C. Authorization

1. Introduction

This Functional Block describes all the authorization-related functionalities, it contains different ways of authorizing a user, online and/or offline and the AuthorizeRequest message handling/behavior, Authorization Cache functionality, etc.

When a user wishes to unplug the electric vehicle from the Charging Station, the Charging Station needs to verify that the user is either the one that initiated the charging or that the user is in the same group and thus allowed to terminate the charging. Once authorized, the Charging Station informs the CSMS that the charging has been stopped.

- To improve the experience for users, a Charging Station MAY support local authorization of identifiers, using an Authorization Cache.
- The `LocalAuthorizeOffline` Configuration Variable controls whether a Charging Station will authorize a user when *offline* using the Authorization Cache.
- The `LocalPreAuthorize` Configuration Variable controls whether a Charging Station will use the Authorization Cache to start a transaction without performing an authorization with the CSMS.

1.1. ID Tokens

This section is normative

OCPP now makes it possible to use many different types of authorization. Where OCPP 1.x only supported RFID, OCPP now also supports things like: credit card, PIN-code, a simple start button etc.

An IDTokenType contains the identifier to use for authorization. It is defined as a combination of a case insensitive string and a type. Message data elements of the IDTokenType class (including GroupId) MAY contain any data, that is meaningful to a CSMS (e.g. for the purpose of identifying the initiator of charging activity), and Charging Stations MUST NOT make any presumptions as to the format or content of such data, other than is provided in the description of the IdTokenType (e.g. by assuming that it is a UID-like value that must be hex characters only and/or an even number of digits). IdToken data acquired via local token reader hardware is usually a (4, 7 or 10 bytes) UID value of a physical IdToken, typically represented as 8, 14 or 20 hexadecimal digit characters.

NOTE To promote interoperability, based on common practice to date in the case of IdTokenType data has type: *ISO14443*, it is RECOMMENDED that such UIDs be represented as hex representations of the UID bytes. According to ISO 14443-3, byte 0 should come first in the hex string. (Most significant nibble of byte 0 first)

1.1.1. Additional Info

AdditionalInfo can be used to send extra information which can be validated by the CSMS in addition to the regular authorization with *IdToken*.

AdditionalInfo contains one or more custom types, which need to be agreed upon by all parties involved. When AdditionalInfo is implemented the Charging Station SHALL also cache and include *AdditionalInfo* during regular operations and set the Configuration Variable `AdditionalInfoItemsPerMessage`. When *AdditionalInfo* is NOT implemented or a not supported *AdditionalInfo.type* is used, the CSMS/Charging Station MAY ignore the *AdditionalInfo*.

1.2. Group ID Tokens

This section is normative

A CSMS has the ability to treat a set of identity tokens as a "group", thereby allowing any one token in the group to start a transaction and for the same token, or another token in the same group, to stop the transaction. This supports the common use-cases of families or businesses with multiple drivers using one or more shared electric vehicles on a single recharging contract account. IDTokenTypes used as "GroupId" may often use a shared central account identifier for the GroupId, instead of a UID of the first/master RFID card of an account.

Tokens (idTags) are grouped for authorization purposes by specifying a common group identifier in the optional *groupIdToken* element in IdTokenInfo: two IdTokens are considered to be in the same group if their GroupIdTokens match (and they are not empty).

NOTE Even though the GroupId has the same nominal data type (IdTokenType) as an idToken, the value of this element may not be in the common format of IdTokenTypes and/or may not represent an actual valid IdTokenType (e.g. it may be a common shared "account number"): therefore, the GroupId value SHOULD NOT be used for comparison against a presented Token value (unless it also occurs as an idToken value).

1.3. Authorization Cache

A Charging Station MAY implement an Authorization Cache that **autonomously** maintains a record of previously presented identifiers that have been successfully authorized by the CSMS. The Authorization Cache can be used to speed up the authorization process at the Charging Station, since using a locally stored cache means that the user does not have to wait for the Charging Station to check the authorization at the CSMS. Operation of the Authorization Cache, when present, is reported (and controlled, where possible) by the `AuthCacheEnabled` Configuration Variable. The optional expiration time of general Authorization Cache entries can be set in the Configuration Variable `AuthCacheLifeTime`. If a different expiration time is desired for a specific entry, this can be set in the cacheExpiryDateTime that is returned in iDTokenInfo of, for example, the AuthorizeResponse.

Please refer to the use cases C10 - Store Authorization Data in the Authorization Cache and C12 - Start Transaction - Cached Id for more information on how to implement / use the Authorization Cache functionality.

When a Charging Station supports both the Authorization Cache and Tariff information (see: Tariff & Cost), it should not store the tariff information in the Authorization Cache, since this information could become outdated.
A Charging Station MAY support the authorization of *any* presented identifier when *offline*, to avoid refusal of charging to bona fide users that cannot be explicitly authorized by Authorization Cache entries. This functionality is explained in more detail in Unknown Offline Authorization.
It is RECOMMENDED to store personal information in the Authorization Cache securely, e.g. by only storing hashed idTokens in the cache.

1.4. Local Authorization List

The Local Authorization List is a list of identifiers that can be synchronized with the CSMS. It allows authorization of a user when offline and faster (apparent) authorization response time when communication between Charging Station and CSMS is slow. The CSMS can synchronize the list by either sending a complete list of identifiers to replace the Local Authorization List or by sending a list of changes (add, update, delete) to apply to the Local Authorization List. The operations to support this are GetLocalListVersion and SendLocalList.

This list contains the authorization status of all (or a selection of) identifiers and the corresponding expiration date. These values may be used to provide more fine grained information to users (e.g. by display message) during local authorization.

Please refer to the use cases D01 - Send Local Authorization List, C13 - Offline Authorization through Local Authorization List and C14 - Online Authorization through Local Authorization List for more information on how to implement / use the Local Authorization List functionality.

NOTE	Please note the difference between the Authorization Cache and Local Authorization List mechanisms: the Authorization Cache is an autonomous mechanism at the Charging Station, whereas the Local Authorization List is a list that is synchronized between CSMS and Charging Station (originating from the CSMS).

NOTE	The Authorization Cache and Local Authorization List are **distinct** logical data structures. When both Authorization Cache as well as Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.

The following Configuration Variables are used by the Charging Station to give information about the Local Authorization List

- `LocalAuthListEntries` (Also reports the maximum amount of IdTokens in the Local Authorization List)
- `LocalAuthListEnabled`
- `LocalAuthListAvailable`
- `ItemsPerMessageSendLocalList`
- `BytesPerMessageSendLocalList`

1.5. Unknown Offline Authorization

When *offline*, a Charging Station MAY allow automatic authorization of any "unknown" identifiers that are not found in the Local Authorization List and/or Authorization Cache. Operation of the Unknown Offline Authorization capability, when supported, is reported (and controlled, where possible) by the `OfflineTxForUnknownIdEnabled` Configuration Variable.
When connection to the CSMS is restored, the Charging Station has to send the queued TransactionEventRequest messages. These may contain transactions that were authorized *offline*, as explained in transaction-related message handling. Please refer to C15 - Unknown Offline Authorization for the options that the Charging Station has to continue / stop the transaction in this

situation.

2. Use cases & Requirements

2.1. Authorization options

C01 - EV Driver Authorization using RFID

Table 59. C01 - EV Driver Authorization using RFID

No.	Type	Description
1	**Name**	EV Driver Authorization using RFID
2	**ID**	C01
	Functional block	C. Authorization
3	**Objective(s)**	To enable the Charging Station to request the CSMS to authorize an EV Driver to start or stop charging.
4	**Description**	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first before the charging can be started or stopped.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver wants to start or stop charging the EV and presents an RFID card. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message indicates whether or not the IdToken is accepted by the CSMS.
	Alternative scenario(s)	C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver is authorized and can start *or* stop charging. **Failure postcondition:** If the authorize message is *Invalid*, *Blocked*, *Expired* or *Unknown*, the EV Driver can *not* start or stop charging, except in the case where the EV Driver presents the same token used to start the transaction.

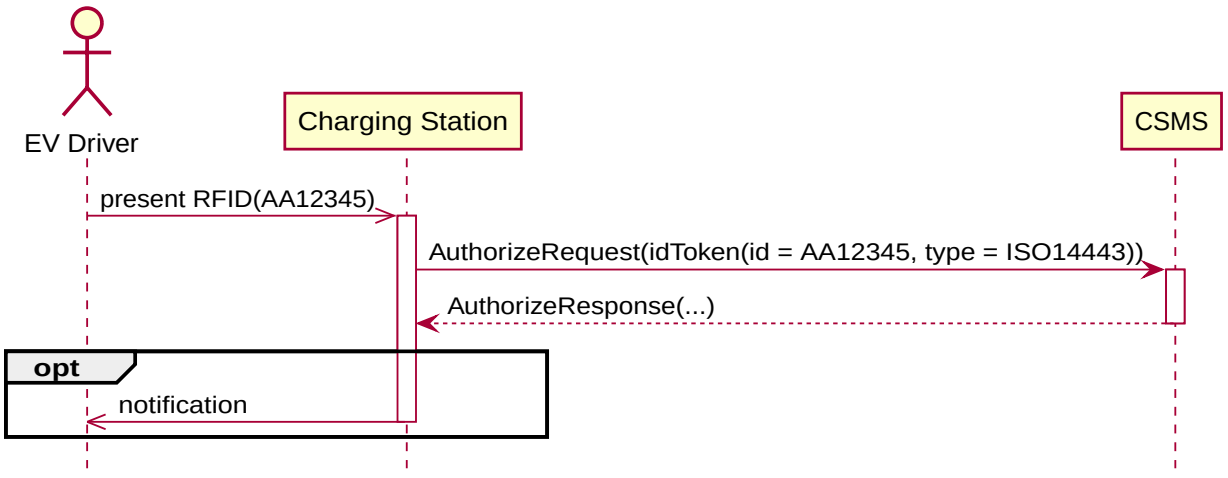

Figure 22. Sequence Diagram: EV Driver Authorization

7	**Error handling**	When the Authorization is not 'Accepted', the AuthorizeResponse contains an authorization status value indicating the reason for rejection.

8	**Remark(s)**	Assuming *idToken* is valid for charging and the Charging Station has 3 EVSEs, what is the content of *idTokenInfo*, when *idToken* is allowed to charge: . at all EVES: *idTokenInfo.status* = `Accepted`. . at EVSE 1: *idTokenInfo.status* = `Accepted`, *idTokenInfo.evseId* = [1]. . at EVSE 1 + 2: *idTokenInfo.status* = `Accepted`, *idTokenInfo.evseId* = [1, 2]. . at none of the EVSEs: _idTokenInfo.status=`NotAtThisLocation`.

C01 - EV Driver Authorization using RFID - Requirements

Table 60. C01 - Requirements

ID	Precondition	Requirement definition	Note
C01.FR.01	Configuration setting AuthEnabled is true.	The Charging Station SHALL only offer energy after authorization.	
C01.FR.02	If an idToken presented by the EV Driver is not present in the Local Authorization List or Authorization Cache	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.	
C01.FR.03	When stopping a transaction	The Charging Station SHALL NOT send an AuthorizeRequest when (a) the IdToken used for stopping the transaction is the same as the IdToken that started the transaction OR (b) when the IdToken used for stopping the transaction is in the Local Authorization List or the Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the transaction.	
C01.FR.04		AuthorizeRequest SHALL only be used for the authorization of an identifier.	
C01.FR.05	If an IdToken is present in the Local Authorization List or Authorization Cache.	The Charging Station MAY send AuthorizeRequest to the CSMS.	
C01.FR.06		AuthorizeResponse sent by the CSMS to a Charging Station MAY include groupIdToken.	
C01.FR.07		AuthorizeResponse SHALL include an authorization status value indicating acceptance or a reason for rejection.	See AuthorizationStatusEnumType for the possible reasons of rejection.
C01.FR.08	If the field: language1 is set AND the Charging Station contains messages in that *language*.	The Charging Station SHALL show messages to the user in **language1**.	
C01.FR.09	If the field: language1 is set AND the Charging Station does not contain messages in that *language* AND if the field: language2 is set AND the Charging Station contains messages in that *language*	The Charging Station SHALL show messages to the user in **language2**.	
C01.FR.10	If the field: language1 is not set	The field: language2 SHALL NOT be set.	
C01.FR.11		Field: language1 SHALL be different from field language2.	
C01.FR.12		It is RECOMMENDED to implement messages in **English** as fall-back.	
C01.FR.13	If both language1 AND language2 don't match installed languages in the Charging Station	It is RECOMMENDED to show messages to the EV Driver in **English**.	
C01.FR.17		Language SHALL be specified as RFC-4646 tags, see: [RFC5646], example: US English is: "en-US".	
C01.FR.18	If the IdToken is valid AND the EV driver is NOT allowed to charge at the type of EVSE(s) this Charging Station provides.	The CSMS SHALL send an AuthorizeResponse with idTokenInfo.status *NotAllowedTypeEVSE*.	
C01.FR.19	*idToken* is allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *idTokenInfo* has an empty (or absent) *evseId* list.	This will be the most common case. Even though the *idToken* might be allowed on any EVSE, the *idTokenInfo.status* still needs to be `Accepted` before charging is allowed.

ID	Precondition	Requirement definition	Note
C01.FR.20	*idToken* is allowed for a subset of EVSEs of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *IdTokenInfo* has an *evseId* list with the allowed EVSEs.	Note the difference between validity of an *idToken* and the fact whether this (type of) token is allowed on an EVSE. The *idTokenInfo.status* still needs to be `Accepted` before charging is allowed.
C01.FR.21	C01.FR.20	The Charging Station SHALL only allow charging on the EVSEs mentioned in the AuthorizeResponse.	
C01.FR.22	*idToken* is not allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *idTokenInfo.status* is `NotAtThisLocation` and *evseId* list is empty (or absent).	Status `NotAtThisLocation` needed in order to differentiate with the situation in which *idToken* is allowed on all EVSEs.

C02 - Authorization using a start button

Table 61. C02 - Authorization using a start button

No.	Type	Description
1	**Name**	Authorization using a start button
2	**ID**	C02
	Functional block	C. Authorization
3	**Objectives**	Make it possible for a Charging Station that has a start button to start charging.
4	**Description**	For some chargers authorization of a user might not be a requirement. A simple charger might have a button instead of a more expensive RFID reader to start charging. When such a Charging Station start charging, it is not needed to send an AuthorizeRequest. In the TransactionEventRequest (eventType = Started), IdTokenType information needs to be given, which the CSMS then cannot reject.
	Actors	EV Driver, Charging Station, CSMS
	Scenario description	**1.** The EV Driver plugs in the charging cable between EV and Charging Station. **2.** The Charging Station sends a StatusNotificationRequest and TransactionEventRequest (eventType = Started) to notify the CSMS about the cable being plugged in. **3.** The EV Driver presses the start button to start Charging. **4.** The Charging Station starts Charging of the EV. **5.** The Charging Station sends a TransactionEventRequest (eventType = Updated) message with IdTokenEnumType: *NoAuthorization* to the CSMS to notify the CSMS of the charging that has started. **6.** Upon receipt of TransactionEventRequest (eventType = Updated), the CSMS responds with TransactionEventResponse with: IdTokenInfo.status set to *Accepted*
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Charging Station has a start button, instead of an RFID reader to start charging of an EV.
6	**Postcondition(s)**	Transaction ongoing on Charging Station, CSMS is aware of transaction.

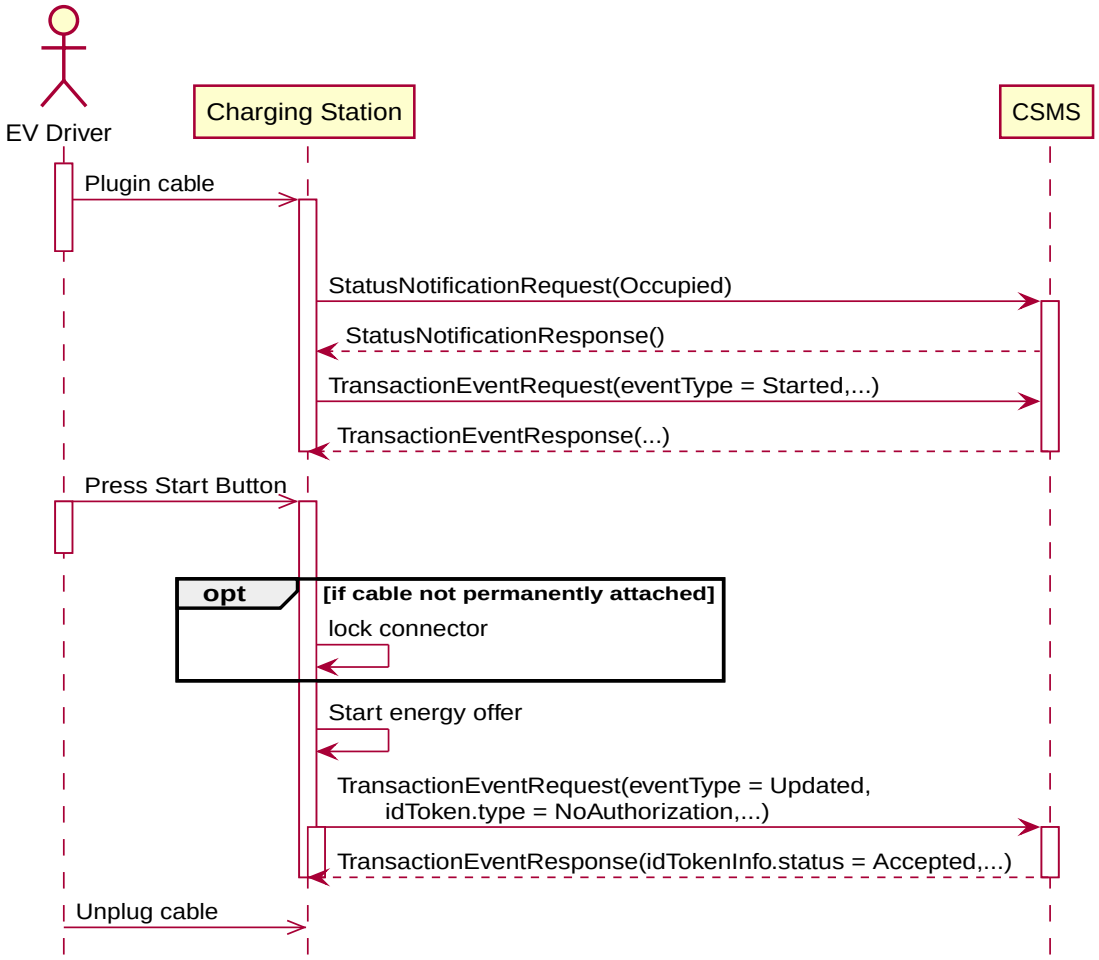

Figure 23. Sequence Diagram: Authorization using a start button

7	**Error Handling**	n/a
8	**Remarks**	The start button might also be a mechanical key or something similar. Note that the start button can even be omitted if the Charging Station is configured to start charging upon cable connection. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C02 - Authorization using a start button - Requirements

Table 62. C02 - Authorization using a start button - Requirements

ID.	Precondition	Requirement definition
C02.FR.01	When a transaction is started with a button.	The Charging Station SHALL send TransactionEventRequest with an IdTokenType of type: NoAuthorization and the field: idToken left empty (empty string).
C02.FR.02	CSMS receives a TransactionEventRequest with an IdTokenType of type: NoAuthorization	The CSMS SHALL respond with a TransactionEventResponse with IdTokenInfo.status set Accepted.
C02.FR.03	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type NoAuthorization in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.

C03 - Authorization using credit/debit card

Table 63. C03 - Authorization using credit/debit card

No.	Type	Description
1	**Name**	Authorization using credit card
2	**ID**	C03
	Functional block	C. Authorization
3	**Objectives**	Make it possible to start a transaction using a credit card.
4	**Description**	A Charging Station with a credit/debit card terminal built inside the housing, or belonging to a group of Charging Stations that has a central payment terminal/kiosk. An EV Driver uses his card to pay for charging. The transaction is authorized by the payment company, the CSMS receives a message from the Payment System, and send a RequestStartTransactionRequest to the Charging Station to start the transaction.
	Actors	EV Driver, Payment System, CSMS, Charging Station
	Scenario description	**1.** The EV Driver plugs in the Charging Cable **2.** The Charging Station sends an StatusNotificationRequest and TransactionEventRequest (eventType = Started) to notify the CSMS about the cable being plugged in. **3.** The Driver uses the credit/debit card terminal to authorize/pay for charging. **4.** The terminal communicates with its own server/back-office. **5.** The Payment System sends a message to the CSMS authorizing the user. **6.** The CSMS generates a unique id to be used as IdToken for this transaction. **7.** The CSMS sends a RequestStartTransactionRequest with the generated IdToken to the Charging Station. **8.** The Charging Station accepts the RequestStartTransactionRequest by sending a RequestStartTransactionResponse with Accepted. **9.** The Charging Station start Charging of the EV. **10.** The Charging Station send an TransactionEventRequest (eventType = Updated) to notify the CSMS about the charging having started.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Charging Station has a credit/debit card terminal, or belongs to a group of Charging Stations that has a central payment terminal, to start charging of an EV.
6	**Postcondition(s)**	Transaction ongoing on Charging Station

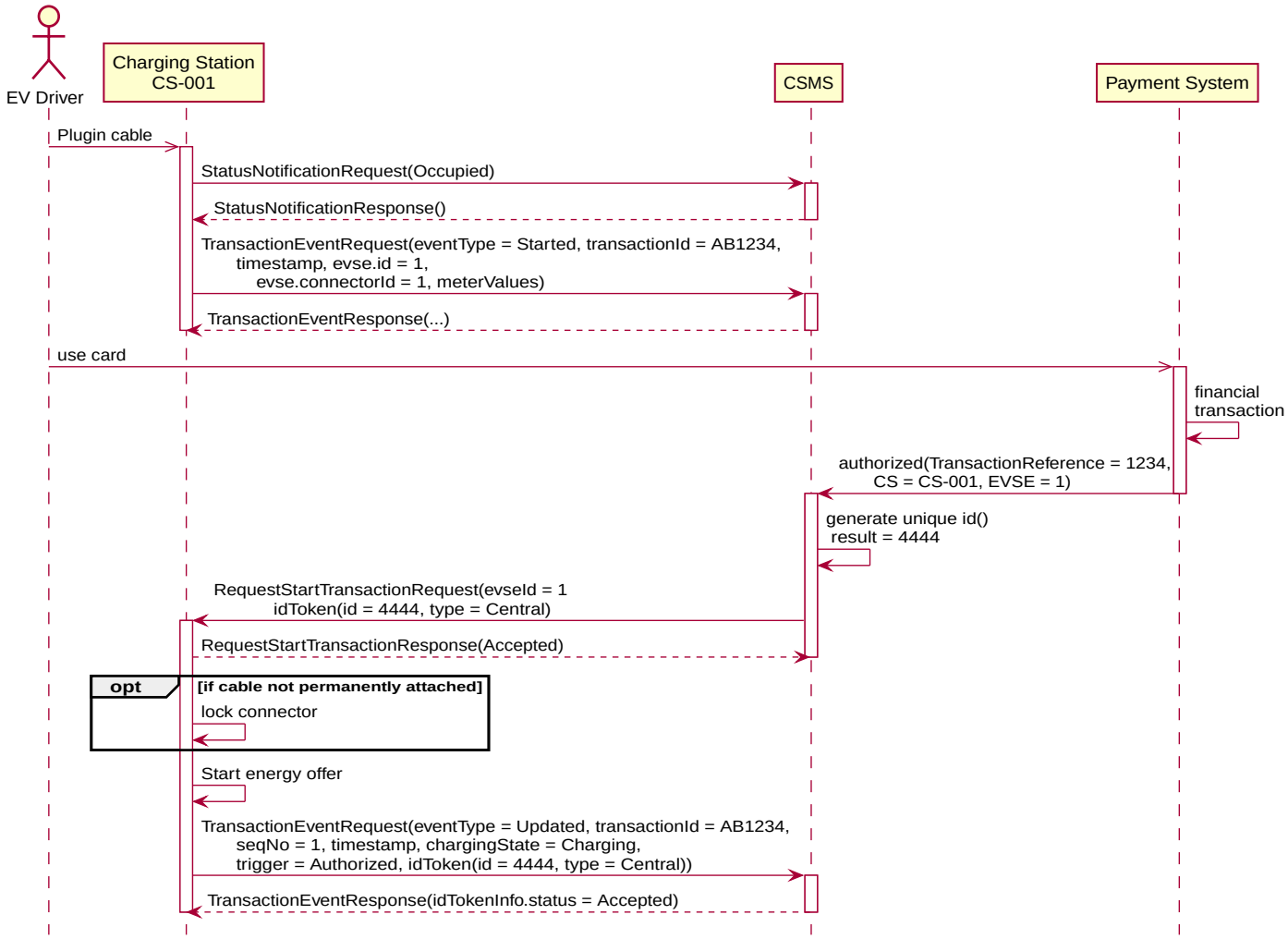

Figure 24. Sequence Diagram: Authorization using credit/debit card

7	**Error Handling**	n/a
8	**Remarks**	This use case is an example of how the existing OCPP messages can be used to handle a transaction that is started with a credit/debit card, it is not required to implement a credit/debit card payment solution in this way. A Payment System may consist of multiple components handling the authorization of the user. The interface of these components and the communication between the Payment System and CSMS are not in scope of this document. Stopping a transaction started with a credit/debit card is not defined, this is left to the implementer, this could for example be: Unplugging the cable on the EV side and/or a stop button etc. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C03 - Authorization using credit/debit card - Requirements

Table 64. C03 - Authorization using credit/debit card - Requirements

ID.	Precondition	Requirement definition
C03.FR.01	If the Charging Station receives a RequestStartTransactionRequest with an IdTokenType of type Central	The Charging Station SHALL NOT send an AuthorizeRequest for the received IdTokenType.

ID.	Precondition	Requirement definition
C03.FR.02	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type Central in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.

C04 - Authorization using PIN-code

This is an informative use case, its purpose is to demonstrate the use of the KeyCode id type. An other use of KeyCode is for example a licence plate number.

Table 65. C04 - Authorization using PIN-code

No.	Type	Description
1	**Name**	Authorization using PIN-code
2	**ID**	C04
	Functional block	C. Authorization
3	**Objectives**	To make it possible for a Charging Station that has a key entry terminal to authorize the PIN-code.
4	**Description**	When a Charging Station has a PIN-code entry terminal, an EV driver enters his/her PIN-code. This PIN-code is sent to the CSMS for validation using an AuthorizeRequest.
	Actors	EV Driver, Charging Station, CSMS
	Scenario description	**1.** The EV Driver wants to start or stop charging the EV and enters his/her PIN-code into the terminal. **2.** The Charging Station sends an AuthorizeRequest message, with the field: IdTokenEnumType set to *KeyCode*, to the CSMS to request authorization. **3.** Upon receipt of the AuthorizeRequest, the CSMS responds with an AuthorizeResponse. This response indicates whether or not the KeyCode is accepted by the CSMS.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Charging Station has a PIN-code entry terminal to start charging of an EV.
6	**Postcondition(s)**	Transaction ongoing on Charging Station, CSMS is aware of transaction.

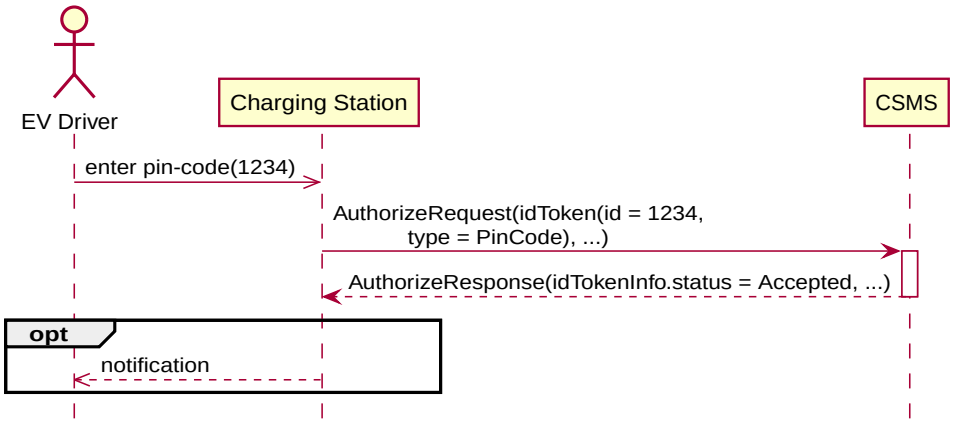

Figure 25. Sequence Diagram: Authorization using PIN-code

7	**Error Handling**	n/a
8	**Remarks**	When the PIN-code is validated in the Charging Station, instead of the CSMS, use case C02 - Authorization Using a Start button applies.

C04 - Authorization using PIN-code - Requirements

Table 66. C04 - Authorization using PIN-code - Requirements

ID.	Precondition	Requirement definition
C04.FR.01	When the CSMS receives an AuthorizeRequest with a keyCode that is not valid at this Charging Station	The CSMS SHALL respond with an AuthorizeResponse message with status = Invalid.

ID.	Precondition	Requirement definition
C04.FR.02	When the CSMS receives an AuthorizeRequest with a keyCode that is valid and the EV Driver is allowed to charge at this Charging Station	The CSMS SHALL respond with an AuthorizeResponse message with status = Accepted.
C04.FR.03		A Charging Station MAY store keyCodes in the Authorization Cache.
C04.FR.04	If an idToken of type keyCode is used	The Charging Station or CSMS SHALL NOT show the IdToken in any logging. key codes should never appear in logs.
C04.FR.05		Language SHALL be specified as RFC-5646 tags, see: [RFC5646], for example: US English is: "en-US".
C04.FR.06	If an idToken of type keyCode is used	It is RECOMMENDED to take measures to prevent brute force attacks, for example by increasing backoff times after attempts to enter an incorrect keyCode.

C05 - Authorization for CSMS initiated transactions

Table 67. C05 - Authorization for CSMS initiated transactions

No.	Type	Description
1	**Name**	Authorization for CSMS initiated transactions
2	**ID**	C05
	Functional block	C. Authorization
3	**Objectives**	Enable the CSMS to start a transaction on a Charging Station with a server generated IdToken.
4	**Description**	When a CSMS needs to start a Transaction on a Charging Station for a Driver that has no RFID, or the RFID is not known. For Example, the EV Driver uses an App to start a transaction. The CSMS needs to determine an IdToken and tell the Charging Station this is not an RFID, so it should not be cached and an authorization is also not needed.
	Actors	EV Driver, CSMS, Charging Station
	Scenario description	**1.** The EV Driver uses his app to start a charging. **2.** The app sends a start request to the CSMS. **3.** The CSMS determines an IdToken. It can generate a unique id to be used as IdToken for this transaction or can use a token that is provided by the app (for example the ID of the contract of the user). **4.** The CSMS sends a RequestStartTransactionRequest with the IdToken from the previous step to the Charging Station. **5.** The Charging Station accepts the RequestStartTransactionRequest by sending a RequestStartTransactionResponse with Accepted. **6.** The Charging Station sends a StatusNotificationRequest. **7.** The Charging Station sends a TransactionEventRequest (eventType = Updated) to notify the CSMS about the cable being plugged in.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Cable is plugged in.
6	**Postcondition(s)**	Transaction ongoing on Charging Station

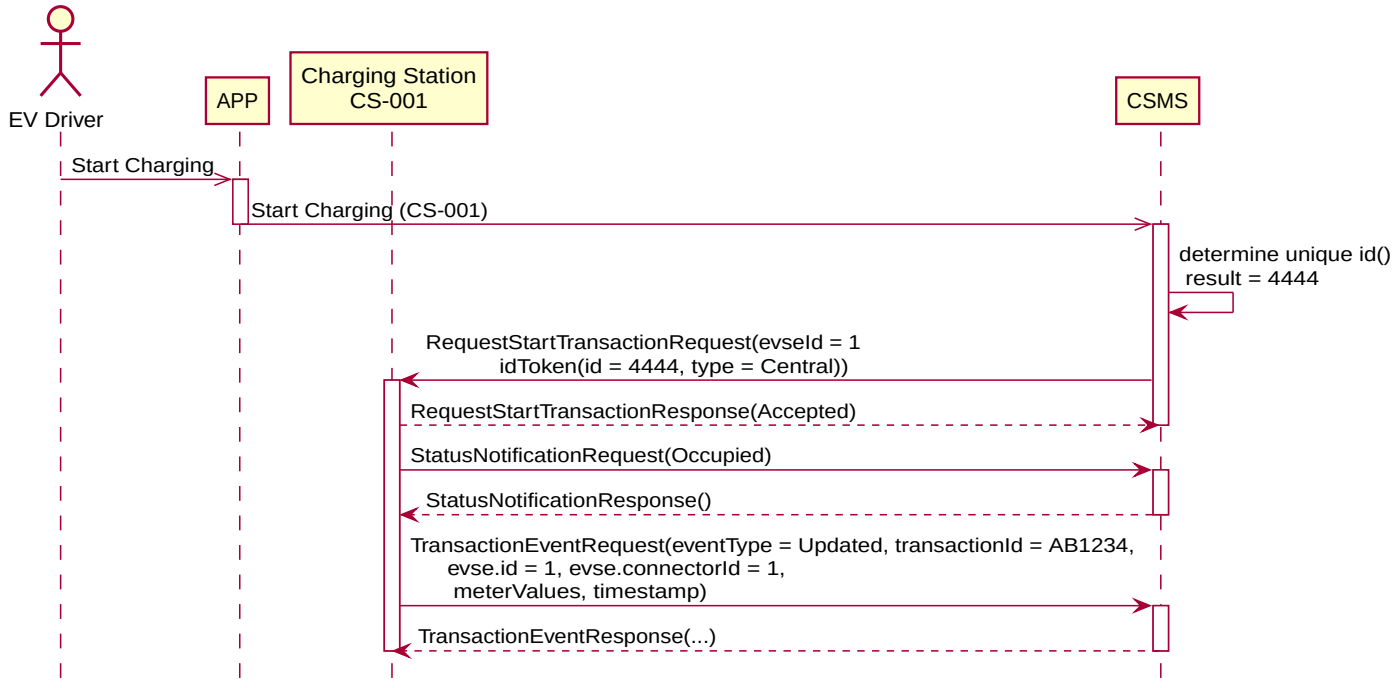

Figure 26. Sequence Diagram: Authorization for CSMS initiated transactions

7	**Error Handling**	n/a
8	**Remarks**	IdTokens MAY be (single use) virtual transaction authorization codes or virtual RFID tokens that deliberately use a non-standard UID format to avoid possible conflict with real UID values. These virtual single use IdTokens are sent with type *Central* and it is pointless to either cache or authorize these tokens. This use case uses an App as example, but this is not a requirement. This use case is valid for any RequestStartTransactionRequest with a server generated IdToken. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C05 - Authorization for CSMS initiated transactions Requirements

Table 68. C05 - Authorization for CSMS initiated transactions Requirements

ID.	Precondition	Requirement definition
C05.FR.01	If the Charging Station receives a RequestStartTransactionRequest with an IdTokenType of type Central.	The Charging Station SHALL NOT send an AuthorizeRequest for the received IdTokenType.
C05.FR.02	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type Central in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.
C05.FR.03		The RemoteStartId SHALL be provided at least once in a TransactionEventRequest.
C05.FR.04		Language SHALL be specified as RFC-4646 tags, see: [RFC5646], example: US English is: "en-US".
C05.FR.05		idToken SHALL also be provided once in the first TransactionEventRequest after a RequestStartTransactionRequest.

C06 - Authorization using local id type

This is an informative use case, its purpose is to demonstrate the use of the Local id type.

Table 69. C06 - Authorization using local id type

No.	Type	Description
1	**Name**	Authorization using local id type
2	**ID**	C06
	Functional block	C. Authorization
3	**Objectives**	Enable the Charging Station to start charging with a locally generated IdToken.
4	**Description**	When a Charging Station needs to start a Transaction for a Driver that has no RFID, or the RFID is not known. For Example, the EV Driver uses a parking ticket to start charging.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	**1.** An EV driver drives into a garage, takes a parking ticket at the barrier at the entrance. **2.** Parks his EV at a Charging Station. **3.** Plugs in the charging cable. **4.** Scans/inserts his parking ticket on the Charging Station to start Charging **5.** EV is charging, driver leaves. **6.** EV driver returns, inserts parking ticket into a payment kiosk **7.** Pays for parking and charging **8.** The Payment terminal/kiosk sends a stop command via the CSMS to the Charging Station. **9.** EV driver unplugs the charging cable and drives away.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Integrated parking & charging payment system
6	**Postcondition(s)**	The transaction has completed at the Charging Station and Transaction information is available at the CSMS.

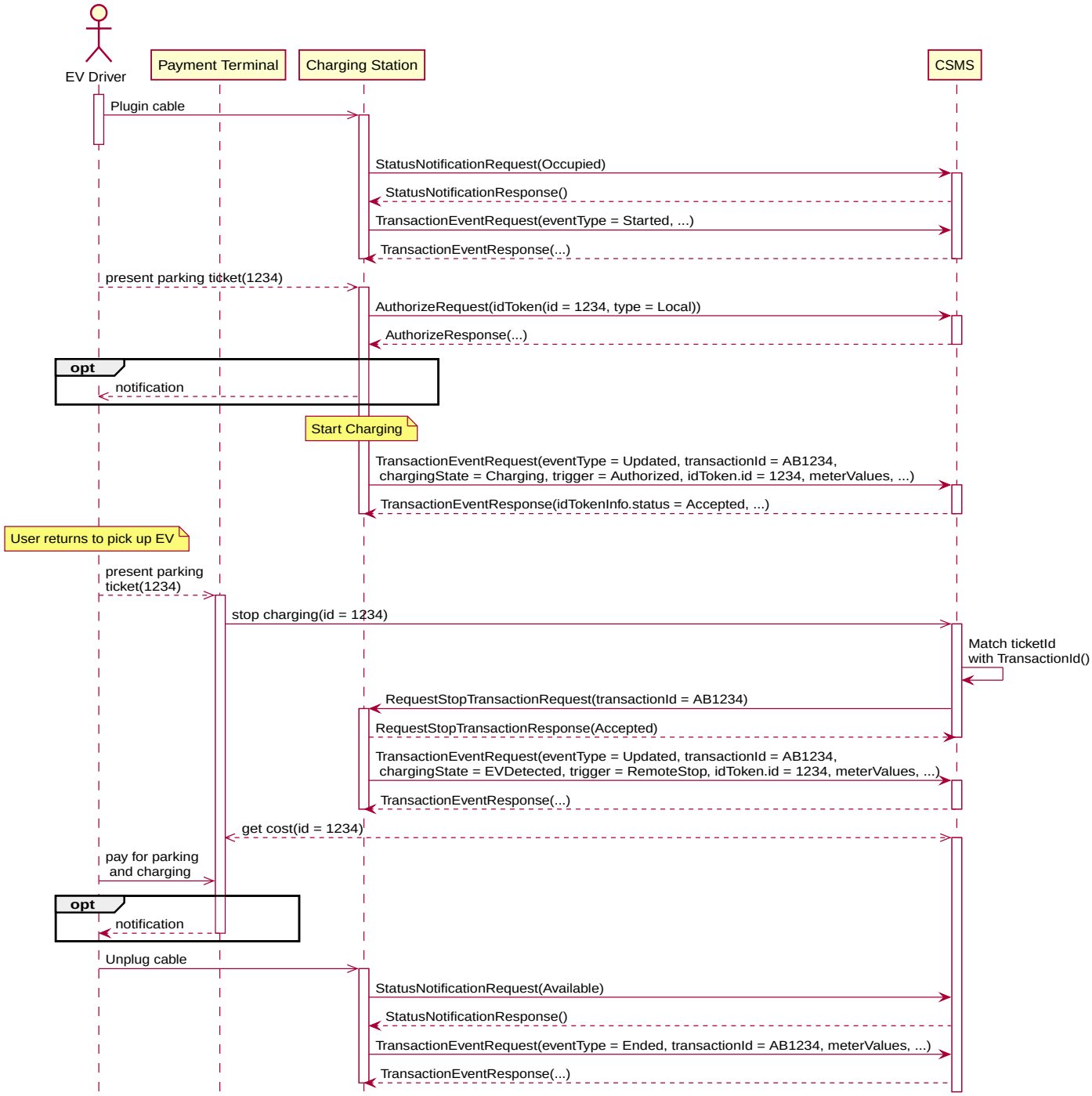

Figure 27. Sequence Diagram: Authorization using local id type

7	**Error Handling**	n/a
8	**Remarks**	This use case uses an Parking Ticket as example, but this is not a requirement. The communication between the Payment Terminal and the CSMS is outside of scope of OCPP. The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: `TxStartPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer `TxStopPoint`: ParkingBayOccupancy, EVConnected This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.

C06 - Authorization using local id type - Requirements

Table 70. C06 - Authorization using local id type - Requirements

ID	Precondition	Requirement definition
C06.FR.01		The Charging Station SHALL only offer energy after authorization.
C06.FR.02	If an IdTokenType with type Local is presented by the EV Driver.	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.
C06.FR.03		AuthorizeRequest SHOULD only be used for the authorization of an identifier for charging.
C06.FR.04	If the CSMS receives an AuthorizeRequest.	it SHALL respond with an AuthorizeResponse and SHALL include an authorization status value indicating acceptance or a reason for rejection.

2.2. ISO 15118 Authorization

This authorization section originates from ISO15118-1 for the use of Plug & Charge functionalities.

C07 - Authorization using Contract Certificates

Table 71. C07 - Authorization using Contract Certificates

No.	Type	Description
1	**Name**	Authorization using Contract Certificates
2	**ID**	C07
	Functional block	C. Authorization
	Reference	ISO15118-1 D2
3	**Objectives**	See ISO15118-1, use case Objective D2, page 26.
4	**Description**	See ISO15118-1, use case Description D2 (first bullet), page 26.
	Actors	Actors: EV, Charging Station, CSMS, OCSP
	Scenario description	**15118**: See ISO15118-1, use case Description D2, Scenario Description, first 2 bullets, page 26. **OCPP**: **3.** The Charging Station sends an AuthorizeRequest message to the CSMS containing the eMAID and data needed for an OCSP request with regards to the contract certificate and certificate chain. **4.** The CSMS replies with an agreement or non-agreement, and the certificate status. **5.** Service starts after successful authorization of the IDs.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	A contract Certificate is installed in the EV.
6	**Postcondition(s)**	The validity of the Contract Certificate is determined.

Figure 28. Authorization using Contract Certificates

7	**Error handling**	
8	**Remark(s)**	In edition 1 of 15118, the message timeout of the PaymentDetailsReq/Res message is 5 seconds. In case certificate verification cannot be completed in that time it is possible to complete this during the AuthorizationReq/Res, which can be extended up to 60 seconds. When the Charging Station is offline, it is recommended to omit the payment option for ISO 15118 contract certificates from the ServiceDiscoveryRes and revert to External Identification Means (use case C08), because certificate status cannot be checked.

C07 - Authorization using Contract Certificates - Requirements

Table 72. C07 - Requirements

ID	Precondition	Requirement definition
C07.FR.01	When Charging Station is online	The Charging Station SHALL send an AuthorizeRequest to the CSMS for validation.
C07.FR.02	C07.FR.01	The AuthorizeRequest SHALL contain the eMAID and data needed for an OCSP request with regards to the contract certificate and certificate chain.
C07.FR.04	If the CSMS receives an AuthorizeRequest.	It SHALL respond with an AuthorizeResponse and SHALL include an authorization status value indicating acceptance or a reason for rejection.
C07.FR.05	C07.FR.02	The CSMS SHALL verify validity of the certificate and certificate chain via real-time or cached OCSP data using the hash data provided in *iso15118CertificateHashData* field.
C07.FR.06	C07.FR.01 AND If Charging Station is not able to validate a contract certificate, because it does not have the associated root certificate AND `CentralContractValidationAllowed` is *true*	The Charging Station SHALL pass the contract certificate to the CSMS in *certificate* attribute (in PEM format) of AuthorizeRequest for validation by CSMS.
C07.FR.07	When Charging Station is offline AND `ContractValidationOffline` is *false*	The Charging Station SHALL NOT allow charging.
C07.FR.08	When Charging Station is offline AND `ContractValidationOffline` is *true*	The Charging Station SHALL try to validate the contract certificate locally.

ID	Precondition	Requirement definition
C07.FR.09	C07.FR.08 AND Contract certificate is valid AND `LocalAuthorizeOffline` is *true*	The Charging Station SHALL lookup the eMAID in Local Authorization List or Authorization Cache.
C07.FR.10	C07.FR.09 AND eMAID found in Local Authorization List	The Charging Station SHALL behave according to use case C13 - Offline Authorization through Local Authorization List.
C07.FR.11	C07.FR.09 AND eMAID found in Authorization Cache	The Charging Station SHALL behave according to use case C12 - Start Transaction - Cached Id.
C07.FR.12	C07.FR.09 AND eMAID is not found AND `OfflineTxForUnknownIdEnabled` = *true*	The Charging Station SHALL allow charging according to use case C15 - Offline Authorization of unknown Id.

C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)

Table 73. C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)

No.	Type	Description
1	**Name**	Authorization at EVSE using ISO 15118 External Identification Means (EIM)
2	**ID**	C08 / 15118-1 D4
	Functional block	C. Authorization
	Reference	ISO15118-1 D4
3	**Objectives**	To authorize the EV via the Charging Station, with help of the CSMS. Also see ISO15118-1, use case Objective D4, page 28.
4	**Description**	The Charging Station sends an AuthorizeRequest message based on information provided by the EV. Also see ISO15118-1, use case Description D4 up to and including "NOTE", page 28.
	Actors	Actors: EV, Charging Station, CSMS
	Scenario description	**15118** See ISO15118-1, use case Description (Scenarion Description) D4, page 28. **OCPP** **1.** The Charging Station sends an AuthorizeRequest with an idToken containing the External Identification Means (EIM). **2.** The CSMS responds with an AuthorizeResponse.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C15 - Unknown Offline Authorization
5	**Prerequisites**	Communication between EV and EVSE SHALL be established successfully.
6	**Postcondition(s)**	Authorization is successful. Also see ISO15118-1, use case End conditions D4, page 28.

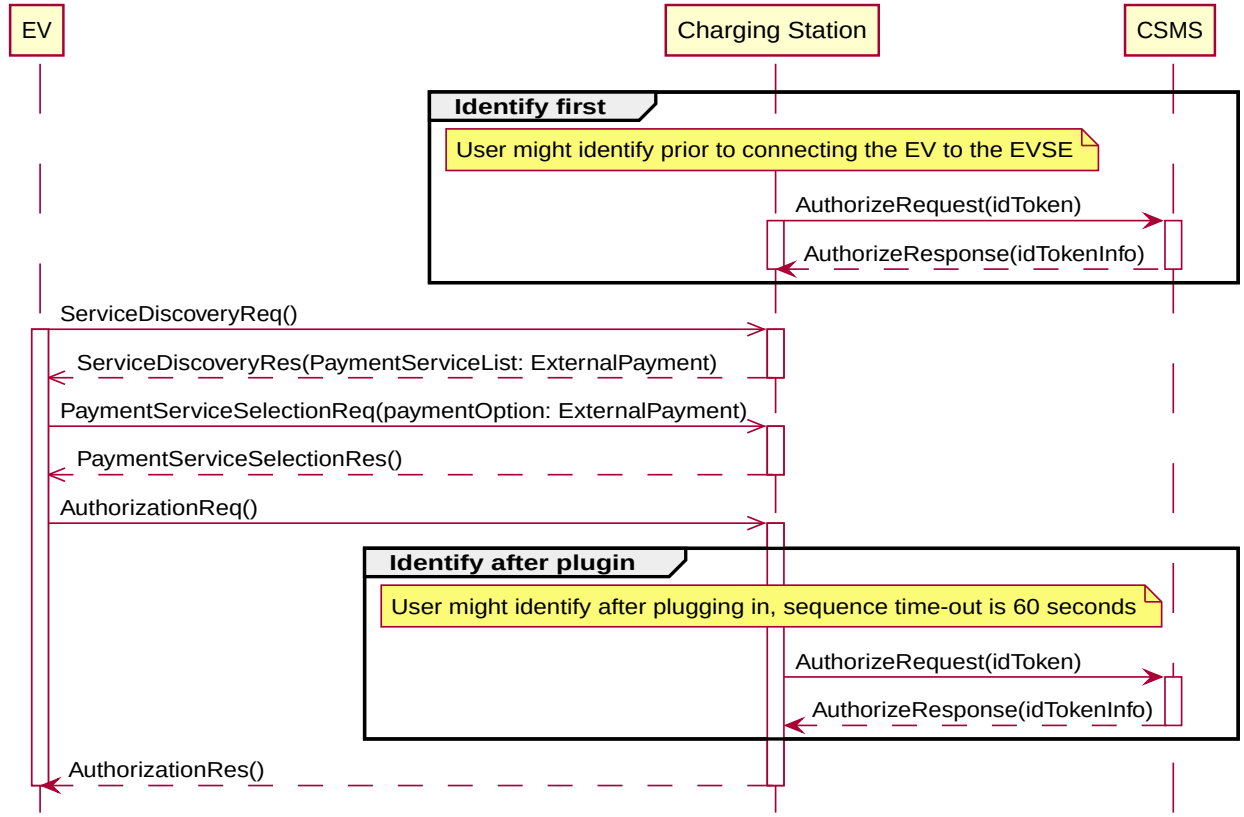

Figure 29. Sequence Diagram: Authorization at EVSE using external credentials performed with help of SA.

7	**Remark(s)**	Please note that all identification means mentioned in the previous section can be applied to this use case. The only difference is the availability of 15118 communication.

Source: ISO15118-1

C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) - Requirements

Table 74. C08 - Requirements

ID	Precondition	Requirement definition
C08.FR.01		The Charging Station SHALL send the identification to the CSMS for validation.
C08.FR.02		EV Driver SHALL activate the authorization within a specific time after connecting the EV to the EVSE or the EVSE SHALL have an HMI to authorize the restart of the identification process.

2.3. GroupId

C09 - Authorization by GroupId

Table 75. C09 - Authorization by GroupId

No.	Type	Description
1	**Name**	Authorization by GroupId
2	**ID**	C09
	Functional block	C. Authorization
3	**Objective(s)**	To enable 2 EV drivers with different IdTokens to be authorized using the same GroupId.
4	**Description**	This use cases covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).
	Actors	Charging Station, CSMS, EV Driver1, EV Driver2
	Scenario description	**1.** EV Driver 1 presents an IdToken. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId. **4.** The Charging Station stores the GroupIdToken with the authorization information of EV Driver 1. **5.** EV Driver 2 presents an IdToken. **6.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **7.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId. **8.** Based on the matching GroupId information in both responses, the Charging Station authorizes the action.
5	**Prerequisite(s)**	EV Driver 1 and EV Driver 2 have the same GroupId.
6	**Postcondition(s)**	GroupId is known by the Charging Station.

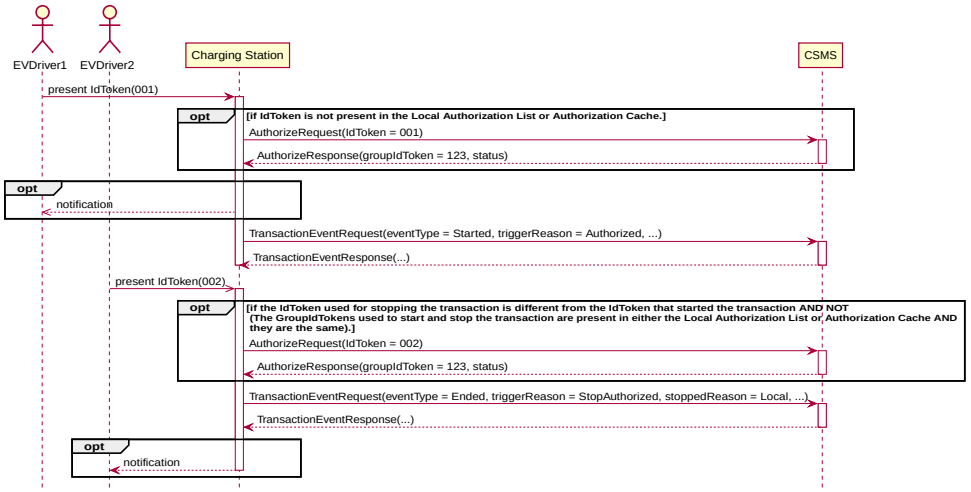

Figure 30. Sequence Diagram: Authorization by GroupId

7	**Error handling**	n/a
8	**Remark(s)**	idTokenType data used as groupId may often use a shared central account identifier for the GroupId, instead of using one of the idTokens belonging to an account. The groupId mechanism as described in this use case also works when using the Authorization Cache, as the groupId is stored in the cache.

C09 - Authorization by GroupId - Requirements

Table 76. C09 - Requirements

ID	Precondition	Requirement definition
C09.FR.02		IdTokens that are part of the same group for authorization purposes SHALL have a common group identifier in the optional *groupIdToken* element in IdTokenInfo
C09.FR.03	When a transaction has been authorized/started with a certain IdToken.	An EV Driver with a different, valid IdToken, but with the same groupIdToken SHALL be authorized to stop the transaction.
C09.FR.04	C09.FR.03 AND If both IdTokens with their corresponding GroupIdTokens are present in either the Local Authorization List or Authorization Cache.	The Charging Station MAY send an AuthorizeRequest to the CSMS.
C09.FR.05	C09.FR.03 AND If NOT both IdTokens with their corresponding GroupIdTokens are present in either the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest to the CSMS.
C09.FR.06	If an idToken presented by the EV Driver is not present in the Local Authorization List or Authorization Cache	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.
C09.FR.07	C09.FR.03	The Charging Station SHALL NOT send an AuthorizeRequest when (a) the IdToken used for stopping the transaction is the same as the IdToken that started the transaction OR (b) when the IdToken used for stopping the transaction is in the Local Authorization List or the Authorization Cache and is valid and has the same GroupIdToken as the IdToken that started the transaction.
C09.FR.08	If an IdToken is present in the Local Authorization List or Authorization Cache.	The Charging Station MAY send AuthorizeRequest to the CSMS.
C09.FR.09	If the CSMS accepts the IdToken.	AuthorizeResponse MAY include groupIdToken.
C09.FR.10		AuthorizeResponse SHALL include an authorization status value indicating acceptance or a reason for rejection.
C09.FR.11	C09.FR.03 AND A different IdToken is presented for stopping, which has the same GroupIdToken, but does not have *status* = `Accepted`	The Charging Station SHALL NOT stop the transaction and SHALL return an authorization status value indicating a reason for rejection.

2.4. Authorization Cache

C10 - Store Authorization Data in the Authorization Cache

Table 77. C10 - Store Authorization Data in Authorization Cache

No.	Type	Description
1	**Name**	Store Authorization Data in the Authorization Cache
2	**ID**	C10
	Functional block	C. Authorization
3	**Objective(s)**	To store all the latest received IdTokens in the Authorization Cache.
4	**Description**	This use case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station receives a AuthorizeResponse, ReserveNowRequest or TransactionEventResponse response message from the CSMS. **2.** The Cache is updated by the Charging Station using all received IdTokenInfo from the response message from the CSMS.
	Alternative scenario(s)	n/a
5	**Prerequisite(s)**	An Authorization Cache is implemented and and the value of the `AuthCacheEnabled` Configuration Variable is set to 'true'.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station stored the newly received IdTokenInfo data in the Authorization Cache. **Failure postcondition:** The Charging Station was *not* able to store the Authorization Cache.

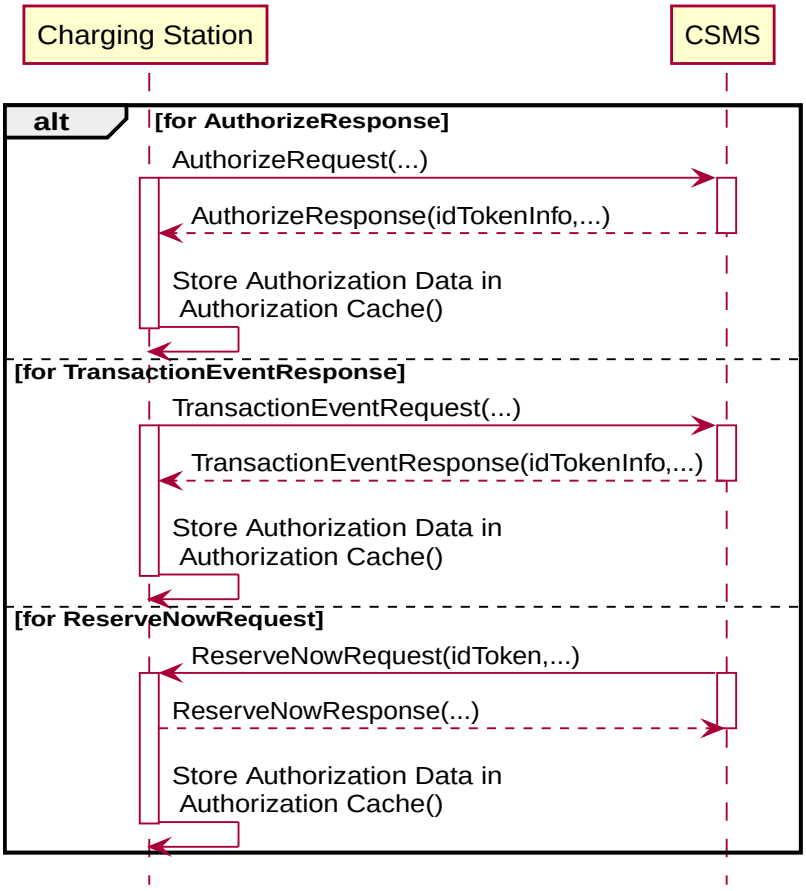

Figure 31. Sequence Diagram: Store Authorization Data in the Authorization Cache

7	**Error handling**	n/a

8	**Remark(s)**	n/a

C10 - Store Authorization Data in the Authorization Cache - Requirements

Table 78. C10 - Requirements

ID	Precondition	Requirement definition	Note
C10.FR.01		The Authorization Cache SHALL contain all the latest received identifiers (regardless of their status).	
C10.FR.02		Cache values SHOULD be persistent across reboots and power outages.	Hence cache values SHOULD be stored in non-volatile memory.
C10.FR.03	When an IdToken is presented that is stored in the Authorization Cache with status other than *Accepted*, and the Charging Station is online.	AuthorizeRequest SHALL be sent to the CSMS to check the current state of the IdToken.	To check the current state of the identifier.
C10.FR.04	Upon receipt of AuthorizeResponse.	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the response as described under Authorization Cache.
C10.FR.05	Upon receipt of TransactionEventResponse.	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the response as described under Authorization Cache.
C10.FR.06	Upon receipt of ReserveNowRequest.	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the request as described under Authorization Cache.
C10.FR.07		The Charging Station SHALL have a mechanism to accept new cache entries even when it is full, by deleting older entries.	It is suggested to remove any entries with status other than *Accepted* first, and then the oldest valid entries to make space for the new entry.
C10.FR.08		The time a token may live in the cache is determined by the Configuration Variable `AuthCacheLifeTime`. This variable indicates how long it takes until a token expires in the Authorization Cache since it is last used.	This expiry of the cache is **not** the same as the expiration date that is set for the IdToken (e.g. RFID card expiry date).
C10.FR.09	The Charging Station supports Tariff & Cost	The Charging Station SHALL NOT store the tariff information in the Cache.	
C10.FR.10	When the validity of an Authorization Cache entry expires.	The Authorization Cache entry SHALL be removed from the cache or changed to `Expired`.	
C10.FR.11		Whether the Authorization Cache is enabled or disabled SHALL be controlled by the `AuthCacheEnabled` Configuration Variable.	
C10.FR.12		It is RECOMMENDED to store personal information in the Authorization Cache securely	E.g. by only storing hashed idTokens in the cache.

C11 - Clear Authorization Data in Authorization Cache

Table 79. C11 - Clear Authorization Data in Authorization Cache

No.	Type	Description
1	**Name**	Clear Authorization Data in Authorization Cache
2	**ID**	C11
	Functional block	C. Authorization
3	**Objective(s)**	To clear all IdTokens in the Authorization Cache.
4	**Description**	This use case covers how the CSMS can request a Charging Station to clear its Authorization Cache.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS requests the Charging Station to clear its Authorization Cache by sending ClearCacheRequest. **2.** The Charging Station responds with the status *Accepted*.
5	**Prerequisite(s)**	Authorization Cache is supported and enabled by the `AuthCacheEnabled` Configuration Variable.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* cleared the Authorization Cache. **Failure postcondition:** The Charging Station was *not* able to clear the Authorization Cache.

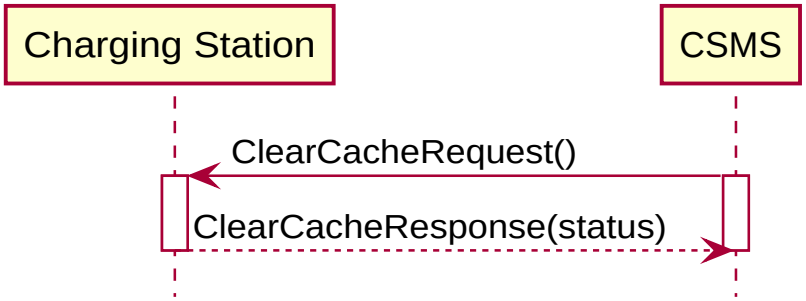

Figure 32. Sequence Diagram: Clear Authorization Data in Authorization Cache

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C11 - Clear Authorization Data in Authorization Cache - Requirements

Table 80. C11 - Requirements

ID	Precondition	Requirement definition
C11.FR.01	If the CSMS sends a ClearCacheRequest.	The Charging Station SHALL attempt to clear its Authorization Cache.
C11.FR.02	C11.FR.01	The Charging Station SHALL send ClearCacheResponse message indicating whether it was able to clear its Authorization Cache.
C11.FR.03	C11.FR.02 AND Charging Station successfully cleared its Authorization Cache.	The Charging Station SHALL send ClearCacheResponse message with the status *Accepted*.
C11.FR.04	C11.FR.02 AND Configuration variable `AuthCacheEnabled` is false	The Charging Station SHALL send ClearCacheResponse message with the status *Rejected*.
C11.FR.05	C11.FR.02 AND Charging Station failed to clear its Authorization Cache.	The Charging Station SHALL send ClearCacheResponse message with the status *Rejected*.

C12 - Start Transaction - Cached Id

Table 81. C12 - Start Transaction - Cached Id

No.	Type	Description
1	**Name**	Start Transaction - Cached Id
2	**ID**	C12
	Functional block	C. Authorization
3	**Objective(s)**	To enable the EV Driver to *Online* start a transaction by using the Authorization Cache. So the Charging Station can respond faster, as no AuthorizeRequest is being sent.
4	**Description**	This use case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver plugs in the cable. **2.** The Charging Station starts the transaction. **3.** The EV Driver presents an IdToken. **4.** The Charging Station verifies the IdToken with the Authorization Cache. **5.** The Charging Station updates the transaction. **6.** The Charging Station starts charging. **7.** E02 - Start Transaction - Cable Plugin First applies.
5	**Prerequisite(s)**	`AuthCacheEnabled` = true `LocalPreAuthorize` = true The Id of the EV Driver is Cached in the Authorization Cache Id is valid
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver is authorized to start a transaction by using the Authorization Cache. **Failure postcondition:** The UserId was not found in the Authorization Cache and: * Online Charging Station: the Charging Station issues an AuthorizeRequest and that fails too. * In an offline situation, behaviour of the Charging Station is defined by Configuration Variable OfflineTxForUnknownIdEnabled.

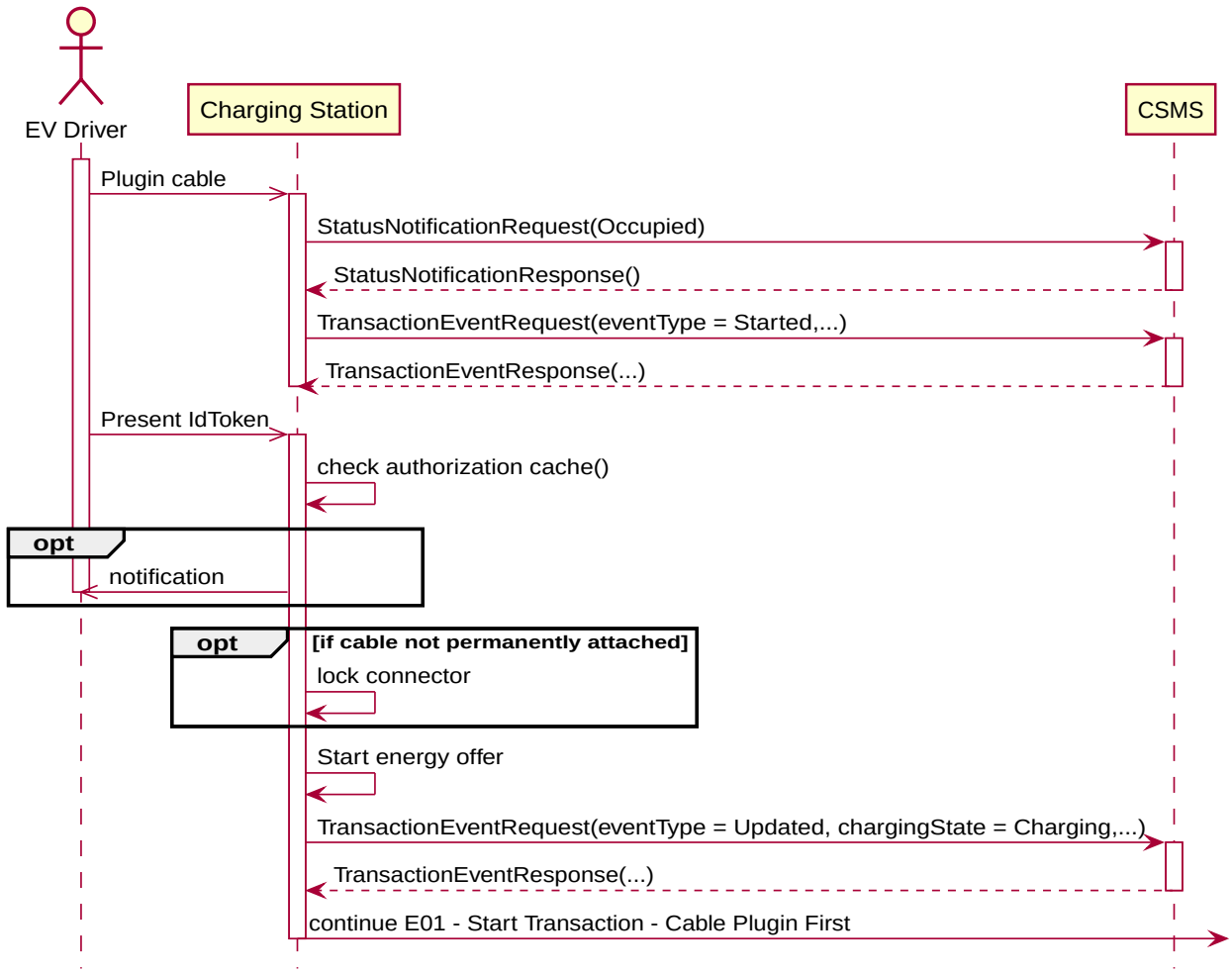

Figure 33. Sequence Diagram: Start Transaction - Cached Id

7	**Error handling**	When the Charging Station has an IdToken in the Authorization Cache, which is valid in the Authorization Cache, but is no longer valid in the CSMS: The Charging Station will receive the IdTokenInfo in the TransactionEventResponse which contains the newer invalid status. What happens in such a cases depends on the Configuration Variables: `MaxEnergyOnInvalidId` and `StopTxOnInvalidId`.
8	**Remark(s)**	If the Charging Station has implemented an Authorization Cache, then upon receipt of a AuthorizeResponse message the Charging Station updates the Cache entry. For a Cached valid IdToken it is not logical to send AuthorizeRequest. The TransactioneventResponse message also contains the IdToken information. If the IdToken has become no longer valid, the Charging Station will learn this from this TransactioneventResponse. So if the IdToken is no longer valid, the Charging Station might decide to stop the energy offering, and depending on the configuration even stop the transaction. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C12 - Start Transaction - Cached Id - Requirements

Table 82. C12 - Requirements

ID	Precondition	Requirement definition	Note
C12.FR.02	When an identifier is presented that is stored in the Authorization Cache as *Accepted*.	The Charging Station SHALL send a TransactionEventRequest with *idToken* to the CSMS.	

ID	Precondition	Requirement definition	Note
C12.FR.03	C12.FR.02	The CSMS SHALL check the authorization status of the IdToken when processing this TransactionEventRequest.	
C12.FR.04	C12.FR.02 AND The cable is plugged in.	The Charging Station SHALL start the energy offer.	
C12.FR.05	When an identifier is presented that is stored in the Authorization Cache with status other than *Accepted*, and the Charging Station is online.	The Charging Station SHALL send an AuthorizeRequest to the CSMS.	To check the current state of the identifier.
C12.FR.06	When IdTokenInfo is received for an identifier in the Cache.	The Authorization Cache SHALL be updated using the received IdTokenInfo.	
C12.FR.09	IdTokens that have a groupId equal to `MasterPassGroupId`	SHALL NOT be allowed to start a transaction.	

2.5. Local Authorization list

C13 - Offline Authorization through Local Authorization List

Table 83. C13 - Offline Authorization through Local Authorization List

No.	Type	Description
1	**Name**	Offline Authorization through Local Authorization List
2	**ID**	C13
	Functional block	C. Authorization
3	**Objective(s)**	To authorize an idToken by using the Local Authorization List while *Offline*.
4	**Description**	This use case describes how to authorize an IdToken, when communication with the CSMS is not possible. The Local Authorization List is a list of idTokens that can be synchronized with the CSMS. The list contains the authorization status of a selected set of idTokens as managed by the CSMS.
	Actors	EV Driver, Charging Station
	Scenario description	**1.** The Charging Station is *Offline* **2.** The EV Driver presents IdToken. **3.** The Charging Station checks if the IdToken is known and has status *Accepted* in the Local Authorization List. **4.** The Charging Station start charging.
5	**Prerequisite(s)**	*Local Authorization List* is available *Local Authorization List* is enabled via `LocalAuthListEnabled` Charging Station is *Offline* The Id of the EV Driver is in the *Local Authorization List* Id is valid
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station accepts tokens on the Local Authorization List when it is offline. **Failure postcondition:** The Charging Station does not accept tokens on the Local Authorization List when it is offline.

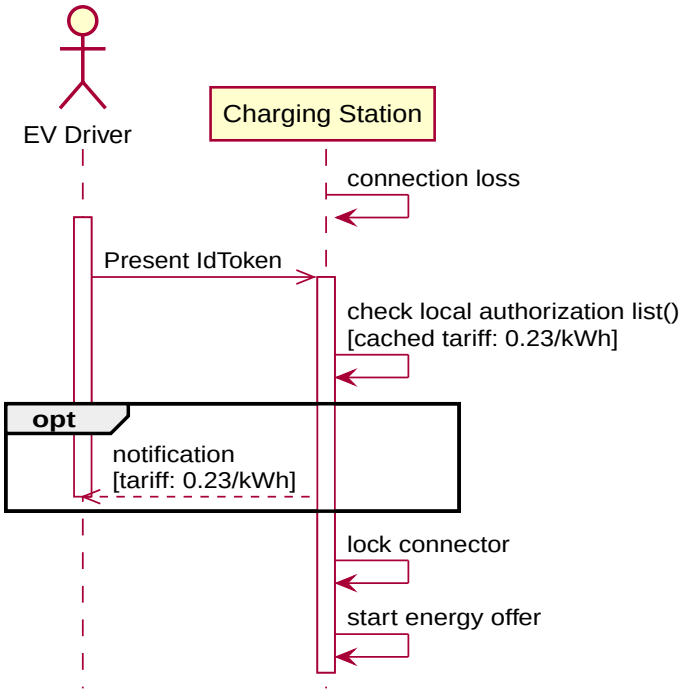

Figure 34. Sequence Diagram: Offline Authorization through Local Authorization List

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C13 - Offline Authorization through Local Authorization List - Requirements

Table 84. C13 - Requirements

ID	Precondition	Requirement definition	Note
C13.FR.01		Where both Authorization Cache and Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.	
C13.FR.02	If configuration variable `OfflineTxForUnknownIdEnabled` is false AND The Charging Station is offline.	Only identifiers that are present in a Local Authorization List that have a status *Accepted* SHALL be allowed to start a transaction.	
C13.FR.03		The Charging Station MAY authorize the IdToken locally without involving the CSMS.	As described in Local Authorization List.
C13.FR.04	If configuration variable `OfflineTxForUnknownIdEnabled` is true AND The Charging Station is offline.	Any identifier SHALL be allowed to start a transaction.	

C14 - Online Authorization through Local Authorization List

Table 85. C14 - Online Authorization through Local Authorization List

No.	Type	Description
1	**Name**	Online Authorization through Local Authorization List
2	**ID**	C14
	Functional block	C. Authorization
3	**Objective(s)**	To authorize an idToken by using the Local Authorization List while *Online*.
4	**Description**	This use case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.
	Actors	EV Driver, Charging Station

No.	Type	Description
	Scenario description	**1.** The EV Driver presents IdToken **2.** The Charging Station checks if the IdToken is known and has status *Accepted* in the Local Authorization List. **3.** If the IdToken is not known, or the IdToken is not *Accepted* the Charging Station sends an AuthorizeRequest **4.** The Charging Station starts charging.
5	**Prerequisite(s)**	*Local Authorization List* is available *Local Authorization List* is enabled via `LocalAuthListEnabled` The Id of the EV Driver is in the *Local Authorization List* Id is valid `LocalPreAuthorize` is set to *true*
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station accepts tokens on the Local Authorization List. **Failure postcondition:** The Charging Station does not accept tokens on the Local Authorization List.

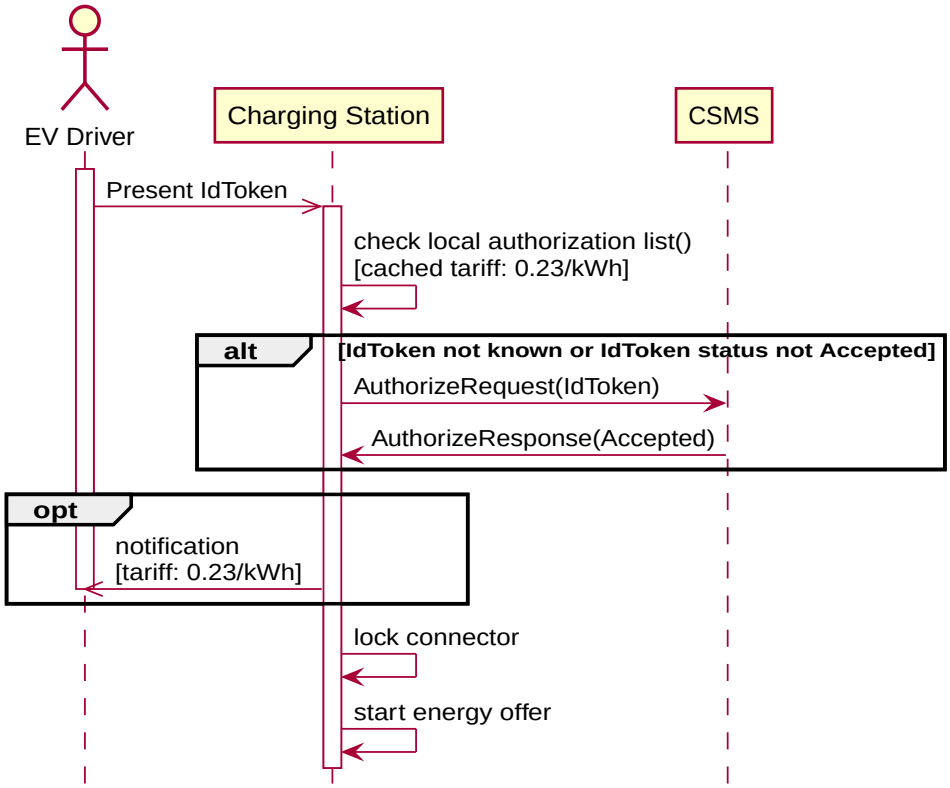

Figure 35. Sequence Diagram: Online Authorization through Local Authorization List

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C14 - Online Authorization through Local Authorization List - Requirements

Table 86. C14 - Requirements

ID	Precondition	Requirement definition
C14.FR.01		Where both Authorization Cache and Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.
C14.FR.02	Identifiers presented is in the Local Authorization List with a status *Accepted*	The Charging Station SHALL start charging without sending an AuthorizeRequest.
C14.FR.03	Identifiers presented is in the Local Authorization List with a status OTHER than *Accepted*	The Charging Station SHALL send and AuthorizeRequest to try to authorize this IdToken.

2.6. Offline Authorization

C15 - Offline Authorization of unknown Id

Table 87. C15 - Offline Authorization of unknown Id

No.	Type	Description
1	**Name**	Offline Authorization of unknown Id
2	**ID**	C15
	Functional block	C. Authorization
	Parent use case	C12 - Start Transaction - Cached Id
3	**Objective(s)**	To allow automatic authorization of any "unknown" identifiers that cannot be explicitly authorized by Authorization Cache entries.
4	**Description**	This use case describes the scenario of presented "unknown" identifiers, other than are present in an Authorization Cache or Local Cache entry using `OfflineTxForUnknownIdEnabled`.
	Actors	Charging Station, EV Driver
	Scenario description	**1.** The EV Driver wants to start charging the EV and presents the IdToken. **2.** The Charging Station checks the Authorization Cache, the IdToken is not present in the Authorization Cache. **3.** The Charging Station checks the Local Authorization List, the IdToken is not present in the Local Authorization List. **4.** The Charging Station accepts the unknown IdToken if `OfflineTxForUnknownIdEnabled` is set *True* **5.** The Charging Station rejects the unknown IdToken if `OfflineTxForUnknownIdEnabled` is set *False*
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)
5	**Prerequisite(s)**	The Charging Station is *Offline*. Unknown IdToken presented (Not in the Authorization Cache and/or Local Authorization List).
6	**Postcondition(s)**	**Successful postcondition:** The authorization status in TransactionEventResponse is *Accepted*. **Failure postcondition:** The authorization status in TransactionEventResponse is *not Accepted* when `OfflineTxForUnknownIdEnabled` is *True*.

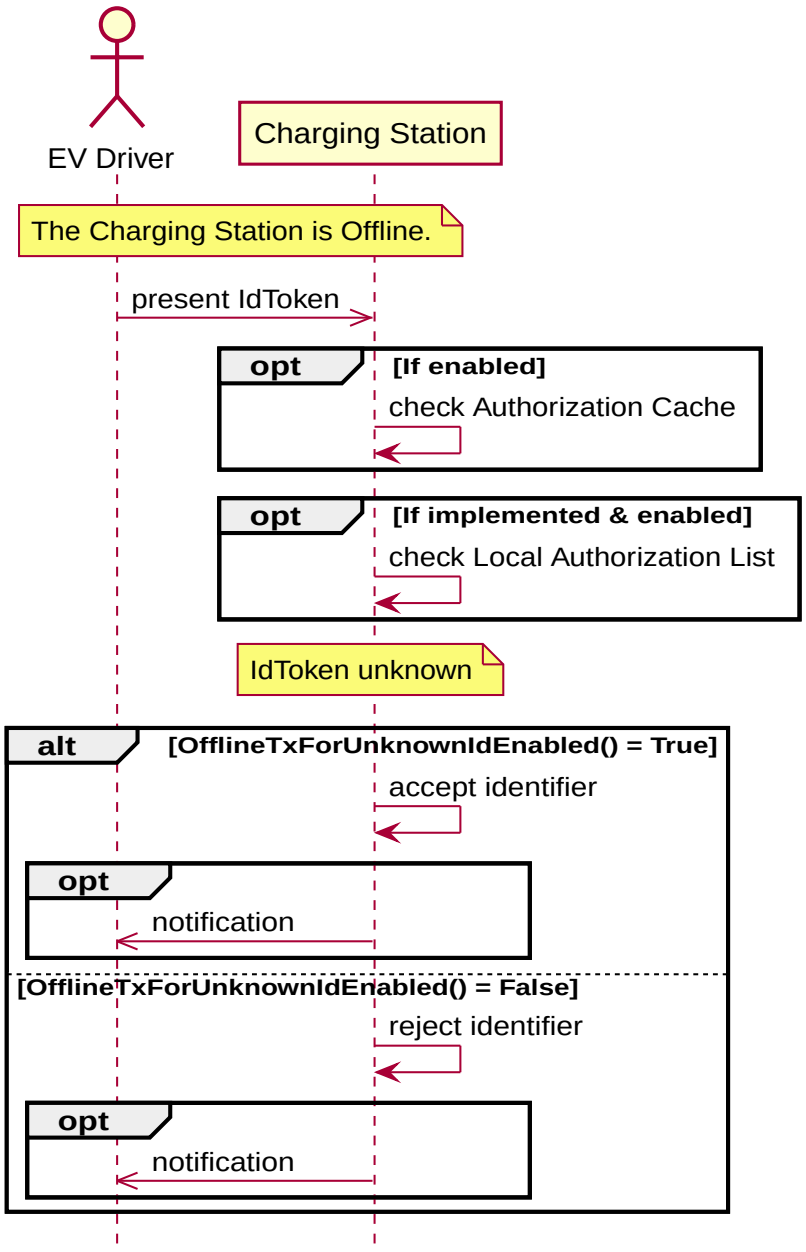

Figure 36. Sequence Diagram: Start Transaction - Unknown Offline Authorization

7	**Error handling**	n/a
8	**Remark(s)**	This applies to all types of identifiers, including an eMAID that is presented as part of an ISO 15118 contract certificate.

C15 - Offline Authorization of unknown Id - Requirements

Table 88. C15 - Requirements

ID	Precondition	Requirement definition	Note
C15.FR.01	If the identifier is authorized via `OfflineTxForUnknownIdEnabled`	The Charging Station SHALL NOT add the token to Authorization Cache	
C15.FR.02	When connection to the CSMS is restored	The Charging Station SHALL send a TransactionEventRequest for any transaction that was authorized *offline*.	As explained in transaction-related message handling

ID	Precondition	Requirement definition	Note
C15.FR.03	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is *true* AND `TxStopPoint` does NOT contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the energy transfer and send TransactionEventRequest (eventType = Updated) with *triggerReason* set to *Deauthorized* and chargingState set to *SuspendedEVSE*.	
C15.FR.04	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is *true* AND `TxStopPoint` does contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the transaction and send TransactionEventRequest (eventType = Ended) with *triggerReason* set to *Deauthorized* and stoppedReason set to *DeAuthorized*.	
C15.FR.05	C15.FR.04 AND If the Charging Station has the possibility to lock the Charging Cable	The Charging Station SHOULD keep the Charging Cable locked until the owner presents his identifier.	
C15.FR.06	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is set to *false* AND `MaxEnergyOnInvalidId` is not implemented or has been exceeded. `TxStopPoint` does NOT contain: (PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the energy delivery to the EV immediately and send TransactionEventRequest (eventType = Updated) with *triggerReason* set to *ChargingStateChanged* and chargingState set to *SuspendedEVSE*	
C15.FR.07	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is set to *false* AND `MaxEnergyOnInvalidId` is set and has NOT been exceeded.	Energy delivery to the EV SHALL be allowed until the amount of energy specified in `MaxEnergyOnInvalidId` has been reached.	
C15.FR.08	When an unknown identifier is presented AND OfflineTxForUnknownIdEnabled is set to *true*	The Charging Station SHALL accept the presented IdToken.	

2.7. Master Pass

C16 - Stop Transaction with a Master Pass

Table 89. C16 - Stop Transaction with a Master Pass

No.	Type	Description
1	**Name**	Stop Transaction with a Master Pass
2	**ID**	C16
	Functional block	C. Authorization

No.	Type	Description
3	**Objectives**	Enable stopping of transactions by use of a Master Pass (for example for: Law Enforcement officials).
4	**Description**	This use case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: `MasterPassGroupId`.
	Actors	Charging Station, CSMS, User
	Scenario description	**1.** The User (Law Enforcement official) presents his IdToken at the Charging Station. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message contains a GroupId that equals the value of the Configuration Variable `MasterPassGroupId` and the idToken is valid. **4a.** If the Charging Station has a UI, then the Charging Station "Shows" the Master Pass UI. **5a.** The user selects which transactions to stop. **6a.** The Charging Station stops the selected transaction(s) AND sends a TransactionEventRequest (eventType = Ended, stopReason = MasterPass) to the CSMS for every stopped transaction. **7a.** Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse. **4b.** If the Charging Station does NOT have a UI, then the Charging Station stops all transactions AND sends a TransactionEventRequest (eventType = Ended, stopReason = MasterPass) to the CSMS for every stopped transaction. **5b.** Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse.
	Alternative scenario(s)	C01 - EV Driver Authorization
5	**Prerequisites**	Ongoing Transaction(s) Configuration Variable: `MasterPassGroupId` set. Users IdToken has groupId equal to the configured `MasterPassGroupId`.
6	**Postcondition(s)**	(Selected) transaction(s) stopped.

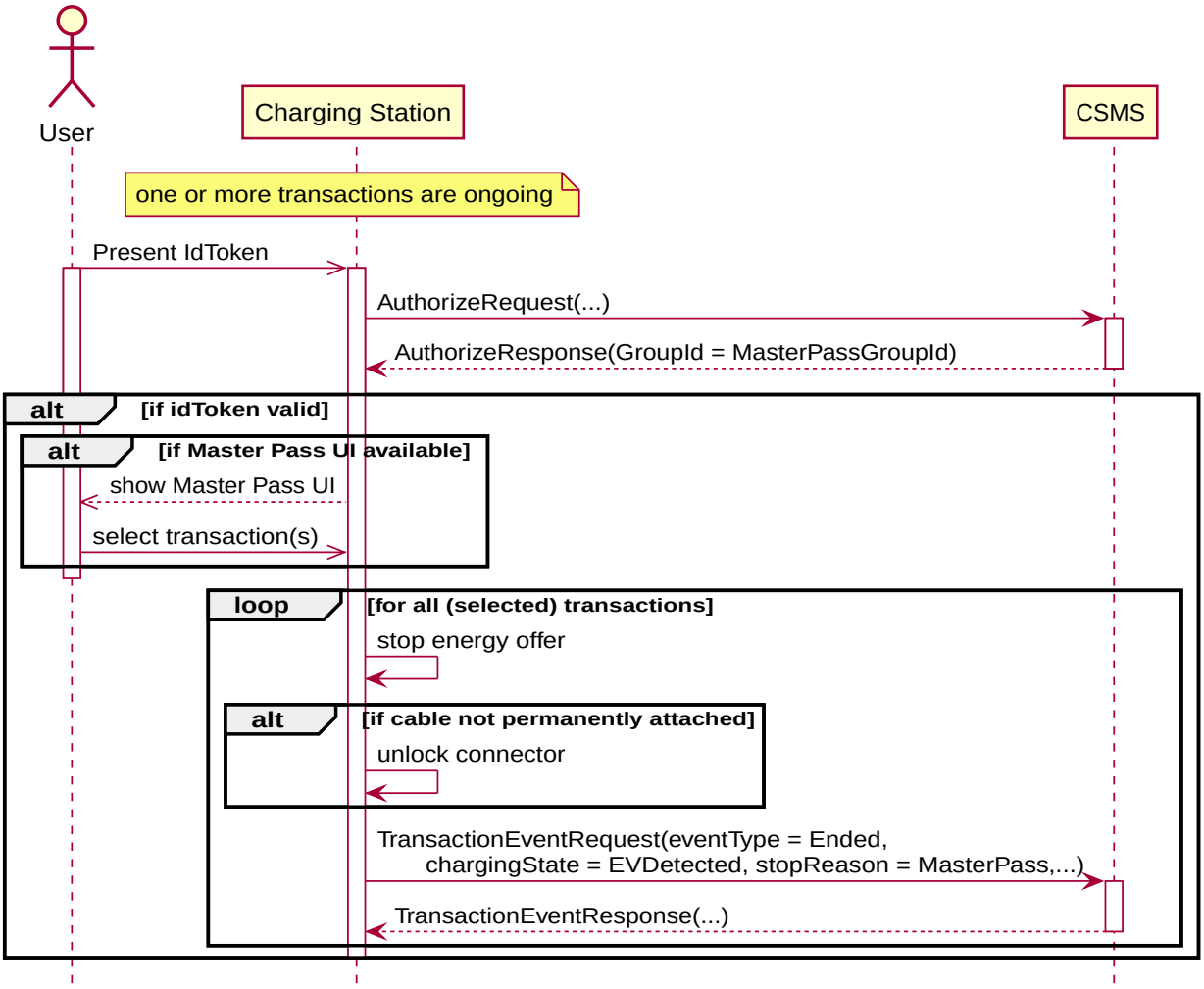

Figure 37. Sequence Diagram: Stop Transaction with a Master Pass

7	**Error Handling**	When the user does not make a selection before an acceptable timeout, the Charging Station SHALL go back to normal operation.
8	**Remarks**	The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are send. For more details see the use case: E06 - Stop Transaction options

C16 - Stop Transaction with a Master Pass - Requirements

Table 90. C16 - Stop Transaction with a Master Pass - Requirements

ID	Precondition	Requirement definition
C16.FR.01	User presents an IdToken that has a groupId equal to `MasterPassGroupId` AND The Charging Station has a UI.	The Charging Station SHALL "show" the Master Pass UI.
C16.FR.02	User presents an IdToken that has a groupId equal to `MasterPassGroupId` AND the Charging Station does NOT have a UI.	The Charging Station SHALL stop all ongoing transactions.
C16.FR.03	IdTokens that have a groupId equal to `MasterPassGroupId`	SHALL NOT be allowed to start a transaction.
C16.FR.04	IdTokens that have a groupId equal to `MasterPassGroupId` present in the Authorization Cache.	The Charging Station MAY also allow authorization of "Master Pass" tokens based on information in the Authorization Cache.
C16.FR.05	IdTokens that have a groupId equal to `MasterPassGroupId` present in the Local Authorization List.	The Charging Station MAY also allow authorization of "Master Pass" tokens based on information in the Local Authorization List.