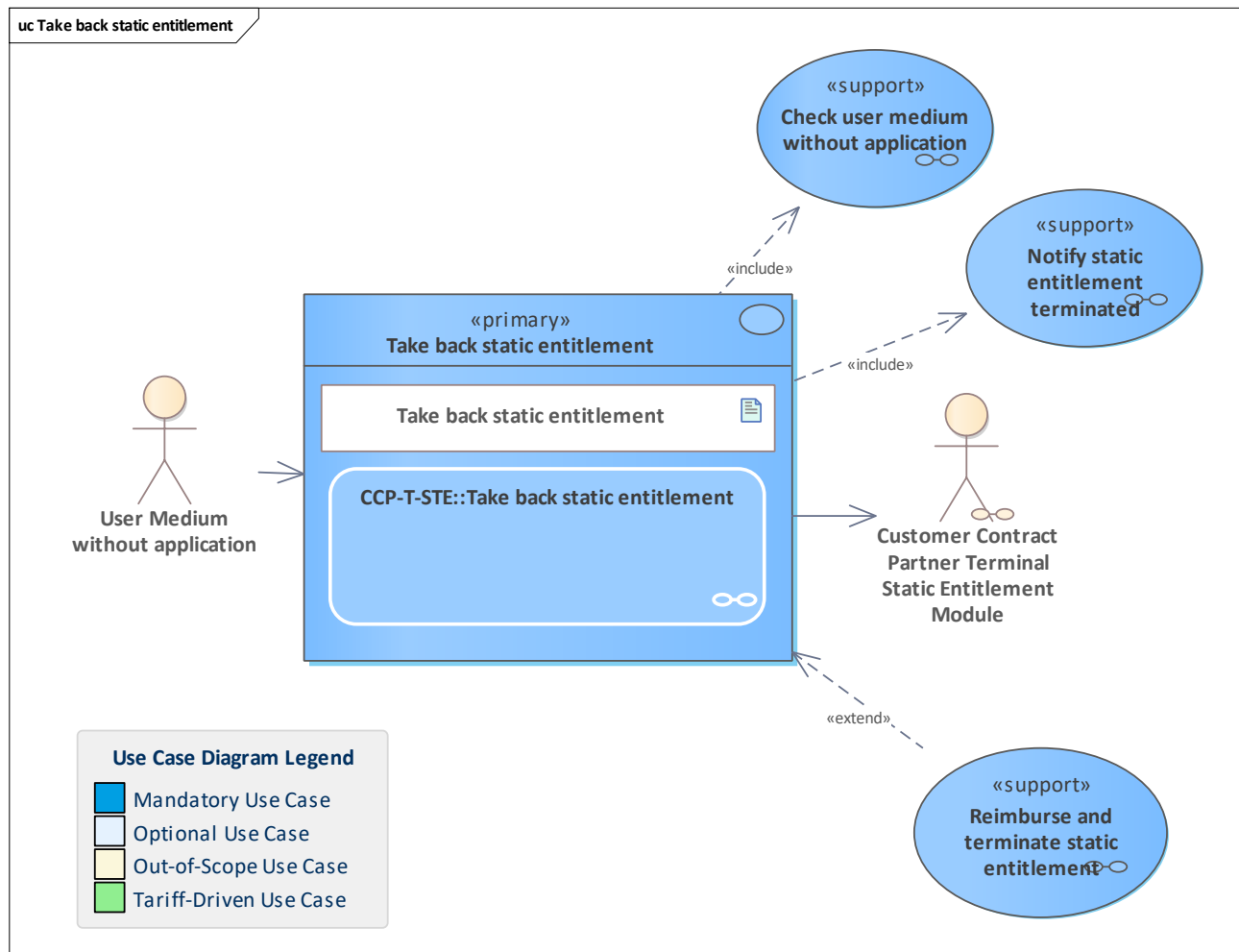


Use Case	Authorise static entitlement
Description	The customer contract partner terminal creates a static entitlement using a SAM. Supporting use case that performs the creation without the output to a user medium (mobile device, paper, etc.). Note that this process only puts a single static entitlement data object into the static entitlements object. By running this process multiple times using the same SAM and afterwards merging the static entitlements objects, more than one entitlement (i.e. static entitlement data object) can be put into a single static entitlements object.
Initiating Actor	User Medium without application
Reacting Actor	Customer Contract Partner Terminal Static Entitlement Module
Preconditions	SAM is in state ACTIVATED and contains product issuance rights for the entitlement
Postconditions	
Linked Use Cases (Extended By)	



Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Order number : OrderNumber</u> <u>Infotext : Infotext</u> <u>Product parameters : ProductParameters</u> <u>Entitlement expiration time : EntitlementExpirationTime</u> <u>Entitlement effective time : EntitlementEffectiveTime</u> <u>CCP organisation ID : OrganisationId</u> <u>Product ID : ProductId</u> <u>SCE ID : AppInstanceId</u>
Outputs	<u>Static entitlement : StaticEntitlements</u>
Error Cases	
Activity Diagram	<u>CCP-T-STE::Authorise static entitlement</u>

21.2.4 Take back static entitlement

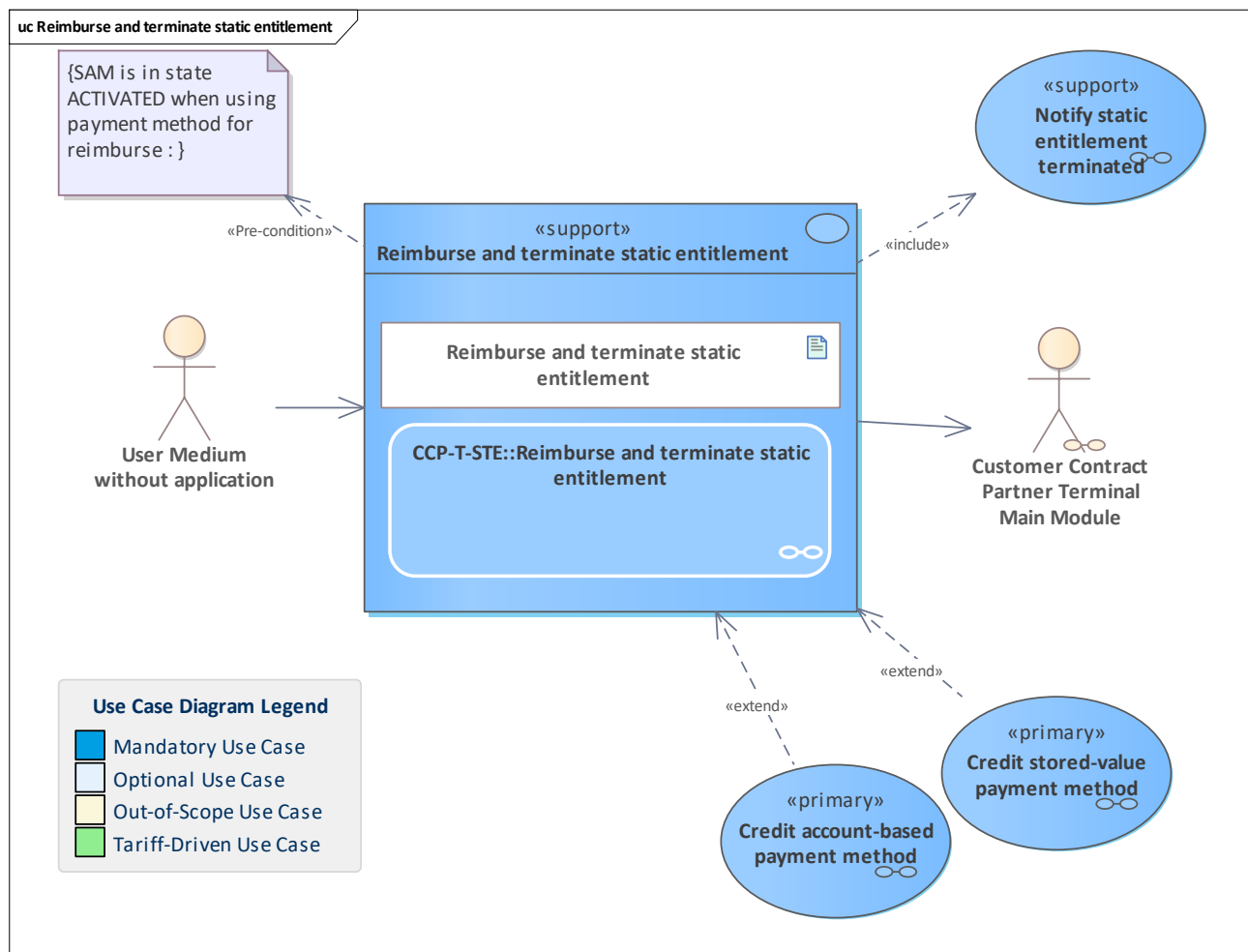


Use Case	<u>Take back static entitlement</u>
Description	<p>A CCP terminal with static entitlement extension starts the process. Taking back of a static entitlement process (including creating and sending notifications) is always required when static entitlements that are still valid in terms of time are taken back and, in particular, when a refund is in question. Since there is no guarantee that copies might still be in circulation, this entitlement must be hotlisted as soon as possible and the option for taking back should be possible only for a pCCP.</p> <p>If a reimbursement is marked as available in the tariff regulations, this use case, in conjunction with a reimbursement, is only permissible with the CCP that issued the entitlement (primary CCP). In this case, reimbursement conditions defined within product modules for a product must be evaluated. Irrespective of whether only the termination was performed or the additional reimbursement, the responsible CCP back-office system will be informed.</p>
Initiating Actor	<u>User Medium without application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Static Entitlement Module</u>



Preconditions	
Postconditions	
Linked Use Cases (Extended By)	Reimburse and terminate static entitlement
Linked Use Cases (Includes)	Check user medium without application / Notify static entitlement terminated / Notify static entitlement terminated
Linked Use Cases (Realises)	
Base Activity	
Inputs	Binary static ticket data
Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Take back static entitlement

21.2.5 Reimburse and terminate static entitlement

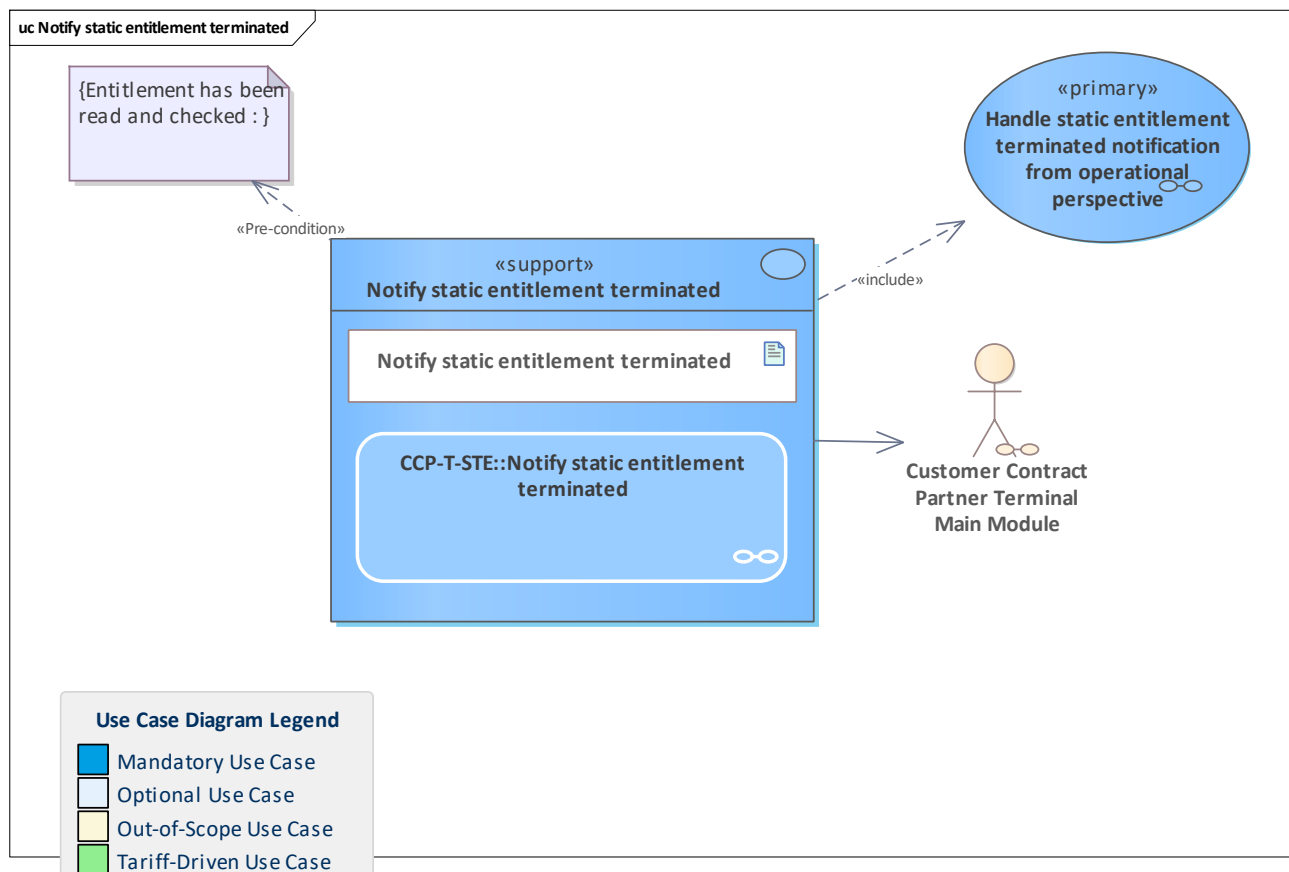


Use Case	Reimburse and terminate static entitlement
Description	<p>A static entitlement is given back/terminated, and it was decided that the static entitlement can be reimbursed.</p> <p>There are 3 possibilities for reimbursement:</p> <ul style="list-style-type: none"> • Credit a stored-value payment method on a user medium with an application if supported • Credit an account-based payment method of the customer if supported • Legal tender (must be supported)
Initiating Actor	User Medium without application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	SAM is in state ACTIVATED when using payment method for reimburse
Postconditions	
Linked Use Cases (Extended By)	Credit stored-value payment method / Credit account-based payment method
Linked Use Cases (Includes)	Notify static entitlement terminated



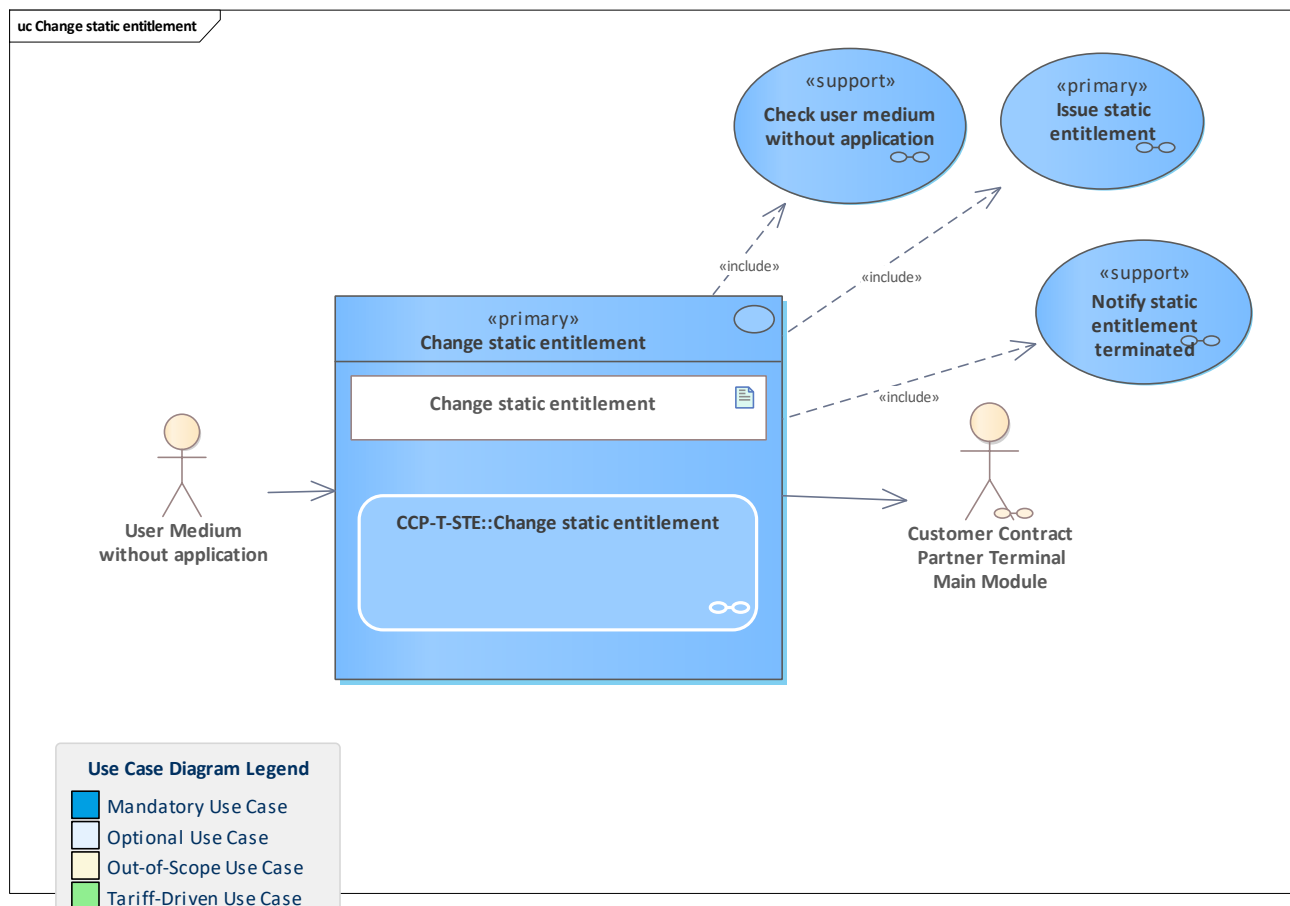
Linked Use Cases (Realises)	
Base Activity	
Inputs	Static entitlement data : StaticEntitlementData
Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Reimburse and terminate static entitlement

21.2.6 Notify static entitlement terminated



Use Case	Notify static entitlement terminated
Description	The termination of a static entitlement is notified by the CCP terminal with static entitlement extension to the responsible CCP back-office system.
Initiating Actor	
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	Entitlement has been read and checked
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Handle static entitlement terminated notification from operational perspective
Linked Use Cases (Realises)	
Base Activity	
Inputs	Static entitlement data : StaticEntitlementData
Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Notify static entitlement terminated

21.2.7 Change static entitlement

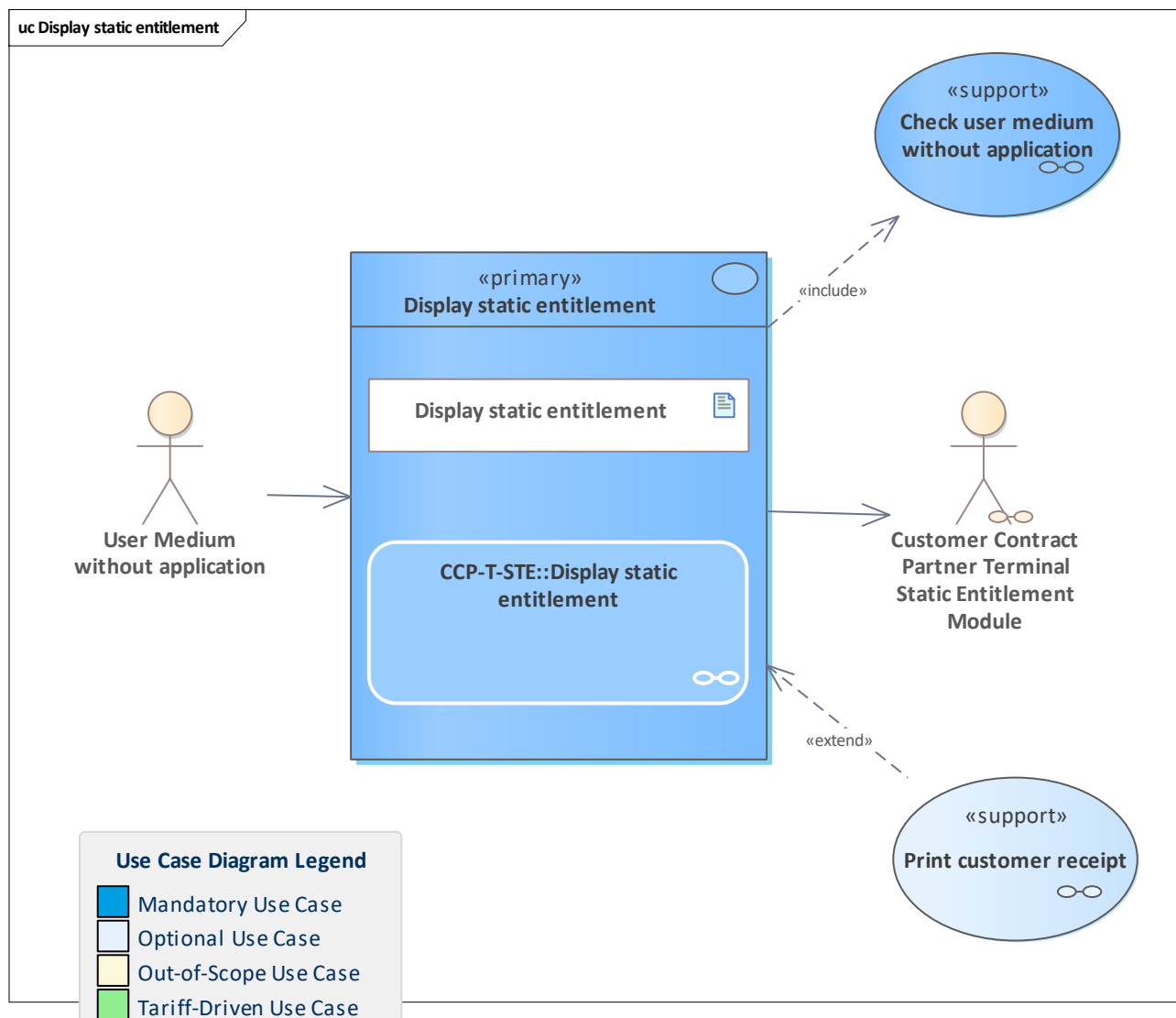


Use Case	Change static entitlement
Description	<p>A static entitlement needs to be replaced with a new one, for example, due to changed product parameters.</p> <p>Please note that it is assumed that there is no need for any payment transaction.</p>
Initiating Actor	User Medium without application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	Issue entitlement / Issue static entitlement / Notify static entitlement terminated / Check user medium without application
Linked Use Cases (Realises)	
Base Activity	
Inputs	Binary static data
Outputs	
Error Cases	



Activity Diagram	CCP-T-STE::Change static entitlement
-------------------------	--

21.2.8 Display static entitlement

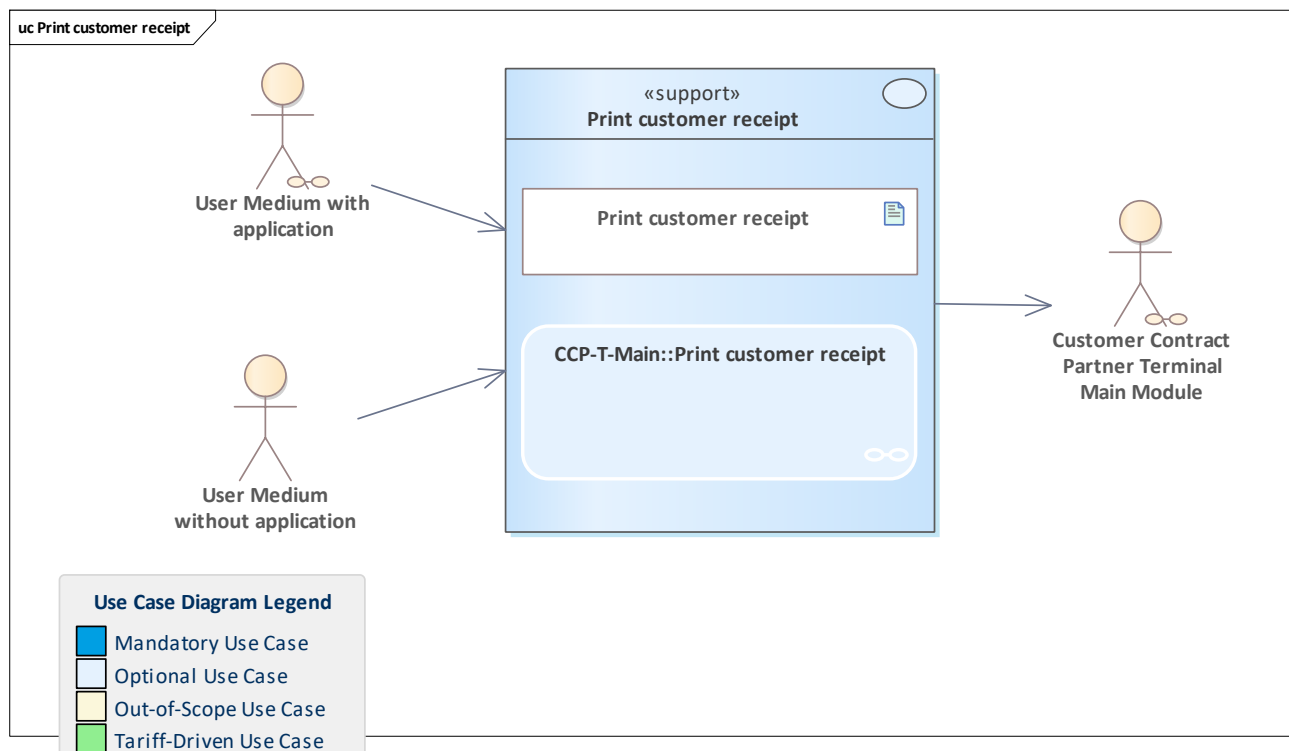


Use Case	<u>Display static entitlement</u>
Description	Details of a static entitlement are shown by customer service in a terminal.
Initiating Actor	<u>User Medium without application</u>
Reacting Actor	<u>Customer Contract Partner Terminal Static Entitlement Module</u>
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	<u>Print customer receipt</u>
Linked Use Cases (Includes)	<u>Check user medium without application</u>
Linked Use Cases (Realises)	
Base Activity	
Inputs	<u>Binary static ticket data</u>



Outputs	
Error Cases	
Activity Diagram	CCP-T-STE::Display static entitlement

21.2.9 Optional: Print customer receipt



Use Case	Print customer receipt
Description	A customer receipt is printed e.g. after selling an entitlement or having entitlements displayed on a terminal.
Initiating Actor	User Medium with application User Medium without application
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases (Extended By)	
Linked Use Cases (Includes)	
Linked Use Cases (Realises)	
Base Activity	
Inputs	List of entitlements : Entitlement
Outputs	
Error Cases	
Activity Diagram	CCP-T-Main::Print customer receipt

22 Miscellaneous Bundle CCP-Terminal

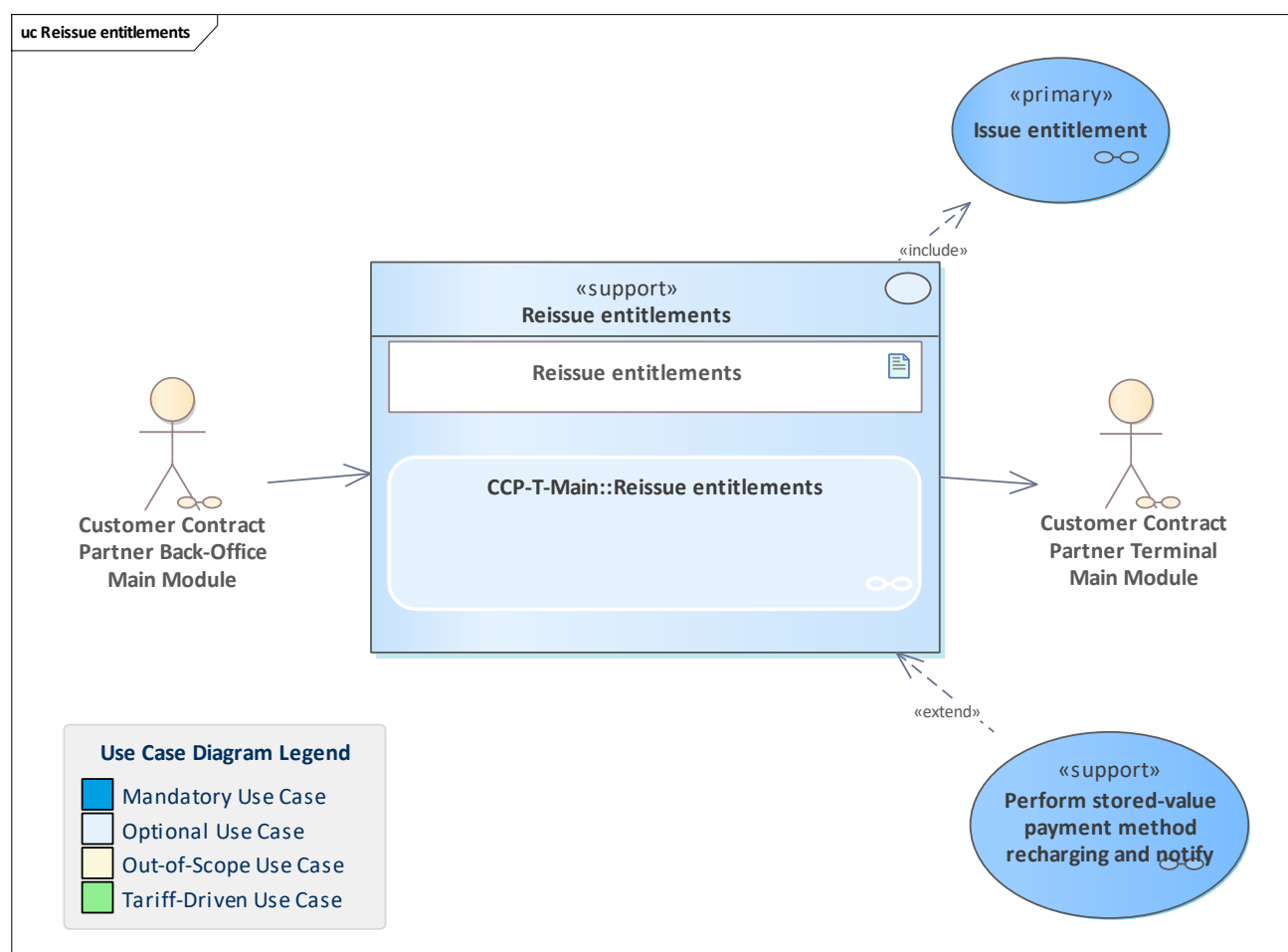
Functionality bundle that covers miscellaneous CCP terminal use cases.

22.1 Overview

Optional: Reissue entitlements

22.2 Use Cases

22.2.1 Optional: Reissue entitlements



Use Case	Reissue entitlements
Description	Reissue entitlements based on existing entitlements. The reason for that might be lost or defective user medium.
Initiating Actor	Customer Contract Partner Back-Office Main Module
Reacting Actor	Customer Contract Partner Terminal Main Module
Preconditions	
Postconditions	
Linked Use Cases	Perform stored-value payment method recharging and notify



(Extended By)	
Linked Use Cases (Includes)	Issue entitlement
Linked Use Cases (Realises)	
Base Activity	
Inputs	Issue entitlements : tIssueEntitlements
Outputs	Terminal issue entitlements response : tIssueEntitlementsResponse
Error Cases	Terminal issue entitlements exception : tIssueEntitlementsException
Activity Diagram	CCP-T-Main::Reissue entitlements

