Annex F (Interface Specifications)

Annex to the EETS Domain Statement concerning the Danish Kilometer Tolling Scheme

Version: 1.2

Date: 13 December 2024

TABLE OF CONTENTS

1 Document history

Date of first appearance of this entry into the register	1 February 2024
Last update	13 December 2024
Next review	First quarter 2025

2 Definitions and abbreviations

All definitions in the EETS Domain Statement shall have the same meaning in this Annex.

In addition to the definitions in the EETS Domain Statement the following definitions shall apply for this Annex:

ADU	Application Data Unit
APDU	Application Protocol Data Unit
API	Application Programming Interface
APIM	API Management
ASN	Abstract Syntax Notation
CCC	Compliance Check Communication
CRL	Certificate Revocation List
DSRC	Dedicated Short-Range Communications (DSRC in this context refers to CEN DSRC)
EDU	Education environment, a training sandbox environment for testing APIs
EETS	European Electronic Tolling Service
EFC	Electronic Fee Collection
FQDN	Fully Qualified Domain Name
HTTPS	Hypertext Transfer Protocol Secure
ISO	International Organisation for Standardisation
JSON	JavaScript Object Notation
KmToll	Kilometer Tolling Scheme in Denmark
OBE	On Board Equipment
REST	Representational State Transfer
TC	Toll Charger
TSP	Toll Service Provider. Note this is also referred to as an EETS Provider
UAT	User Acceptance Test, this is either referred to the environment where the UAT is conducted or the actual test
UVI	Unique Vehicle Identifier. A combination of OBE ID, number plate and other fields which uniquely identifies the vehicle. Formally defined in Annex E

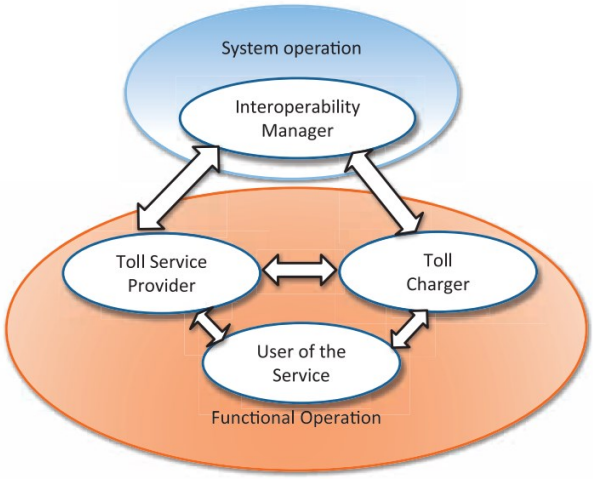
3 Introduction

This Annex describes the interfaces between the EETS Provider[1] (TSP) and Toll Charger (TC) in the Danish Kilometer Tolling (KmToll) Scheme. The purpose for this Annex is to get familiar with the interface design and understand the overall requirements and architecture to get started. This Annex is intended for technical personnel that understand APIs and know how to integrate with APIs.

It is not the intention that this specification describes in detail each field that is exposed in the API endpoint, as this will be documented using the OpenAPI v. 3.0 (OpenAPI Specification v3.0.0) using REST API. This will help reducing the maintenance of this documentation when APIs are going through updates (versioning).

The architecture of the KmToll Scheme is compliant with the ISO 17573 Electronic Fee Collection (EFC) set of standards. DS ISO 17573-1:2019 defines the roles in an EFC system as shown below:

Figure 1. Roles in an EFC scheme as defined in DS ISO 17573-1



Sund & Bælt will be the Toll Charger (TC). Regular users will buy EETS from a TSP accredited onto the KmToll Scheme. Occasional users (or users who are unable or unwilling to enter into a service contract with a TSP) will be able to buy a toll ticket from a website operated by the Toll Charger. Occasional users fall outside the scope of this document. The Interoperability Manager is a divided role with responsibilities split between the TC, The Danish Road Directorate ("Vejdirektoratet") and the Danish Ministry of Taxation ("Skatteministeriet").

This architecture is realised in the diagram shown below (Figure 2), which shows the interfaces between the various system actors. This is adapted from the system architecture diagram in ISO 12855.

Data exchange will as far as is practical follow the requirements of standard EN 16986, which in turn is based on the underlying ISO 12855:2022 toolbox standard. These standards define the data formats for exchange between the various actors in a tolling system. It is essential that a TSP wishing to connect to the Danish KmToll Scheme is familiar with these standards as they provide the underlying data and messaging formats to be used. All references in this document to ISO 12855 refer specifically to the 2022 version (ISO 12855:2022).

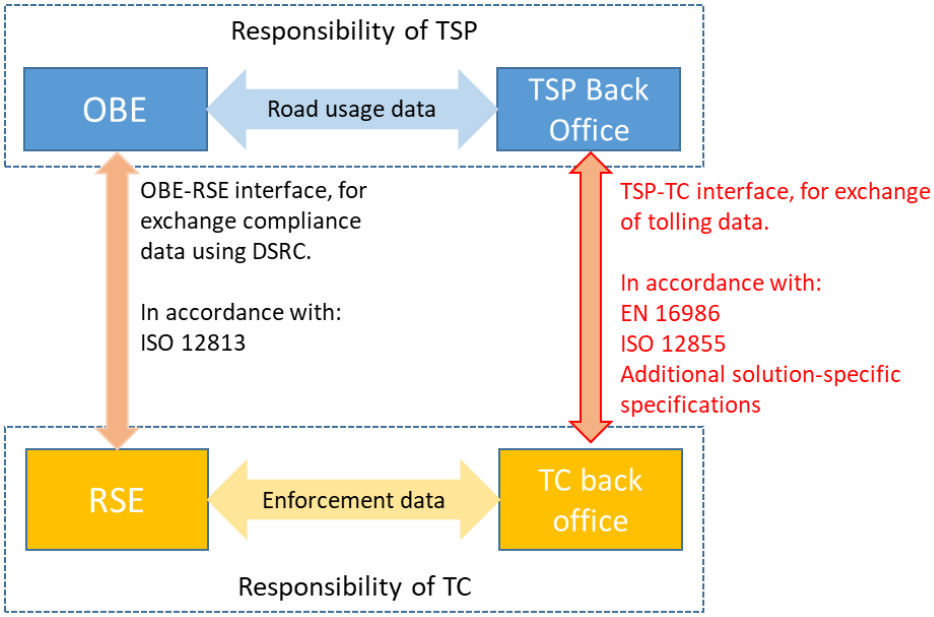
At the time of writing, the current ratified version of EN 16986 is EN 16986:2016. However, this version is outdated and does not adequately address the requirements of a Toll Charger dominant GNSS-based EETS compliant system, hence a new version of EN 16986, currently identified as

[1] The terms EETS Provider (EP) and Toll Service Provider (TSP) are used interchangeably. In this document we use the term TSP.

preliminary version prEN 16986:2023 is in the process of ratification. It is expected that this ratification will be completed within the timescales of this project. However, it is noted that the updates included in prEN 16986:2023 do not adequately meet the requirements of the KmToll Scheme, and the defined interfaces in this document introduce further changes to the standard. As this is no longer a fully compliant implementation of EN 16986, the *aidIdentifier* attribute on all TSP to TC the specifications will be changed in the future contain the value of 32, indicating a private version of the communications protocol, rather than the value of 17 which has been reserved for EN 16986:2023. At the present time, TSPs should continue to use the value of 17 for *aidIdentifier* until informed otherwise.

For the purposes of this Annex, all references to EN 16986 refer to prEN 16986:2023, unless specifically identified differently.

Figure 2. Interfaces in the Danish KmToll Scheme



The On-Board Equipment (OBE) and TSP Back Office are the responsibility of the TSP, so the interface which carries the road usage data between them, is outside the scope of this document[2]. The interface between the TC Back Office and the Roadside Equipment (RSE, used for enforcement) is the responsibility of the TC, so is outside the scope of this document.

The DSRC interface between the RSE and the EETS User's OBE must comply with standard ISO 12813:2019. As this constitutes an interface between the TSP and the TC, this interface is elaborated in this document in section 6.14.

The interface between the TC and TSP Back Office (marked in red) is in scope. These interfaces must be compliant with the Profile standard EN 16986, which in turn uses the toolbox interface standard ISO 12855 as its base standard. While EN 16986 defines the data elements to be used, the standard is insufficiently detailed, so this specification will further elaborate on the data elements to be used, as well as any restrictions to these data elements.

While the TSP to TC interface will follow EN 16986 as closely as possible, a different transfer mechanism will be used from those described in section 6.6.2 of the standard. The options described (web services using SOAP, or file transfer using FTP/FTPS) are no longer appropriate. Instead, a REST API with a JSON payload supported by OpenAPI v3.0 documentation will be used

[2] Note that ISO 12855 requires that this interface must be compliant with ISO 17575. However, compliance with this is the responsibility of the TSP.

for communication between TC and the TSP. The data exchange process is described in more detail in the following section.

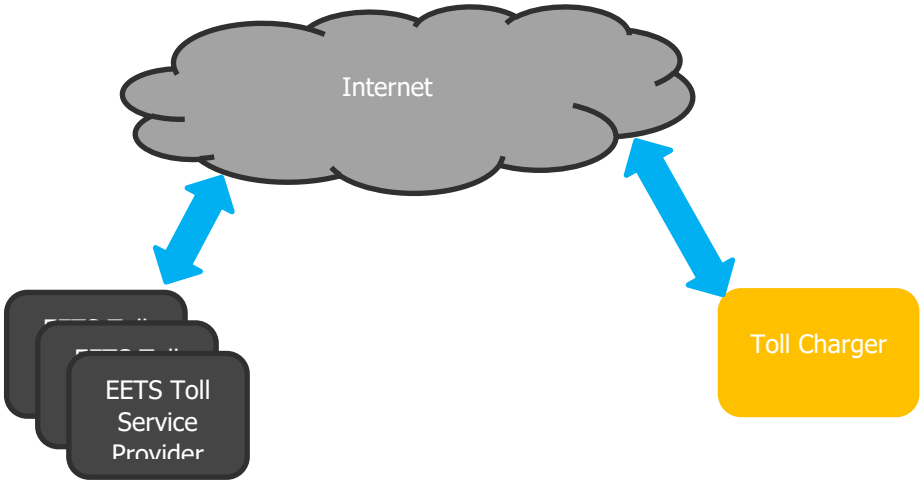
In addition to tolling services using dedicated On-Board Equipment (OBE) as envisaged by EN 16986, the TC intends to allow the TSP to offer a tolling service using non-dedicated nomadic devices as OBE, for example smartphones. For simplicity, we will refer to dedicated OBE as OBE Type 1, and nomadic devices as OBE Type 2.

As OBE Type 2 cannot be assumed to include DSRC capability, additional interfaces between the TSP and TC back offices are specified to facilitate the Compliance Check Communication (CCC) functionality normally implemented on the DSRC interface. These CCC interfaces are not defined in EN 16986, therefore additional solution-specific specifications are defined in this specification.

4 Data exchange process

The transfer of operational data between TSPs and the TC will use Web Services as shown below.

Figure 3. Connections between TSP and TC



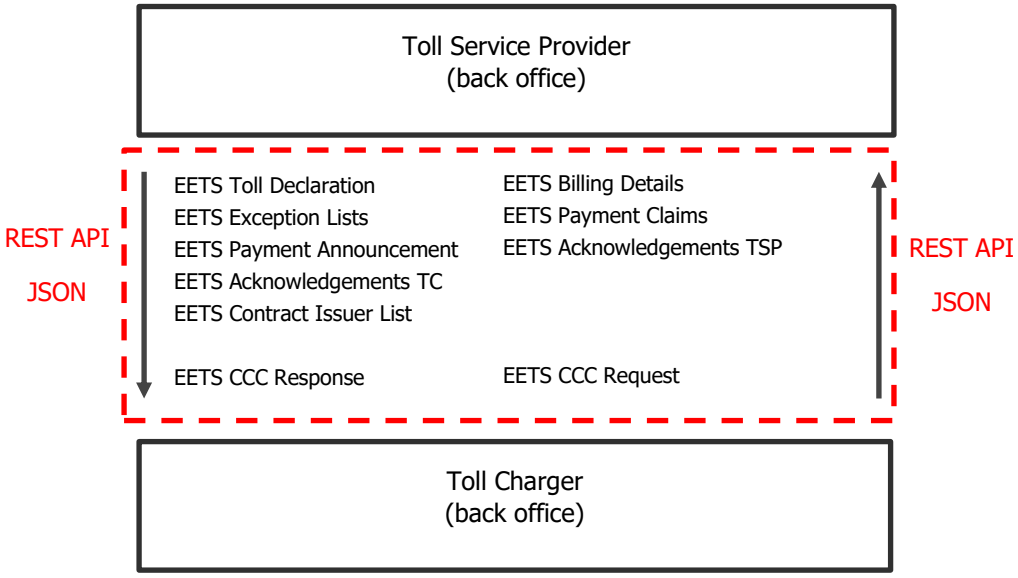
As previously mentioned, data exchange will follow the requirements of standard EN 16986, which in turn is based on the underlying ISO 12855:2022 toolbox standard. These standards define the data formats for exchange between the various actors in a tolling system.

All data exchanges other than Trust Object and Actor Table exchanges will make use of RESTful web services. In a deviation from EN 16986, the exchange of data between the TSP and TC will use REST APIs and JSON payload as shown below.

Note that this deviation from the standard is limited to the Transfer Mechanism described in section 6.6 of the standard. The contents of the messages transferred over the API will be coded in JSON[3] using the ADU structures per the standard, with additional data restrictions where required.

[3] It is recognised that previous versions of EN 16986 have required that data be coded in XML and defined in an XSD schema definition. For this project it has been decided that JSON is a more suitable coding for a RESTful Web interface, hence JSON coding is defined.

Figure 4. TSP to TC interfaces



The picture above illustrates the HTTPS REST APIs towards the TC, using endpoints for exchanging data to and from the TC. As noted before, CCC Data Request and CCC Data Response are new message types which are not defined in EN 16986.

Data exchange will follow the requirements of standard EN 16986, which in turn is based on the underlying ISO 12855:2022 base standard. All data exchanges other than Trust Object and Actor Table exchange are translated into REST APIs described in the OpenAPI 3.0 for sending data, and likewise for receiving data. Both are described in the JSON format.

To help app developers to integrate towards the TC's APIs, a Developer Portal will be deployed shown below (

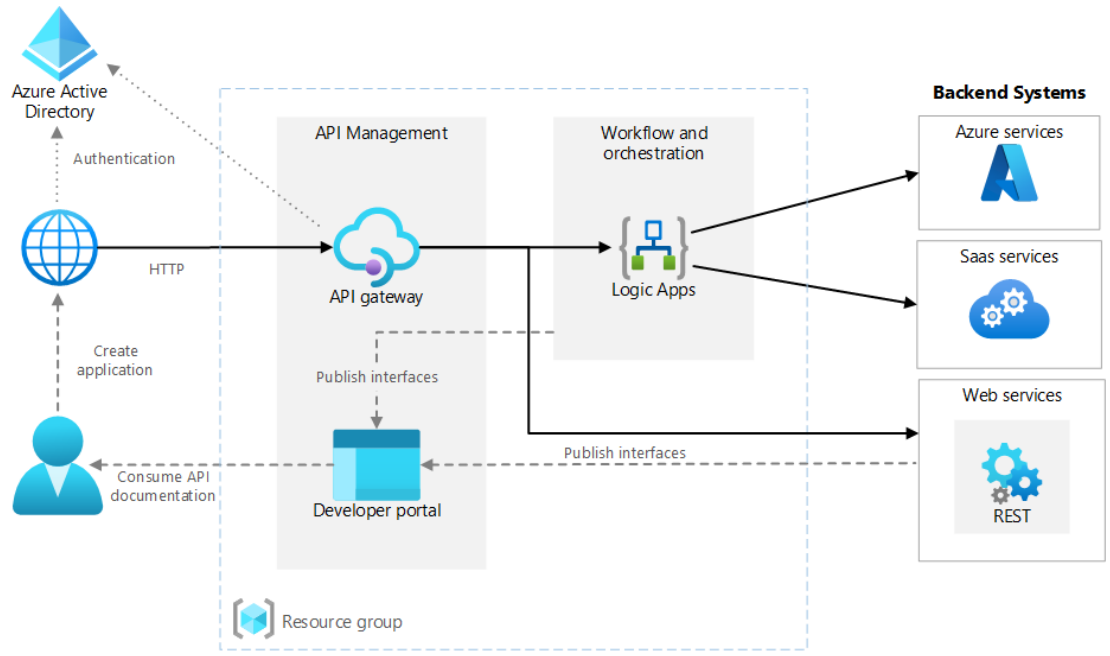
Figure 5). The managed Developer Portal is a feature in Azure API Management that allows internal or external developers to see specifications and try APIs that are published though a product (group of APIs) on the Developer Portal. The TC will use the API Management to also expose any of the TSP's internal APIs as backend APIs, this ensures that all communication and authentication are handled in one place. Therefore, the TSP will need to develop their own APIs on the OpenAPI 3.0 specification and JSON payload defined by the TC. This document only describes the initial view on the APIs, and future updates and change handling of the APIs are to be found in the Developer Portal.

Developers need to get authenticated and authorised by the TC to see the exposed API products available to the user. The Developer Portal will handle the full life cycle of the TC's APIs, being updates, new APIs or deprecating an API. Likewise, any updates to the TSP's APIs will need to be updated and re-configured within the TC API Management, as this will act as a client when calling the TSP's own APIs.

A document will be distributed to the TSP as part of the Accreditation Procedure to disclose the full URL for the Developer Portal as well the processes for exposing the required APIs from the TSP described in point 3 from the

Figure 6.

Figure 5. API Management Developer Portal



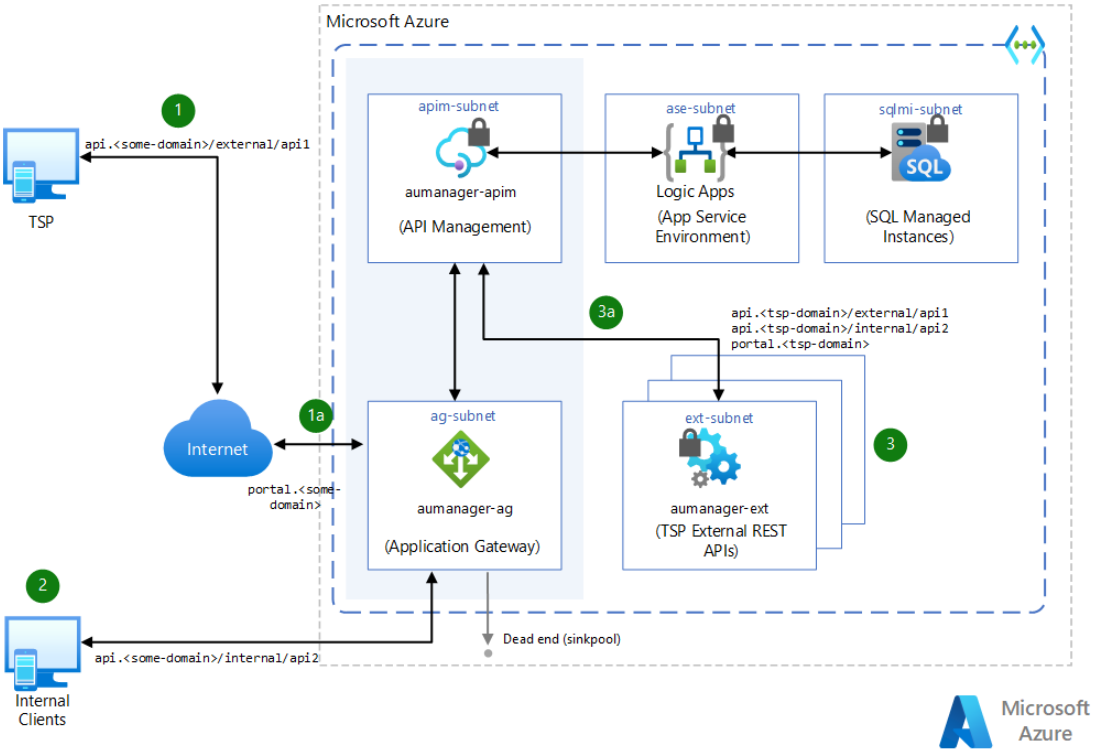
Microsoft Azure

Workflow:

To describe the workflow between the TC and TSP, please see

Figure 6. The TC will have a sandbox for reading OpenAPI documentation for each API in the Developer Portal, which only will be available in the two environments (education environment and UAT). Please see section 0 for further information about the environments. The flow below only describes the actual API call and how it is routed and handled with the TC in the production environment.

Figure 6. External APIs from TSP in APIM



1. Calls from the TSP to TC happens through the Application Gateway
2. Internal private calls within TC workloads in Azure, these are not public available
3. Example of the TSP APIs, which are exposed as "backend APIs", this is used for calling back to the TSP. This ensures scalability for internal TC components, and a single secure integration point between the TSP and TC systems. API Management would play a role as client when calling the TSP APIs.

The API Management is protected with application gateway in front, which setup the URL redirection mechanism that sends the request to the proper backend pool, depending on the URL format of the API call:

URLs formatted like `api.<some-domain>/external/*` can reach the back end to interact with the requested APIs. These are calls meant to be from TSP towards TC.

The APIs that are hosted at the TSP are only exposed in the API Management as pointers with authentication configured (no. 3 +3a in (

Figure 6)). These are showed in the figure as `api.<tsp-domain>/external/api`, but will be replaced with the actual endpoints given by TSPs for a given environment.

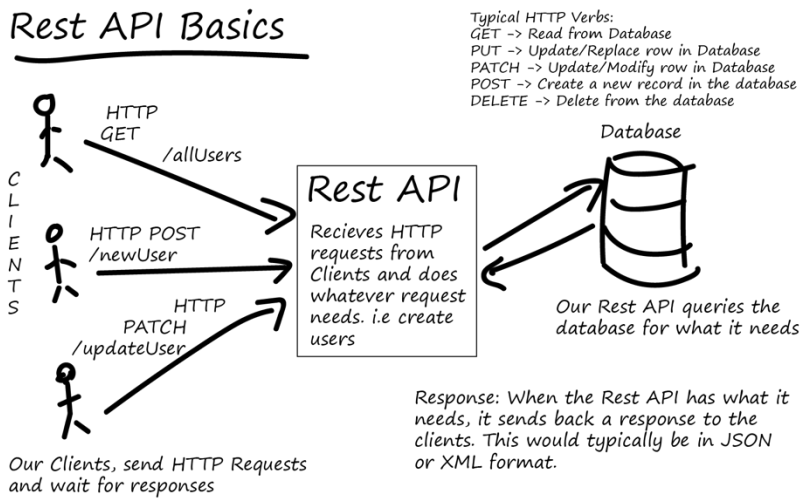
The TC endpoints for each environment are defined in section 0.

4.1 Message Level Protocol: REST

RESTful APIs have become the main standard for enabling communication between the server part of a product and its client, and it will be used for all interfaces when communicating data towards TC and TSP.

Example:

Figure 7. REST API basics



The key principle of REST is to divide the API into logical resources. A Uniform Resource Identified, or URI, is a sequence of symbols that identifies a resource and often allows developers to access representations of that resource. The current and syntax of URIs is defined by the RFC 3986 standard.

Example of the HTTPS methods and URI:

Figure 8. Components of a URI

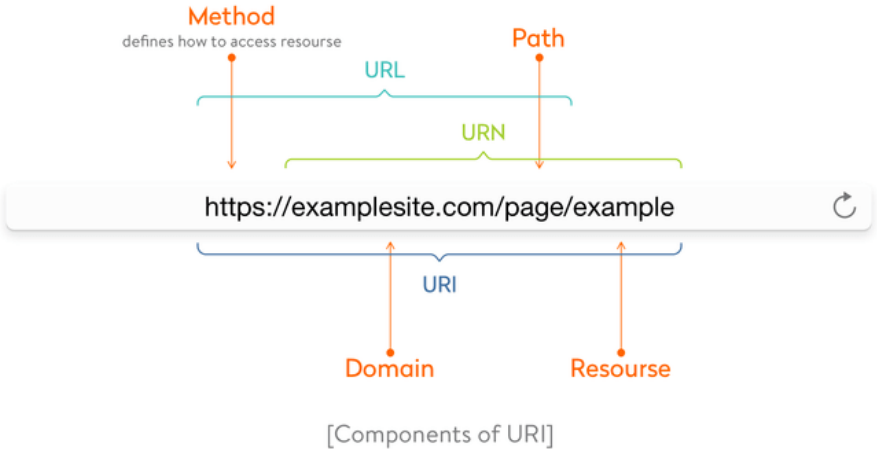


Figure 9. HTTPS Methods

HTTP Method	Action	Examples
GET	Obtain information about a resource	http://example.com/api/orders (retrieve order list)
GET	Obtain information about a resource	http://example.com/api/orders/123 (retrieve order #123)
POST	Create a new resource	http://example.com/api/orders (create a new order, from data provided with the request)
PUT	Update a resource	http://example.com/api/orders/123 (update order #123, from data provided with the request)
DELETE	Delete a resource	http://example.com/api/orders/123 (delete order #123)

Example of response:

The response is a JSON file with an embedded list.

Figure 10. HTTPS Response – 200 OK

HTTP response

```
HTTP/1.1 200 OK

content-length: 0
date: Tue, 19 Sep 2023 11:21:58 GMT
request-context: appId=cid-v1:27ec7f53-2c25-41b7-af6a-8
ff2c674c4d6
requestid: 9e3d71bb-85fd-41bb-bdd0-6acdedfcd242
```

Example on error on a specific parameter validation:

Figure 11. HTTPS Error response

HTTP response

```
HTTP/1.1 400 Bad Request

content-length: 342
content-type: application/json
date: Tue, 19 Sep 2023 11:22:57 GMT
request-context: appId=cid-v1:27ec7f53-2c25-41b7-af6a-8
ff2c674c4d6
requestid: 2802e93a-db4f-44e0-b642-c64d9c0915f9

{
    "Status": "Error",
    "Message": "Format Validation Error",
    "ResponseTime": "2023-09-19T11:22:57.4015517Z",
    "CorrelationId": "2802e93a-db4f-44e0-b642-c64d9c091
5f9",
    "Errors": [{
        "Type": "enum",
        "Field": "/InfoExchange/InfoExchangeContent/apc
i/informationrecipientID/countryCode",
        "Message": "Expected value to match one of the
values specified by the enum"
    }]
}
```

Sample 1


```
{
    "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
    "title": "One or more validation errors occurred.",
    "status": 400,
    "traceId": "|61c569d1-431ac3d5e5d6f164.",
    "errors": {
        "since": [
            "The value '2022-04-26T00:00:00 01:00' is not valid."
        ]
    }
}
```

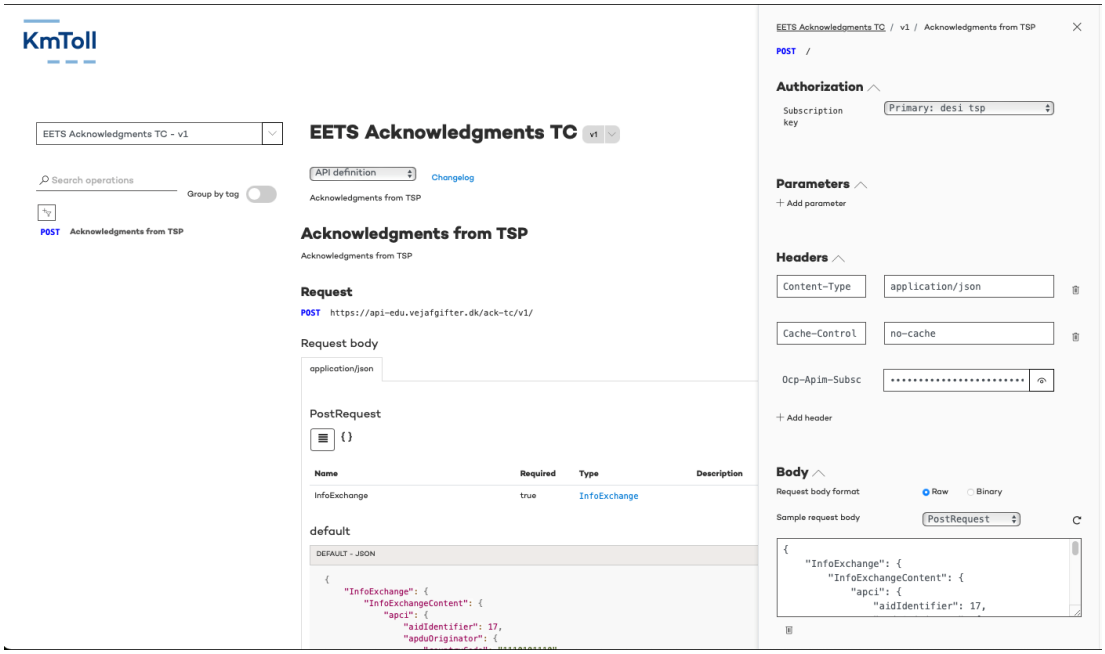

OpenAPI Specification:

Each APIs in API Management will have an OpenAPI documentation as well as a swagger UI in the Developer Portal, so the developers are able to see and try the APIs directly in the portal. Please refer to the following URL for more information:

https://swagger.io/specification/

Swagger UI example in APIM:

Figure 12. Swagger UI example in APIM



4.2 Message types and interfaces

Data to be exchanged will be contained in Application Protocol Data Units (APDUs) which will be defined by the ASN.1 type InfoExchange, as defined in Section 6.2 of ISO 12855:2022.

Data exchanges between the TSP and the TC are called Transactions in EN 16986. The following EN 16986 transactions must be supported:

Table 1. EN16986 Transactions

Transaction	Description	Typical frequency
EXCEPTIONLISTS	Allows a TSP to send exception lists to the TC. White- and blacklists are supported, which must be incremental	Ad-hoc (incremental lists)
TOLLDECLARATIONS_SECT	Allows a TSP to transfer toll declarations to TC in a section-based, autonomous toll scheme	Every five (5) minutes for each active vehicle
BILLINGDETAILS_TC	Allows a TC to transfer billing details to the TSP in a TC-Dominant system	Daily
PAYMENTCLAIM	Allows a TC to invoice a TSP	Contractually agreed
PAYMENTANNOUNCEMENT	Allows a TSP to inform the TC that a payment has been made	Contractually agreed
CONTRACTISSUERLIST	Allows a TSP to send contractual information regarding OBE to the TC	Ad-hoc
TRUSTOBJECTS*	Allows the exchange of trust objects between the TSP and the TC	Ad-hoc

* Note that the TRUSTOBJECTS transaction is expected to be implemented manually, and hence will not require a web interface.

In a previous version of this document, the REPORTABNORMALOBE transaction was specified, along with an associated Web API. The implementation of this transaction and its associated interface has been suspended until further notice. TSPs will not be required to implement this interface during the initial roll-out of the KmToll system. Sund & Bælt reserve the right to require that TSPs support this interface at some future date.

In addition to the transactions defined in EN 16986, a transaction (CCCDATAEXCHANGE) has been defined which allows the TC to request compliance check data (CCC) from a TSP. This is to be used in the case where compliance data cannot be directly obtained from the roadside over the DSRC compliance check interface. This has been created to support OBE which does not have a DSRC capability, for example where an OBE is implemented as an app running on a smartphone platform.

A transaction (ACTORTABLEEXCHANGE) to allow TSPs and TCs to exchange company-specific information (for example addresses, contact information, bank details etc) is also required. This will not be an API-based interface, but will be based on email exchanges.

The transactions are described below:

Table 2. Additional transactions not defined in EN 16986

Transaction	Description	Typical frequency
CCCDATAEXCHANGE	Allows a TC to request CCC information for a specific user from a TSP. The TSP must respond with a package of CCC data which is similar in content to that supplied over the DSRC CCC interface for a DSRC-capable OBE. Only required if the TSP supports OBE Type 2.	Ad-hoc
ACTORTABLEEXCHANGE	Allows the TC or TSP to send updated information regarding addresses, bank details etc securely. This transaction is only expected to be required infrequently, so an API will not be defined for it. Instead, actor table information will be exchanged using a manual exchange mechanism e.g. secure email, secure file share etc. defined by the Toll Charger.	Ad-hoc

A Transaction usually consists of either one, two or three messages, typically a message containing data, and a message acknowledging the data. Each message consists of an InfoExchange Application Protocol Data Unit (APDU). An APDU is a self-contained message transferred between TSP and TC. An APDU in turn consists of Application Protocol Control Information (ACPI), one or more Application Data Units (ADUs) which contains the operational data to be transferred, and optionally authentication data. See ISO 12855:2022 for more details. The KmToll Scheme will not use authentication as defined in the InfoExchange definition as authentication will be provided by the underlying communications services.

Messages between the TSP and TC will make use of a number of predefined interfaces. These interfaces, which are defined in this document, are:

Table 3. Interfaces to be implemented as Web APIs

Name	Implemented on	Description	Used in EN 16986 Transactions:
EETS Acknowledgements TC	TC	Generic acknowledgement from TSP. This allows third party suppliers to validate incoming data which has already been validated at the API level. A single interface will be implemented at the TC to receive Acks from TSPs. The Acks will be used for multiple transactions.	BILLINGDETAILS_TC PAYMENTCLAIM REPORTABNORMALOBE
EETS Acknowledgments TSP	TSP	Generic acknowledgement from TC A single interface will be implemented at the TSP to receive Acks from TCs. The Acks will be used for multiple transactions.	TOLLDECLARATIONS_SECT EXCEPTIONLIST* PAYMENTANNOUNCEMENT

EETS Toll Declaration	TC	Allows TSP to send road usage data in the form of Toll Declarations to the TC	TOLLDECLARA-TIONS_SECT
EETS Billing Details	TSP	Allows the TC to send Billing Details to the TSP	BILLINGDETAILS_TC
EETS Incremental Exception List	TC	Allows the TSP to send full exception lists (blacklists and whitelists) to the TC	EXCEPTIONLIST
EETS Payment Claim	TSP	Allows the TC to send a Payment Claim (invoice) to the TSP	PAYMENTCLAIM
EETS Payment Announcement	TC	Allows the TSP to announce to the TC that an invoice (payment claim) has been paid	PAY-MENTANNOUNCE-MENT
EETS Contract Issuer List	TC	Allows the TSP to send contractual information regarding OBE to the TC (if implemented)	CONTRACTISSU-ERLIST
EETS CCC_Data_Request	TSP	Allows the TC to request CCC data from the TSP	CCCDATAEXCHANGE (not EN 16986)
EETS CCC Data Response	TC	Allows the TSP to send a response to the CCC_Data_Request to the TC	CCCDATAEXCHANGE (not EN 16986)

* See detailed description of the EETS Exception Lists for more information regarding the acknowledgement of Exception Lists.

Each interface will be implemented as a REST web service. Support for Trust Object and Actor Table Exchange transactions will not use REST interfaces. See sections 0 and 6.13 for further details.

To allow the TC to send messages to the TSP, the TSP must therefore implement the following REST interfaces:

- EETS Ack TC
- EETS Billing Details
- EETS Payment Claim
- EETS CCC Data Request (only if TSP supports OBE Type 2)

The TC will implement the following interfaces to allow the TSPs to send data to the TC:

- EETS Ack TSP
- EETS Toll Declaration
- EETS Incremental Exception List
- EETS Payment Announcement
- EETS CCC Data Response
- EETS Contract Issuer List

4.3 Interfaces, Transactions and ADUs

It is important to understand the relationship between transactions, ADUs and interfaces. To gain a fuller understanding of these relationships, a complete understanding of the relevant standards (ISO 12855 and EN 16986) is essential. The reader is referred to these standards for further details.

4.4 Validation

All data transfers will use acknowledgements to validate a message. However, the acknowledgements are implemented at two levels, the API level and the ADU level.

- All messages in both directions (to the TC and from the TC) will be acknowledged at the API level, using the standard HTTP response codes (See the Developer Portal for response code used). This acknowledgement confirms that the HTTP message has been successfully received and conforms to the required message format.
- For some messages, the API response will be the only acknowledgement used. The sender must therefore assume that, if a HTTP response code 200 is received, the message has been correctly received and will be processed. The messages which only use the HTTP acknowledgements are:
    - Contract Issuer List (TSP – TC)
    - CCC Data Request (TC – TSP). Note that this request expects a CCC Data Response from the TSP, which can be seen as an implicit acknowledgement
    - CCC Data Response (TSP – TC)
    - Acknowledgement TC (TSP – TC)
    - Acknowledgement TSP (TC – TSP). Note that the Acknowledgement TC and Acknowledgement TSP messages are the acknowledgement mechanism used to confirm messages at the ADU level. See the next bullet point regarding ACK ADUs.
- Some messages will be explicitly acknowledged with a separate ACK ADU, as described in ISO 12855:2022. The ACK ADU is used to confirm that, in addition to conforming to the format requirements (as confirmed by the HTTP response), the ADU contents have been checked and the result of content validation is stated in the ACK ADU. The messages which require a specific ACK ADU are:
    - Toll Declaration (TSP – TC)
    - Exception Lists (TSP – TC)
    - Payment Announcement (TSP – TC)
    - Billing Details (TC – TSP)
    - Payment Claim (TC – TSP)

Resend on ADU level with actionCode *resend* is not allowed in any ADU in the KmToll Scheme.

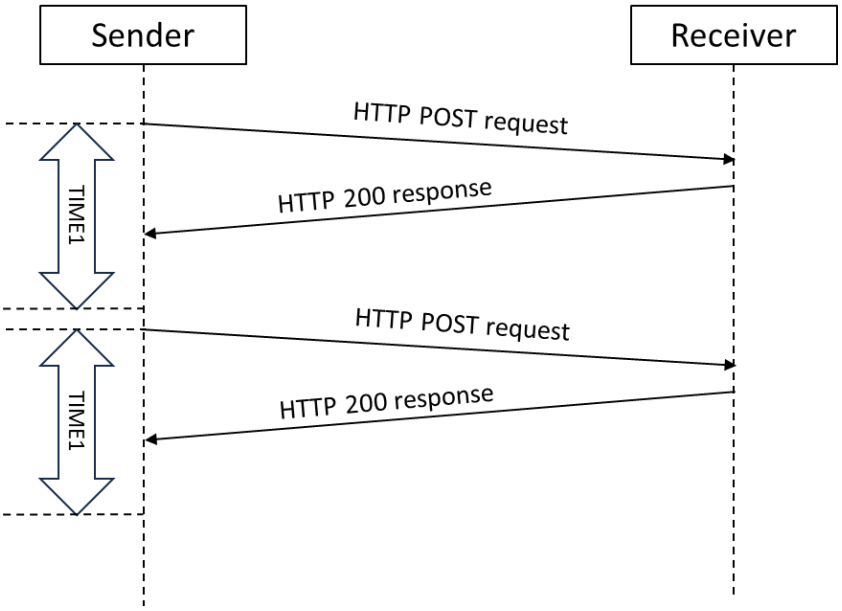
4.4.1 Validation at the API level

At the API level, all messages are checked for conformance to the message formats defined in the relevant schema definitions. Any messages failing this validation check will be rejected at the API level. It is the responsibility of the sender to ensure the appropriate action to be taken, based on the reason for the rejection. More details on the rejection at the API level are given in section 5.5. Messages rejected at the API level will NOT be further processed, and will thus not result in an ACK_TC or ACK_TSP being sent.

At the API level, all data exchanges make use of the HTTPS protocol, using HTTP commands which are responded to in the normal way, using response code 200 for positive acknowledgements. All other response codes must be treated as a transmission or payload failure and are not processed further. See the Developer Portal for response codes used.

A typical HTTP exchange is shown below:

Figure 13. Typical HTTP message exchange



The HTTPS message handling follows normal HTTP protocols:

- The sender sends an HTTP POST request
- The receiver must respond within the TIME1 timeout period, which is set to 30 seconds. The sender should consider a timeout as a non-successfully sent payload and retry up to a maximum of five times.
- Responses other than 200 must be dealt with as defined in section 5.5.

Messages validated at the API level (i.e. those resulting in a Successful Response as described in section 5.5 below) will be passed on to the back-end processes, where the contents of the message may be subject to further validation. These validation checks are more comprehensive than those at the API level, see section 4.4.2 below.

HTTP error codes which are common all interfaces are shown in the table below.

Table 4. HTTP error codes

Data	Validation	Acknowledgement
All data fields	The format of the message received was not understood by the recipient.	HTTP 400 Bad request
adus: actionCode	The sent action code is not supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

4.4.2 Validation at the ADU level

ADU level acknowledgments follow the procedures defined in ISO 12855 and related standards, using the Acknowledgement ADU (i.e. the ACK_TC or ACK_TSP message as appropriate).

Messages requiring an Acknowledge ADU message must first have been successfully received and acknowledged at the HTTP level with a 200 HTTP response. However, the reception of an HTTP 200 response code only confirms that the message has been received and is in the correct format. The sender must wait for an Acknowledge ADU before it can assume that the message has been processed in according to purpose.

The Acknowledgement ADU responds to incoming messages in one of two ways:

- The entire message is accepted.
- The entire message is rejected, with a reason code and, if appropriate, one or more Issue Codes. Note that ISO 12855 allows for the partial rejection of messages with this mechanism, for example by accepting some ADUs and rejecting others. In the KmToll Scheme partial acceptance of messages will not be supported, hence the entire message will be rejected.

Messages are negatively acknowledged at the ADU level where the receiver cannot properly interpret the contents of the message, or does not agree with the contents of the message. Unlike messages rejected at the HTTP level, resending a message which has been negatively acknowledged is not allowed in any interfaces on ADU level as it is unlikely to have a different outcome. So, the sender must instead take action to investigate the reason for the negative acknowledgement. If sender can solve issue themselves the message can be send as a new message with a new APDU identifier. Else the sender must raise an incident ticket.

Note that while ISO 12855 allows for partial acceptance of ADUs, this is not allowed in KmToll – ADUs are either accepted or rejected in their entirety.

The sender is expected to take the following actions on receipt of a negative acknowledgement:

1. Decode the reason and issue codes contained in the negative acknowledgement. Investigate whether the reason for the rejection can be acted on and "re-sent" as a new message(s) with new APDU the relevant message(s), once appropriate changes have been made. The reason codes are described in the relevant standards and in this document.
2. If the sender is not able to make the changes, a support case should be raised on the helpdesk of the receiver.

If an Acknowledge ADU is not received at all, the original sender should, after a suitable timeout period, raise a support request with the receiver. Resending of un-acknowledged messages is not allowed.

4.5 Synchronous and Asynchronous Acknowledgements

Acknowledge ADUs can be either synchronous (the Acknowledge ADU must be received before a subsequent message of the same type can be sent), or asynchronous (the sender may send multiple messages of the same type without waiting for an Acknowledge ADU).

4.5.1 Synchronous Acknowledgements at the ADU protocol level

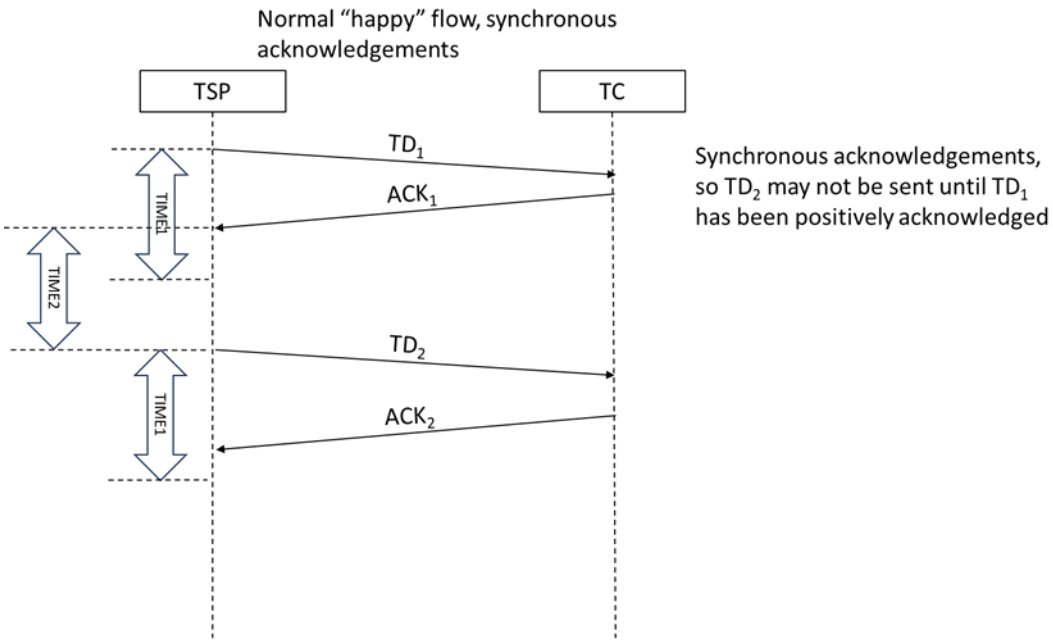
In this section we cover the requirements for synchronous acknowledgements. The only message type which uses synchronous acknowledgements is the Toll Declaration, so this will be used as the example.

Note that the synchronous flow ONLY relates to Toll Declarations for the same Unique Vehicle Identifier (UVI – see Annex E for the definition of UVI). Toll Declarations for different UVIs may be sent in parallel.

4.5.1.1 Message flow for successful flow

Using the Toll Declaration as an example, a successful flow is illustrated below:

Figure 14. "Happy" flow, synchronous acknowledgements



The rules are as follows:

- The synchronous flow ONLY relates to toll declarations for the same UVI. Toll declarations for different UVIs may be sent in parallel.
- TSP sends a Toll Declaration (TD1).
- The TSP expects to receive an Acknowledge ADU within TIME1 timeout period (TIME1_MAX in EN 16986).
- TIME1 maximum is 1 minute.
- TSP should not send a subsequent TD2 until TD1 has been positively acknowledged.
    - In the specific case of toll declarations, the TDs must be sent in sequential order based on the GNSS timestamps.
    - Should the TSP send TD2 if TD1 has not been acknowledged, TD2 will be acknowledged and used. If TD1 is subsequently sent, it will be negatively acknowledged as being out of sequence and will not be further processed.

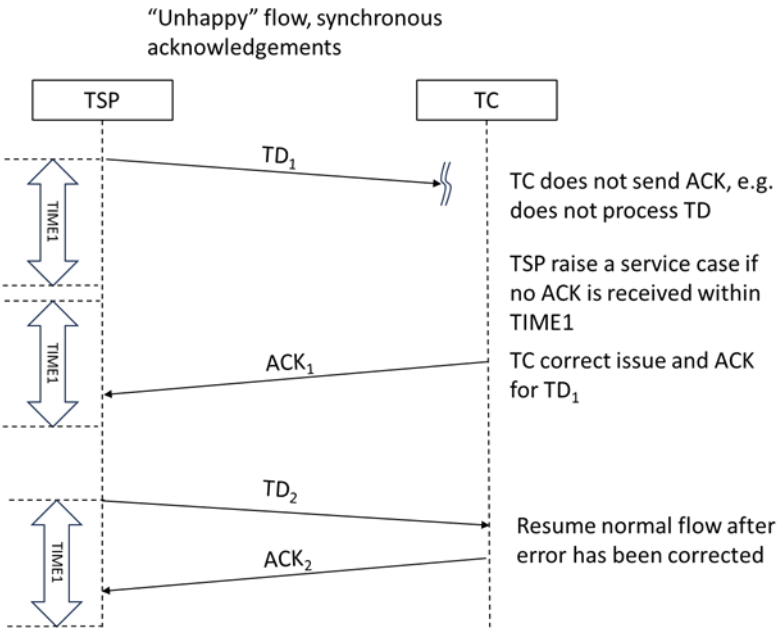
4.5.1.2 Message flows for unsuccessful flow

There are multiple possible "unhappy" flows.

1) The TD is not acknowledged within the timeout period

The TC has received the message (The message was acknowledged with HTTP response 200), but an ACK has not been received or processed by the TSP, as shown below.

Figure 15. "Unhappy" flow 1, synchronous acknowledgements

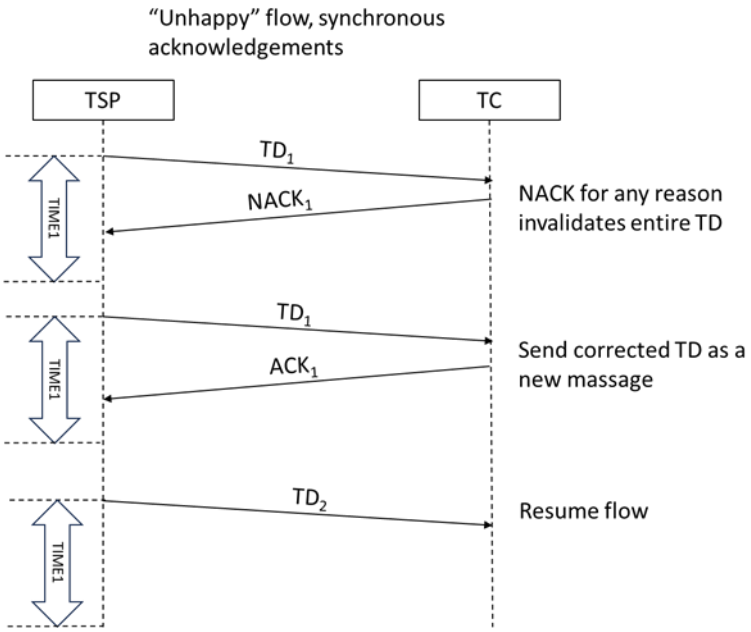


The rules are as follows:

- The TSP sends a Toll Declaration (TD1) which is positively acknowledged at HTTP level.
- If after TIME1 timeout an Acknowledge ADU is NOT received or processed
  - Raise a fault notification with the TC
  - Wait for resolution from TC.
  - TC will send ACK when resolved.
  - After ACK resume transfer of TDs on specific UVI
- If the issue cannot be resolved, TC will instruct TSP to restart sending. TD1 should be re-sent as a new message with new AduIdentifier and ApduIdentifier (under the KmToll rules, once a message has been acknowledged at the HTTP level, its ApduIdentifier cannot be re-used)

2) The TC negatively acknowledges a TD

Figure 16. "Unhappy" flow 2, synchronous acknowledgements

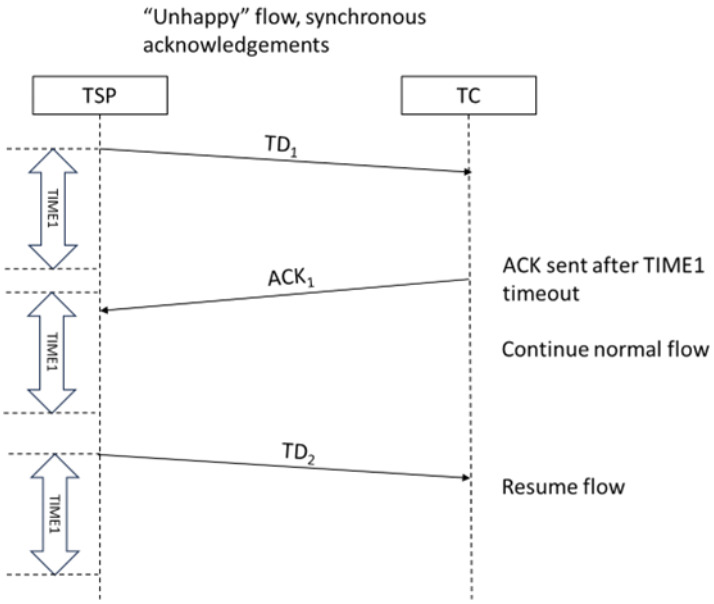


The rules are as follows:

- TSP sends a Toll Declaration (TD1) which is received by the TC and negatively acknowledged.
- The TSP must examine the reason code for the NACK.
- If appropriate, and after necessary corrections, the TSP sends TD1.A new ApduIdentifier and AduIdentifier are required.
- If a positive ACK is received, the TSP can now assume that the TC has received TD1 and continue with the normal flow.

3) The TC acknowledges a TD, but it is not received in time by the TSP

Figure 17. "Unhappy" flow 3, synchronous acknowledgements



The rules are as follows:

- TSP sends a Toll Declaration (TD1) which is received by the TC and positively acknowledged at http level.
- The ACK sent by the TC is received by the TSP after the TIME1 timeout period. Under the rules of the above scenarios, the TSP should have raised a service case.
- The TSP can now assume that the TC has received TD1 and continue with the normal flow. and close e.g. service case.

4.5.2 Asynchronous Acknowledgements at the ADU protocol level

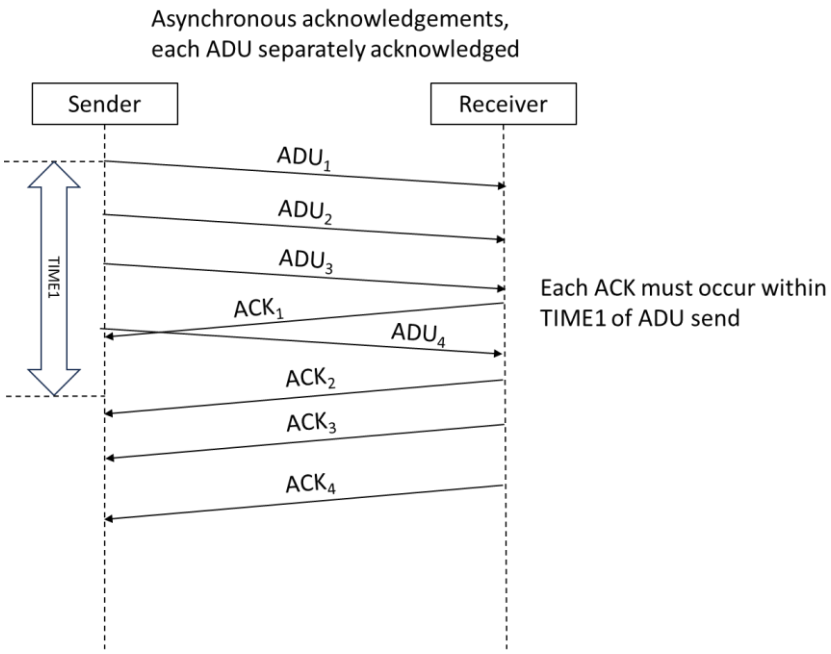
In this case the sender does not need to wait for an acknowledgement before sending the next payload. This also allows the receiver to acknowledge multiple messages with a single acknowledgement for chunked massages, as will be described below.

Asynchronous acknowledgements are used for the following message types:

- Exception Lists (TSP – TC)
- Payment Announcement (TSP – TC)
- Billing Details (TC – TSP)
- Payment Claim (TC – TSP)

Most messages (Payment Announcement, Billing Details, and Incremental Exception Lists) require that each message is individually acknowledged (i.e., a separate acknowledgement must be sent for each for each message), as shown below.

Figure 18. Asynchronous acknowledgements



The rules are as follows:

- The sender may send multiple ADUs simultaneously, or in quick succession, i.e., the value for TIMEA_MIN in EN 16876 is zero.
    - There may be limit to the number of simultaneous connections allowed at the API level.
- Each ADU must be acknowledged with a separate Ack ADU within a timeout period of TIME1

Maximum default values for TIME1 are shown in the table below.

Table 5. Maximum default values for TIME1

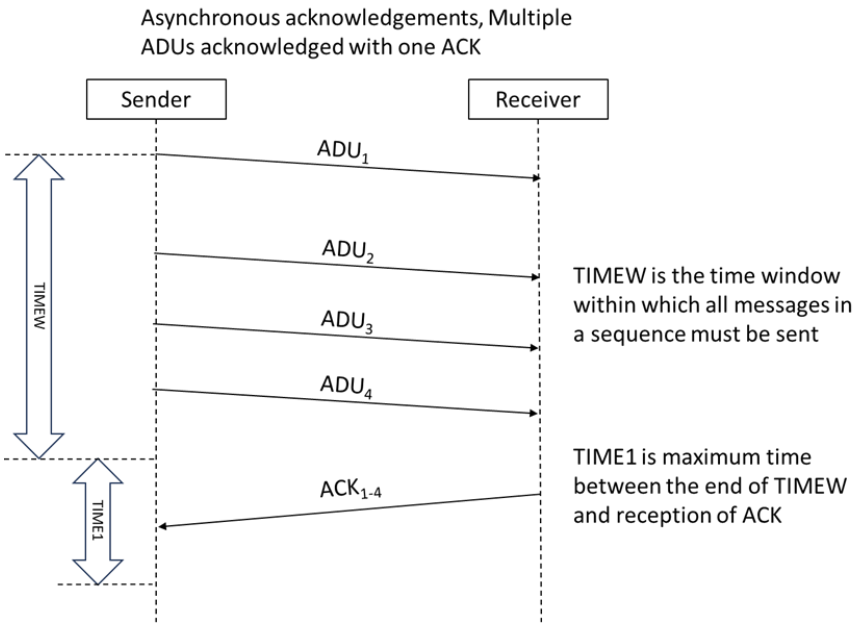
ADU	TIME1_Min	TIME1_Max
Payment Announcement	0	1 hour
Billing Details	0	1 hour
Incremental Exception Lists	0	1 hour

Negative asynchronous acknowledgments follow the same logic as described in the section for synchronous acknowledgements above.

4.5.3 Acknowledgement of multiple ADUs with a single Acknowledge ADU

All messages described above require a separate Acknowledge ADU for each message. However, the messaging protocol allows for a single Acknowledge ADU to acknowledge multiple messages of the same type. This is shown in the figure below.

Figure 19. Asynchronous acknowledgements



The rules are as follows:

- The sender may send multiple ADUs simultaneously, or in quick succession, i.e., the value for TIMEA_MIN in EN 16876 is zero.
- There may be limit to the number of simultaneous connections allowed at the API level.
- The sequence of messages is sent within a defined time window TIMEW. This time window can be pre-defined as will be the case for full exception lists or can start with the transmission of the first message in a sequence as will be the case for payment claims.
- All ADUs may be acknowledged with a single Ack ADU within a timeout period of TIME1 from the end of the time window within which the sequence of message can be sent. Data within the ADU will allow the receiver to determine if the full sequence of ADUs has been received.
- A multiple ACK ADU must be split into several if they contain more than 10.000 acknowledgements.

Note that at this time only Payment Claim messages allow the use of a single acknowledgment for multiple ADUs.

Maximum default values for TIME1 and TIMEW are shown in the table below

Table 6. Maximum default values for TIME1 and TIMEW

ADU	TIME1_Min	TIME1_Max	TIMEW
Payment Claim	0	1 hour	1 hour

4.6 Coding of Acknowledge ADUs

When a message is acknowledged using the Acknowledge ADU, the Acknowledge is in itself also an InfoExchange message, following a similar layout and principle, as shown below in the ackTspAdu example:

Sample 2

```
"InfoExchange":
    "InfoExchangeContent": {
        "apci": {
            "aidIdentifier",
            "apduOriginator",
            "informationSenderID",
            "informationrecipientID",
            "apduIdentifier",
            "apduDate"
        }
        "adus":
            "ackAdus": { "type": "array",
                "aduIdentifier",
                "apduAckCode",
                "issues":{ "type": "array",
                    "issueAduIdentifier",
                    "issueLocation",
                    "issueContent",
                    "issueCode",
                    "issueText"
                }
                "actionCode"
            }
    }
```

Note that multiple "ackAdus" are possible in the "adus" section, and that multiple "issues" are allowed per ADU.

The ackAdu is used as follows:

4.6.1 The entire message being acknowledged is positively accepted

In this case, both the message format and content are accepted. The apduAckCode is given a value of "2" as defined in ISO12855:2022, Table 9 "apduOk", and there are no entries in the issues array fields.

An example is shown below:

Sample 3

```
"InfoExchange": {
    "InfoExchangeContent": {
        "apci": {
            "aidIdentifier": 32,
            "apduOriginator": {
                "countryCode": "1001011110",
                "providerIdentifier": 5
            },
            "informationSenderID": {
                "countryCode": "1001011110",
                "providerIdentifier": 5
            },
        "informationrecipientID": {
            "countryCode": "1111010011",
            "providerIdentifier": 495
        },
        "apduIdentifier": 987,
        "apduDate": "20230727093000Z"
        },
        "adus": {
            "ackAdus": [
                {
                    "apduIdentifier": 8900,
                    "apduAckCode": 2,
                    "actionCode": 0
                }
            ]
        }
    }
}
```

The following rules apply for handling the reception or sending of an AckAdu with acceptance:

- If the apduAckCode field indicates acceptance (value of 2 "apduOk"), the referred message both APCI and APDU are accepted (syntactically and semantically correct).
- There is no issues array.

4.6.2 The entire message being acknowledged is rejected without ADU issues.

This will normally be due to an error in the apci section, which will cause the message to be rejected irrespective of the contents of the ADU. In this case, the apduAckCode is given any value other than "2" as defined in ISO12855:2022 Table 9, and there are no entries in the issues array fields.

The following rules apply for handling the reception or sending of an AckAdu with rejection:

- If the apduAckCode field indicates rejection (value of apduNotOk), the whole referred APDU and APCI is rejected.
- There is no issues array.

An example is shown below:

Sample 4


```
"InfoExchange": {
    "InfoExchangeContent": {
        "apci": {
            "aidIdentifier": 32,
            "apduOriginator": {
                "countryCode": "1001011110",
                "providerIdentifier": 5
            },
            "informationSenderID": {
                "countryCode": "1001011110",
                "providerIdentifier": 5
            },
        "informationrecipientID": {
            "countryCode": "1111010011",
            "providerIdentifier": 495
        },
        "apduIdentifier": 9871,
        "apduDate": "20230727093000Z"
        },
        "adus": {
            "ackAdus": [
                {
                    "apduIdentifier": 89001,
                    "apduAckCode": 3,
                    "actionCode": 0
                }
            ]
        }
    }
}
```


4.6.3 The message as received has a valid format and the apci information is valid, but there are one or more errors in the message payload.

In this third scenario, the message being acknowledged is correctly formatted and addressed, but there are one or more issues in the ADU.

KM Toll will not support partial acceptance of ADUs – if an issue is identified, the entire payload is rejected, hence the apduAckCode field will always contain the value "3", indicating rejection of the message,

The following rules apply for handling the reception or sending of an AckAdu with rejection of the ADU:

- The apduAckCode is given a value of "3", indicating that the message is rejected
- The issues field contains at least one entry, which is coded according to Table 11 in ISO 12855:2022.

An example is shown below, in which apduidentifier "8899" is being acknowledged. The adu with identifier "92233720" has been detected as a duplicate toll declaration, hence the issuecode has value "600", which we can see in ISO 12855:2022 is the value for "A duplicate of the tollDeclarationId has been identified." The entries issueLocation, issueContent and issueText should be used to provide more information as appropriate. issueLocation, and at least one of issueContent or issueText are required.

Sample 5


```
"InfoExchange": {
    "InfoExchangeContent": {
        "apci": {
            "aidIdentifier": 32,
            "apduOriginator": {
                "countryCode": "1001011110",
                "providerIdentifier": 8
            },
            "informationSenderID": {
                "countryCode": "1001011110",
                "providerIdentifier": 5
            },
            "informationrecipientID": {
                "countryCode": "1111010011",
                "providerIdentifier": 495
            },
            "apduIdentifier": 987,
            "apduDate": "37781208110927Z"
        },
        "adus": {
            "ackAdus": [{
                "apduIdentifier": 8899,
                "apduAckCode": 3,
                "issues": [{
                        "issueAduIdentifier": 92233720,
                        "issueLocation": "JSONPath expression to the location of the issue",
                        "issueContent": "Optional descriptive text",
                        "issueCode": 600,
                        "issueText": "Duplicate toll declaration detected"
                }],
                "actionCode": 0
            }]
        }
    }
}
```

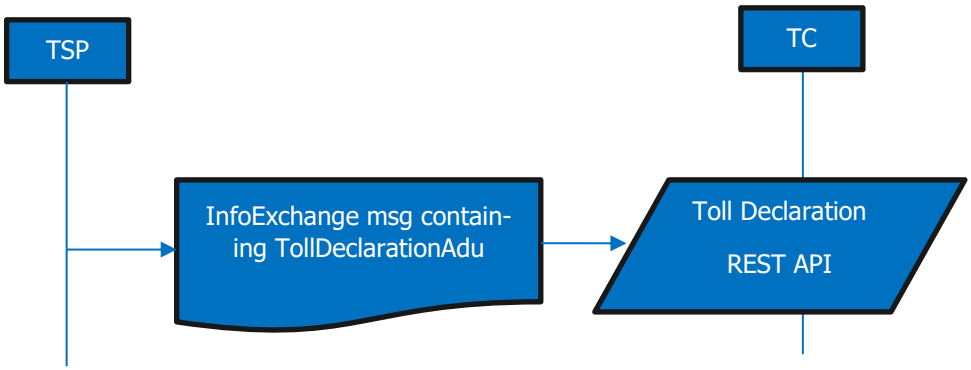

4.7 Example of a TSP to TC transaction using REST

A TSP wishes to send a Toll Declaration to the TC. For this, they will use the TOLLDECLARATIONS_SECT transaction.

The data to be transmitted is coded into a TollDeclarationADU. The TollDeclarationAdu is packaged with the appropriate APCI into an InfoExchange message.

To send the InfoExchange message, the TSP connects to the REST API service interface for Toll Declarations, provided by the TC, and posts the message to the API, as shown below.

Figure 20. InfoExchange message example, Toll Declaration

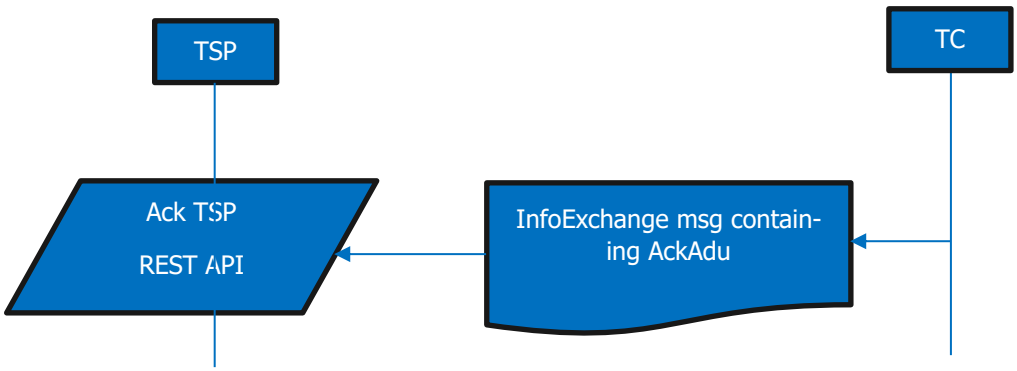


If the message is received correctly, the API will respond with the relevant acknowledgement at the API level. As described above in 4.4, this is only acknowledging that a correctly formatted message has been received; a separate Acknowledgement ADU is used to inform the TSP that the Toll Declaration, containing valid tolling data, has been received by the TC. This is described below.

The TC will process the incoming Toll Declaration, and once it is satisfied that the ADU contains valid Tolling data, it will then initiate an Acknowledge transaction with the relevant TSP.

To send the Ack message, the data to be transmitted is coded into an AckAdu. The AckAdu is packaged with the appropriate APCI into an InfoExchange message. TC will make a backend REST API call to the API service provided by TSP for TC-initiated Ack messages. The TSP will retrieve the Ack message from the Ack TSP API service, as shown below. Note that the AckAdu may not be required in all cases. If the validation undertaken by the REST API fulfils all validation requirements, the AckAdu will not be required. This will be further elaborated in a later version of this document.

Figure 21. InfoExchange acknowledgement example



For cases where the TC wishes to send data to the TSP, for example Billing Details, the above sequence is used, but in reverse; the TC creates the relevant Billing Details InfoExchange message to the Billing Details REST API service implemented by the TSP. Once the message has been processed, the TSP will push an Ack message to the relevant Ack TC REST API service.

In all cases, the message receiver acts as the server – messages are pushed from sender to receiver.

4.8 Acknowledgement of chunked payloads

For some of the interfaces there is a need to split the payload into several chunks as the data size can be larger than what is suitable for API interfaces. This will be implemented for list-based interfaces such as Payment claim.

For example, a Payment Claim is split into multiple chunks but is still considered as one Payment claim and must therefore be acknowledged as a whole.

There are two ways to acknowledge a chunked massage, either:

- Each chunk is acknowledged individually with an acknowledge message containing an Acknowledge ADU. When all chunks have been acknowledged the assessment of whether the entire message is accepted or rejected is performed. To be accepted, all chunks must be positively acknowledged apduAckCode value of "2" (apduOk). If any one chunk is rejected with apduAckCode value of "3" (apduNotOk), the entire payment claim is rejected.
- The acknowledge ADUs for all the chunks can be compiled in to a single acknowledge message payload where reference to all APDUs in the Payment claim are listed in "ackAdus" with potential issues listed in "issues". If there is an issue with any chunk, the entire payment claim is rejected.

The following rules apply for handling the reception or sending of an AckAdu:

- Each chunk of the message must be acknowledged referring to the "apduIdentifier" of the acknowledged chunk.
- If the apduAckCode field indicates rejection (value of apduNotOk) for one or more chunks, the whole message (i.e. all of the chunks which make up the message) is rejected.
- The required fields "issueAduIdentifier", "issueLocation", "issueContent", "issueCode" and "issueText" must be filled with error information and location.
- All apduAckCodes other than '2' from ISO12855 tabel 9 serves as a rejection of the entire messages on both APCI and APDU level.

Shown below is an example of an acknowledge message positively acknowledging 2 ADUs, identified by the apduIdentifiersI 8899 and 8900. If these two apduIdentifiers represent the entirety of a chunked message, then the entire chunked message is accepted.

Sample 6


```
"InfoExchange": {
  "InfoExchangeContent": {
    "apci": {
      "aidIdentifier": 32,
      "apduOriginator": {
        "countryCode": "1001011110",
        "providerIdentifier": 5
      },
      "informationSenderID": {
        "countryCode": "1001011110",
        "providerIdentifier": 5
      },
      "informationrecipientID": {
        "countryCode": "1111010011",
        "providerIdentifier": 495
      },
      "apduIdentifier": 987,
      "apduDate": "20230727093000Z"
    },
    "adus": {
      "ackAdus": [
        {
          "apduIdentifier": 8899,
          "apduAckCode": 2,
          "actionCode": 0
        },
        {
          "apduIdentifier": 8900,
          "apduAckCode": 2,
          "actionCode": 0
        }
      ]
    }
  }
}
```

Shown below is an example of an acknowledge message which positively acknowledges one APDU (9899), and negative acknowledges another (9900). If these two APDUs form part of a chunked message, the entire chunked message is rejected.

Sample 7


```
"InfoExchange": {
  "InfoExchangeContent": {
    "apci": {
      "aidIdentifier": 32,
      "apduOriginator": {
        "countryCode": "1001011110",
        "providerIdentifier": 5
      },
      "informationSenderID": {
        "countryCode": "1001011110",
        "providerIdentifier": 5
      },
      "informationrecipientID": {
        "countryCode": "1111010011",
        "providerIdentifier": 495
      },
      "apduIdentifier": 987,
      "apduDate": "202307270930002"
    },
    "adus": {
      "ackAdus": [
        {
          "apduIdentifier": 9899,
          "apduAckCode": 2,
          "actionCode": 0
        },
        {
          "apduIdentifier": 9900,
          "apduAckCode": 3,
          "issues": [
            {
              "issueAduIdentifier": 92233720,
              "issueLocation": "JSONPath expression to the location of the issue",
              "issueContent": "Optional descriptive text",
              "issueCode": 800,
              "issueText": "Optional explanatory text"
            }
          ],
          "actionCode": 0
        }
      ]

      ]
    }
  }
}
```

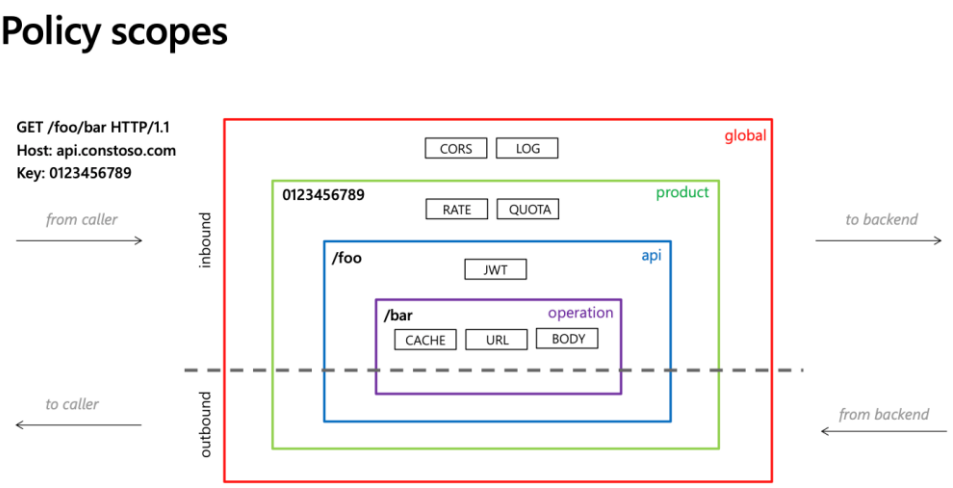

5 Common aspects for all interfaces

This chapter contains the specifications which are common to all the interfaces.

5.1 Control access

Each API will have so called "policies" applied, the scope of these policies is divided into different scopes: global, product and api, and ensures that each API will confine to the governance set by the TC. An example could be limiting to how many times a TSP can call each API within a specified time (throttling), and rerouting policies, as well the logging mechanism.

Figure 22. Policy scopes



Some of the common polices are listed below and more details on what policies are enforced will be provided on request and if needed:

Figure 23. Common policies

Access restriction	Transformation	Advanced	Dapr integration
• Check HTTP header • Limit call rate by subscription • Limit call rate by key • Restrict caller Ips • Set usage quota by subscription • Set usage quota by key • Validate client certificate • Validate JWT	• Convert JSON to XML • Convert XML to JSON • Find and replace string in body • Mask URLs in content • Set backend service • Set body • Set HTTP header • Set query string parameter • Rewrite URL • Transform XML using XSLT	• Send one way request • Send request • Set HTTP proxy • Set variable • Set request method • Set status code • Control flow • Emit metric • Log to Event Hub • Trace • Mock response • Forward request • Limit concurrency • Return response • Retry • Wait	• Send request to a service • Send message to a pub/sub topic • Trigger output binding
Authentication	**Caching**	**Cross Domain**	**Validation policies**
• Authenticate with basic • Authenticate with client certificate • Authenticate with managed identity	• Get from cache • Store to cache • Get value from cache • Store value from cache • Remove value from cache	• Allow cross-domain calls • CORS • JSONP	• Validate content • Validate parameters • Validate headers • Validate status code • Validate GraphQL request

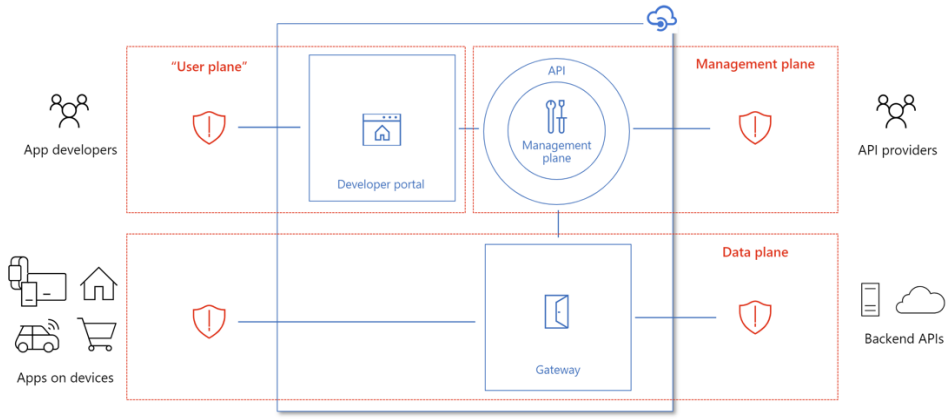
5.2 Security & Authentication for REST APIs

The API endpoints will be secured using HTTPS, this will protect authentication credentials like API keys, passwords and JSON Web Tokens (JWT). The HTTP will also include a timestamp in the request, so the server can compare the request timestamp and accept the request only within a certain timeframe, for instance one or two minutes. This eliminates cases of replay attacks from hackers trying to brute force the system.

The following diagram is a conceptual view of Azure API Management, showing the management plane (Azure control plane), API gateway (data plane), and developer portal (user plane), each with at least one option to secure interaction.

The scope of this document is to cover the "User plane" accessing the Developer portal and the "Data plane" calling the APIs through the gateway as TSP provider.

Figure 24. API Management



When a TSP wishes to gain access to an API product, an API Management subscription key is required. The subscription key is generated by the TC and distributed securely to the TSP.

The API key type "Ocp-Apim-Subscription-Key" must be included in the HTTP header to the request, passing the value of a valid subscription key.

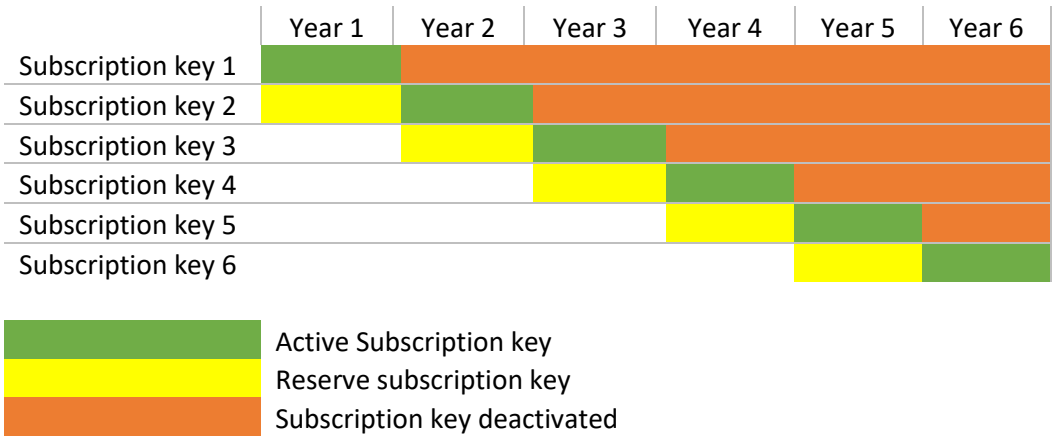
When API Management receives an API request from a client with a subscription key, it handles the request according to these rules:

- Check if it is a valid key associated with an active subscription.
- The subscription is scoped to an API product that is assigned to the TSP.
- If a valid key for an active subscription at an appropriate scope is provided, access is allowed. If a subscription key is incorrect or has expired, the error message "401 Access denied error" will be transmitted.

Subscription keys will have a lifetime of two years.

Each TSP will be provided with two subscription keys, a primary and a secondary. For the first year of operation, the TSP must use the primary key. At the end of each year, the TSP should switch to using the other key, and the first key will be replaced with a new key issued by the TC. This is shown diagrammatically in Table 7 below:

Table 7. Subscription Key Management



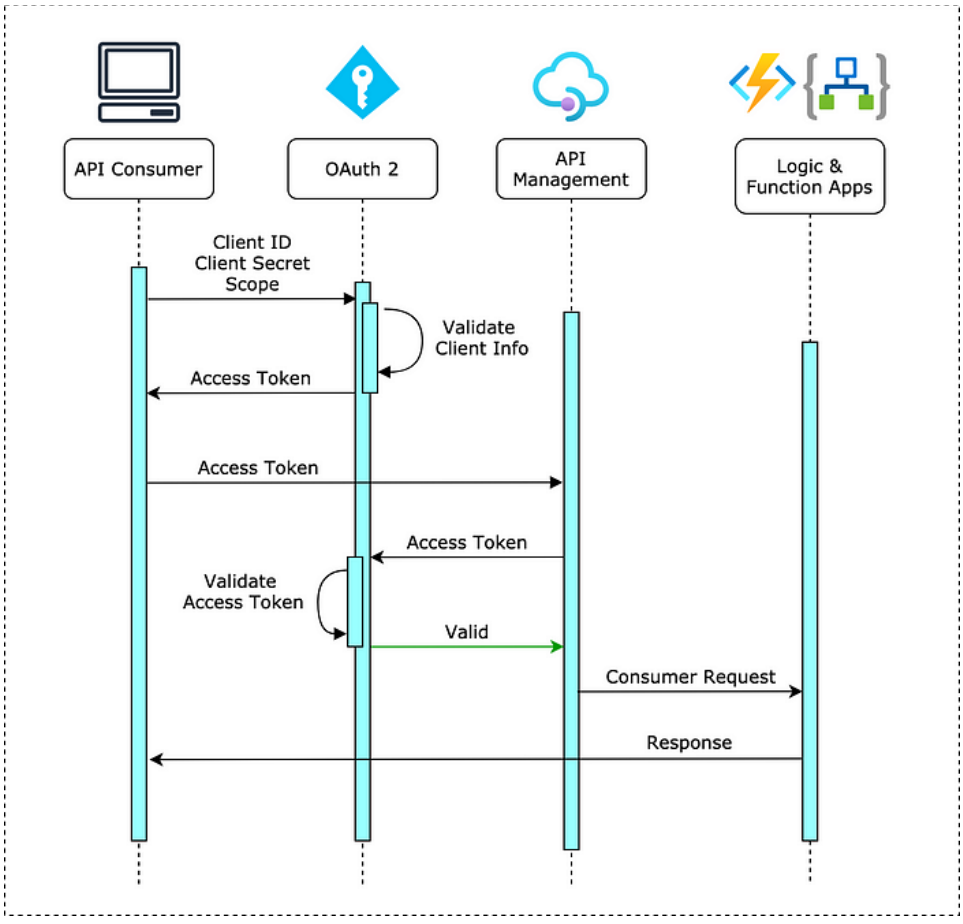
In year 1, two keys are issued, subscription key 1 (Primary key) and subscription key 2 (Secondary key). Subscription key 1 is the active key, and subscription key 2 is the reserve subscription key. At the end of each year, the previous reserve key (key 2) becomes the active key. Subscription key 1 is revoked, and a new subscription key (key 3) is issued and becomes the new reserve key. This is repeated at the end of each year. In this way, keys have a lifespan of 2 years, and the TSP always has 2 keys available for use. The first Primary key will be revoked after one year and therefore only have a life span of one year.

5.2.1 Data plane access

To be able to use the TC provided APIs, API Management has the responsibility to do the initial authentication to prevent unauthorized access. This is done using the standardized OAuth 2.0 flow, API Management can retrieve and refresh access tokens to be used inside of API management or sent back to a client. OAUTH 2.0 is the open standard for access delegation which provides client a secure delegated access to the resources on behalf of the resource owner.

Figure 25 shows the process flow for using the OAuth 2.0 flow for accessing the APIs provided by TC:

Figure 25. Process flow for using the OAuth 2.0

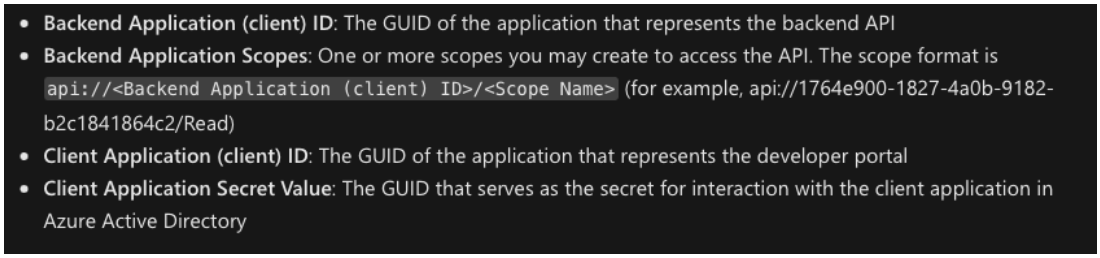


It is a requirement that the TSP as a minimum secures its APIs with the same OAuth flow as shown in Figure 25 and that the TSP register the TC's API Management as client in their OAuth 2.0 Identity provider. TC will keep the keys of the TSP in the TC's Azure Key Vault and will only be used when calling the "backend API" of the TSP.

TSP is required to use Oauth 2.0 "Client Credential flow = machine to machine(M2M). See Oauth 2.0 get-started guide here if needed.

For the purpose of secure exchange, the TSP and TC shall share the following information ('secrets') with each other. The information is to be shared via secure exchange, using a method chosen by the TC:

Figure 26. 'Secrets'



5.2.2 User plane login and subscription key – Developer portal (EDU and UAT environments only)

The TSP is required to provide the TC with an email address to be used for creating an account and providing login access to the Developer Portal.

Only one company-specific email address is allowed per TSP. It is a requirement that the email address is not a 'dead' email, and that the TSP can access and read replies since important information may be received on the email.

The TC is responsible for generating a username and create a user for the TSP. Once a user is created the TSP will receive an email with a link to active the user. The TSP should use the link to generate a password for the TC's organisation.

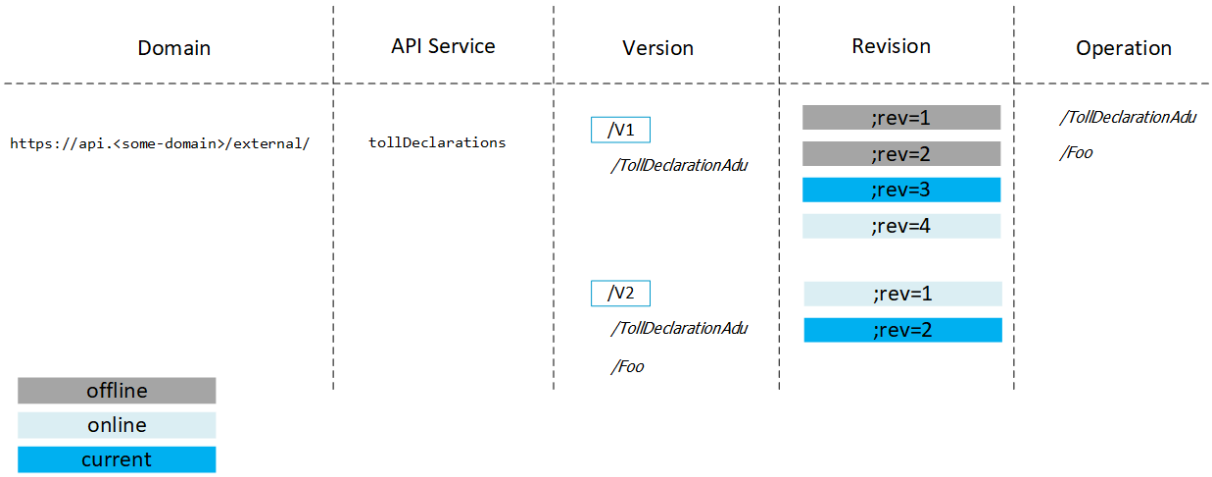
Once access is granted the TC will ensure that the TSP are given access to the relevant product(s) (group of APIs). Then the TSP will have access to the products in the Developer Portal from where they can get the relevant subscription key for the APIs.

It is a requirement that the relevant subscription key be filled out if the TSP wishes to submit request using the 'Try it' function in the Developer Portal.

5.3 Versioning

Each APIs would go through different versions as a natural way to keep them relevant and up to date when modification to existing API or new exchange data between TC and TSP are identified. Below table shows an example how each API URI is represented when changing versions, revisions. The table is also showing that some APIs and revision can be sunset (offline) when they are no longer maintained.

Table 8. Versioning



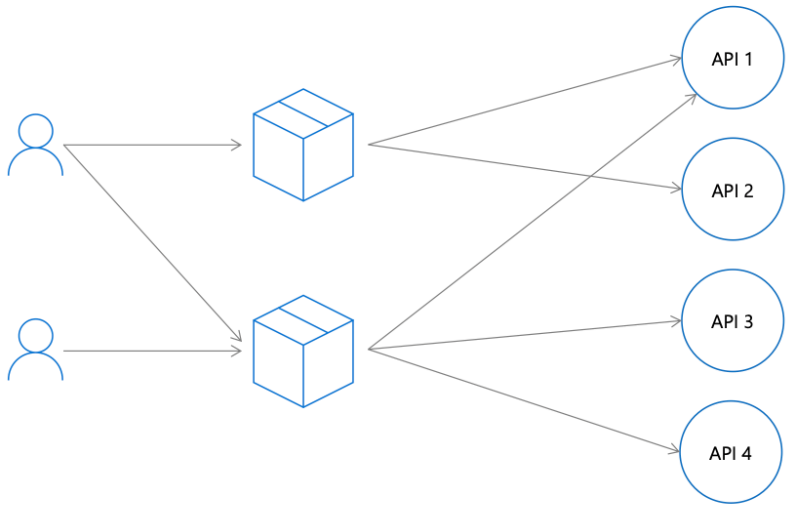
The length of time for which an old version will remain online before being declared offline will depend on the criticality of the version upgrade. Three levels of criticality for incoming interfaces at the TC are defined:

- Critical – a new version is required to correct a defect which directly impacts the correct operation of the system. Critical updates can occur at any time, and the previous version will remain online if possible, for the shortest feasible time, typically one week.
- Urgent – a new version is required to correct a defect which has an impact on the smooth running of the system. Urgent updates can occur at any time, and the previous version will remain online for at least two weeks after release of a new version.
- Routine – an upgrade has been issued with updated or improved functionality. Typical reasons will include implementing changes required by law, changes in scheme rules and IT security etc. Routine upgrades should be expected once per year. The previous version will remain online for at least two months after release of a new version.

5.4 API products

When logging into the developer portal, the APIs would be divided into logical products. These products are only an abstraction layer, and a TSP developer will only see the products that are relevant. APIM is used for API discovery.

Figure 27. Logical products



5.5 Error/validation handling

Basic HTTP response status codes will indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes and returned for as response for a specific API call on the message level, while a specific cause for any error or validation are to be found in the body of the returned call.

Please see the Developer Portal for the implemented HTTP response codes, used by the different APIs.

This schema is composed of five parts:

1. *type* – a URI identifier that categorizes the error.
2. *title* – a brief, human-readable message about the error.
3. *status* – the HTTP response code (optional).
4. *detail* – a human-readable explanation of the error.
5. *instance* – a URI that identifies the specific occurrence of the error.

The body could look like this:

```
{
    "type": "/errors/incorrect-user-pass",
    "title": "Incorrect username or password.",
    "status": 401,
    "detail": "Authentication failed due to incorrect username or password.",
    "instance": "/login/log/abc123"
}
```

Note that the *type* field categorizes the type of error, while *instance* identifies a specific occurrence of the error in a similar fashion to classes and objects, respectively.

These fields provide a client or developer with information to help troubleshoot the problem and also constitute a few of the fields that make up standard error handling mechanisms.

Example of a code 400 error is shown in the code snippet below, where the value expected is not the same as required in the defined 'enum'.

Figure 28. HTTP response - 400 Bad Request

HTTP response

```
HTTP/1.1 400 Bad Request

content-length: 342
content-type: application/json
date: Tue, 19 Sep 2023 11:22:57 GMT
request-context: appId=cid-v1:27ec7f53-2c25-41b7-af6a-8
ff2c674c4d6
requestid: 2802e93a-db4f-44e0-b642-c64d9c0915f9

{
    "Status": "Error",
    "Message": "Format Validation Error",
    "ResponseTime": "2023-09-19T11:22:57.4015517Z",
    "CorrelationId": "2802e93a-db4f-44e0-b642-c64d9c091
5f9",
    "Errors": [{
        "Type": "enum",
        "Field": "/InfoExchange/InfoExchangeContent/apc
i/informationrecipientID/countryCode",
        "Message": "Expected value to match one of the
values specified by the enum"
    }]
}
```

6 Individual interface descriptions

The interfaces between the TC and TSP are defined in this section.

In general, a separate interface is defined for each ADU defined in EN 16986, though there are exceptions to this, for example a separate interface is defined for each of the four supported Exception Lists.

Note that an interface only allows data transfer in one direction. For example, the AckAdu described in section 6.5 of EN 16986 requires two interfaces, one for each direction, so the *EETS Acknowledgements TSP* interface allows the TC to send acknowledgements to the TSP, while the *EETS Acknowledgements TC* interface allows the TSP to send acknowledgements to the TC.

The data formats used in the interfaces described below are defined in EN 16986, which in turn uses the underlying standard ISO 12855:2022. While the standards define the data formats used in the interfaces, they may contain optional elements which are further defined in this section. In addition, some interfaces have additional fields to those defined in the standards. These are also defined in this section.

In addition to the interfaces defined in prEN16986:2023, two additional interfaces are defined, namely EETS CCC Data Request and EETS CCC Data Response. These are required to support app-based OBE which does not have a DSRC interface for compliance checking.

Note that the data formats accessed through and defined in OpenAPI specifications are the definitive definitions. If the data formats and restrictions defined in this document differs from that in OpenAPI specification, the OpenAPI specification takes precedence.

It is important to note that ISO 12855, and hence EN 16986, require that data is encoded using XML, and EN 16986 provides example XML Schema files for each of the profiles defined. The Danish KmToll project has decided that data will be encoded in JSON (JavaScript Object Notation) and defined in appropriate JSON schemas. The JSON schemas for each interface are available in the developer portals after agreement with TC for accreditation and development purposes.

Interfaces described in sections 6.1 to 6.5 are to be implemented at the TSP.

Interfaces described in 6.6 to 6.11 will be implemented at the TC.

6.1 EETS Acknowledgments TSP Interface

6.1.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS Acknowledgments TSP – vn*", where "*vn*" is the version number currently in use.

6.1.2 Purpose of the Interface

The EETS Acknowledgments TSP interface is implemented by the TSP and is used by the TC to acknowledge the correct receipt of message from the TSP. It is never used in isolation and should be considered as a part of the messaging protocol.

Acknowledgements can be synchronous or asynchronous, depending on the message type being acknowledged.

In this version of the specification, the EETS Acknowledgments TSP interface may only be used as part of the following transactions:

- TOLLDECLARATIONS_SECT
- PAYMENTANNOUNCEMENT
- EXCEPTIONLISTS

In future versions, it may be used in other transactions.

6.1.3 Communications Processes

The Acknowledgement will be transferred using a REST Web-Service interface as defined in section 0.

The following additional specifications apply to this interface:

1. The Acknowledgement is used to Acknowledge (positively or negatively) the messages containing other ADUs. It is never used in isolation.
2. Acknowledgments may be synchronous or asynchronous, depending on the transaction. See the descriptions for Toll Declarations, Payment Announcement and Exception List interfaces for more details.

6.1.4 Message Formats

The Ack_TC data is transmitted in an *InfoExchange* message with an *AckAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

The ApduReasonCodes defined in Table 9 of ISO 12855:2022 must be supported.

The AduReasonCodes reported in the *issueCode* field which are defined in Table 11 of ISO 12855:2022 must be accepted by the TSP. While all values in the range 0 – 65535 are allowed, the following specific values are expected to be used in the KmToll system:

- Values 0 – 1 (generic ADU errors)
- Values 600 – 612 (toll declaration related errors)
- Value 3000 (semantic error)
- Value 3010 (action code not supported error)

The following user defined values must also be supported - this list may be further expanded in future versions of this specification:

- Value 10300 (Contract Issuer List ADU not accepted), the data in the Contract Issuer List contains errors or inconsistencies. See the issueText entry for further details.
- Value 10600 (Toll declaration ADU contains out-of-sequence GNSS data), in other words, this ADU contains data which violates the requirement that toll declarations are sent strictly in time-sequence. KmToll will not process the data in this Toll Declaration.
- Value 10601 (Geo-location data is rejected as it is too old). KmToll will not process the data in this Toll Declaration as it contains data which is too old to be processed
- Value 10900 (payment announcement adu not accepted). The Payment Announcement sent by the TSP is rejected. See the IssueText field for further details.

Ranges marked as "reserved" will only be used when defined in future versions of the standards, or their use has been defined in bilateral agreements between the TC and the TSP.

Note further that it is not foreseen that the value 4000 (acceptedWithWarning) will be used, but it may be used in a future version of this interface.

6.1.5 Error Handling

API errors will be handled as described in section 5.5.

6.2 EETS Billing Details interface

6.2.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS Billing Details – vn*", where "*vn*" is the version number currently in use.

6.2.2 Purpose of the Interface

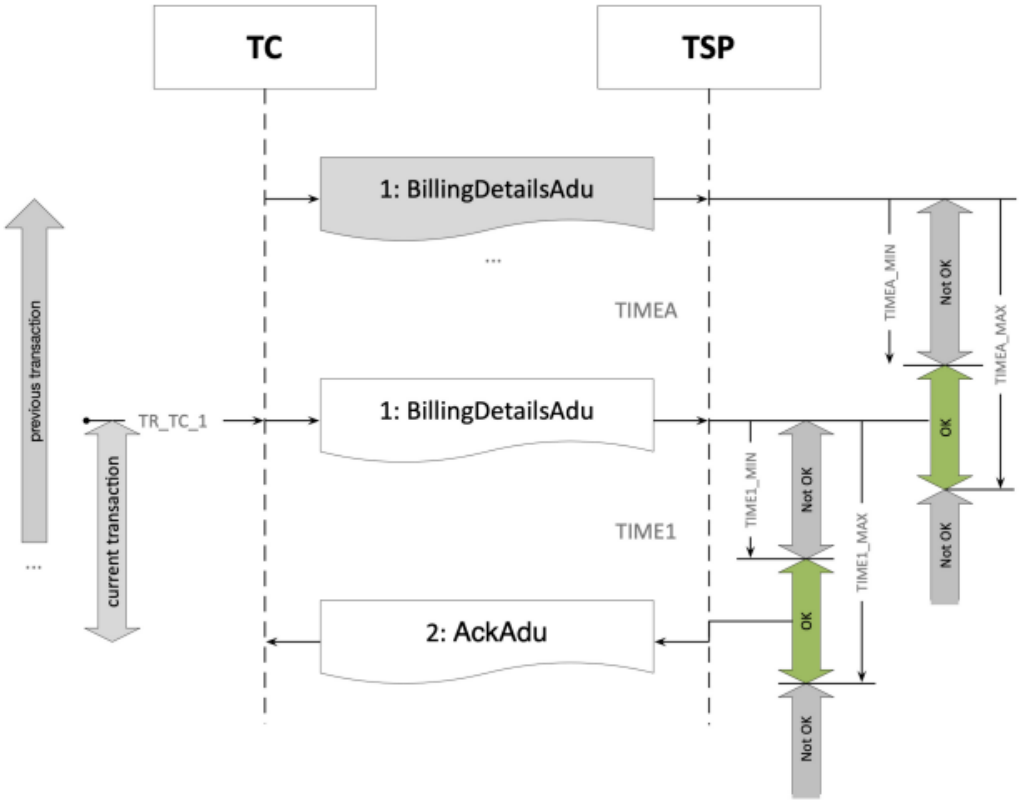
The interface is used by the TC to issue regular Billing Details to the TSP. The Billing Details provides the TSP with details of all Tolls incurred by the EETS Users of the TSP during the billing period covered by the Billing Details.

6.2.3 Communications Processes

The Billing Details will be transferred using a REST Web-Service interface as defined in section 0, and will be Acknowledged (see 6.6).

The following additional specifications apply to this interface – Figure 14 from prEN16986:2023 is reproduced below for reference:

Figure 29. Copy of Figure 14 from EN 16986



1. The TC will issue Billing Details once per billing period, currently assumed to be once every 24 hours. This time is contractually defined.
2. Only one *billingDetailsAdu* is allowed per transaction. This means that a separate Billing Details message will be sent for each Unique Vehicle Identity (UVI). See Annex E for a definition of the UVI.
3. Each Billing Details message must be acknowledged on the EETS Acknowledgements TC interface.
4. Billing details for different users can be sent in parallel. Therefore TIMEA_MIN (prEN16986:2023 Table 57 and Figure 14, is defined as 0s (for different users).
5. TIME1_MIN (prEN16986:2023 Table 57) is defined as 0.
6. TIME1_MAX (prEN16986:2023 Table 57) is set to 24 hours.
7. If TIME1_MAX is reached without an acknowledgement, the billing details may be re-sent. Re-sends will be sent a maximum of twice (i.e., a total of three times) before escalation to the TSP.

6.2.4 Message Formats

The Billing Details is transmitted in an *InfoExchange* message with a *BillingDetailsADU* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

Identification of OBE type

The KmToll Scheme supports both OBE using dedicated hardware and OBE implemented as software on a nomadic platform (e.g., a smartphone), called Type 1 and Type 2 respectively. To identify the OBE type, a new data field called "*obeType"* is introduced as part of the "*obeId"* field within the adu. *obeType* is defined below:

Table 9. obeType

Data Element	Data type	Description
obeType	Integer	Identified the OBE type in use. It can take the values: "1" – OBE is of type 1 "2" – OBE is of type 2 Values between "3" and "127" are reserved for future use.

Coding of *appliedLocationClassId* data field

Provision has been made in the billing details to identify if a charged section is in a low emission zone, and if so, which zone it is in. There are currently five low emission zones defined in Denmark, and they are coded as follows:

Table 10. appliedLocationClassId - Low Emission Zones

appliedLocationClassId	Low Emission Zone applied
0	Untolled
1	Tolled – Not in LEZ
2	Tolled - Odense LEZ
3	Tolled - Aarhus LEZ
4	Tolled - Aalborg LEZ
5	Tolled - Copenhagen LEZ
6	Tolled - Frederiksberg LEZ

Note that *appliedLocationClassId* value "0", indicating that the section is not tolled, is not currently used in the KmToll, but may be used in the future.

Use of the actionCode field

When sending Billing Details to the TSP, the field *actionCode* will normally contain the value *send*, which indicates that this is a new Billing Details which needs to be processed by the TSP. However, it could happen that a previously sent Billing Details needs to be updated, for example following a successful complaint by an end user. This will be achieved by re-sending the Billing Details which are to be changed in a new *billingDetails* message. The re-sent Billing Details will contain the same *aduIdentifier* as the Billing Details being changed, but with the *actionCode* value *revoke,* which indicates to the TSP that the Billing Details should be cancelled. A new updated Billing Details will be sent in a new *billingDetails* message. More details on revoking of Billing Details can be found in Annex E of the Toll Domain Statement.

6.2.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 11. Error codes - Billing Details interface

Data	Validation	Acknowledgement
Adus: **BillingDetailAdus:** **UserId:** **pan** **LPN** **obeId**	The user ID in billing details has been rejected. Validate that the user information is according to Toll Declaration(s)	Ack ADU IssueCode 704 billingDetailsUserIdRejected
Adus: usageListEntry appliedEuCO2EmissionClass appliedVehicleWeight	The declared vehicle class in billing details has been rejected.	Ack ADU IssueCode 709 billingDetailsDeclaredVehicleClassRejected
Adus: billingDetailsAdu aduIdentifier + actionCode	Validation of unique Billing Detail in combination with actionCode	Ack ADU issueCode 10700: Duplicate billing details aduIdentifer rejected.
tollDeclarationEventId: chargeReportCounter + LPN + obeId	Validate that the chargeReportCounter is known for the obeId + LPN	Ack ADU issueCode 10701: chargeReportCounter is invalid for this user
tollEventTime chargeObjectDesignation	Validate double charge of segment at the same time, The entrance charge object in billing details has been rejected.	Ack ADU issueCode 712 charge object rejected
adus: actionCode	The add and revoke action code are the only ones supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

6.2.6 Key payload fields when revoking Billing Details

This section highlights some of the data fields relevant for the revoke flow. It is not a description on how to use the entire messages.

Key fields for Billing Detail and revoked Billing Detail.

- "dateOfService" Specifies the creation date of the Billing Detail and will be updated when a revoked Billing detail or a new Billing Detail is issued.
- "beginOfPeriod" and "endOfPeriod" Specifies the driving date from the Toll declaration included in the Billing detail. This will be the same in the revoked BD and if a new / additional Billing detail is issued.

- "aduIdentifier" / "billingDetailsNum" Unique identifier for this Billing detail, the revoked Billing detail will have same aduidentifier as the original ´Billing Detail, if a new Billing Detail is issued it will have a new aduidentifier as it is a new message.
- "actionCode" determine the action to perform on the massages and action code values "0" (Send) and "1" (revoke) are used. Where "send" is debited and "revoked" is credited.

Original BD

```
"aduIdentifier": 100000000000036191,
"billingDetailsNum": 100000000000036191,
"dateOfService": "20240525003003Z",
"period": {
  "beginOfPeriod": "20240506220000Z",
  "endOfPeriod": "20240507220000Z"
 "actionCode": 0,
```

Revoked BD (refence to the original aduIdentifier)

```
"aduIdentifier": 100000000000036191,
"billingDetailsNum": 100000000000036191,
"dateOfService": "20240528003003Z",
"period": {
  "beginOfPeriod": "20240506220000Z",
  "endOfPeriod": "20240507220000Z"
 "actionCode": 1,
```

New BD (no reference to the original, but same driving date)

```
"aduIdentifier": 100000000000048093,
"billingDetailsNum": 100000000000048093,
"dateOfService": "20240528122805Z",
"period": {
  "beginOfPeriod": "20240506220000Z",
  "endOfPeriod": "20240507220000Z"
"actionCode": 0,
```

6.3 EETS Payment Claim interface

6.3.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS Payment Claim – vn*", where "*vn*" is the version number currently in use.

6.3.2 Purpose of the Interface

The interface will be used by the TC to issue Payment Claim to a TSP.

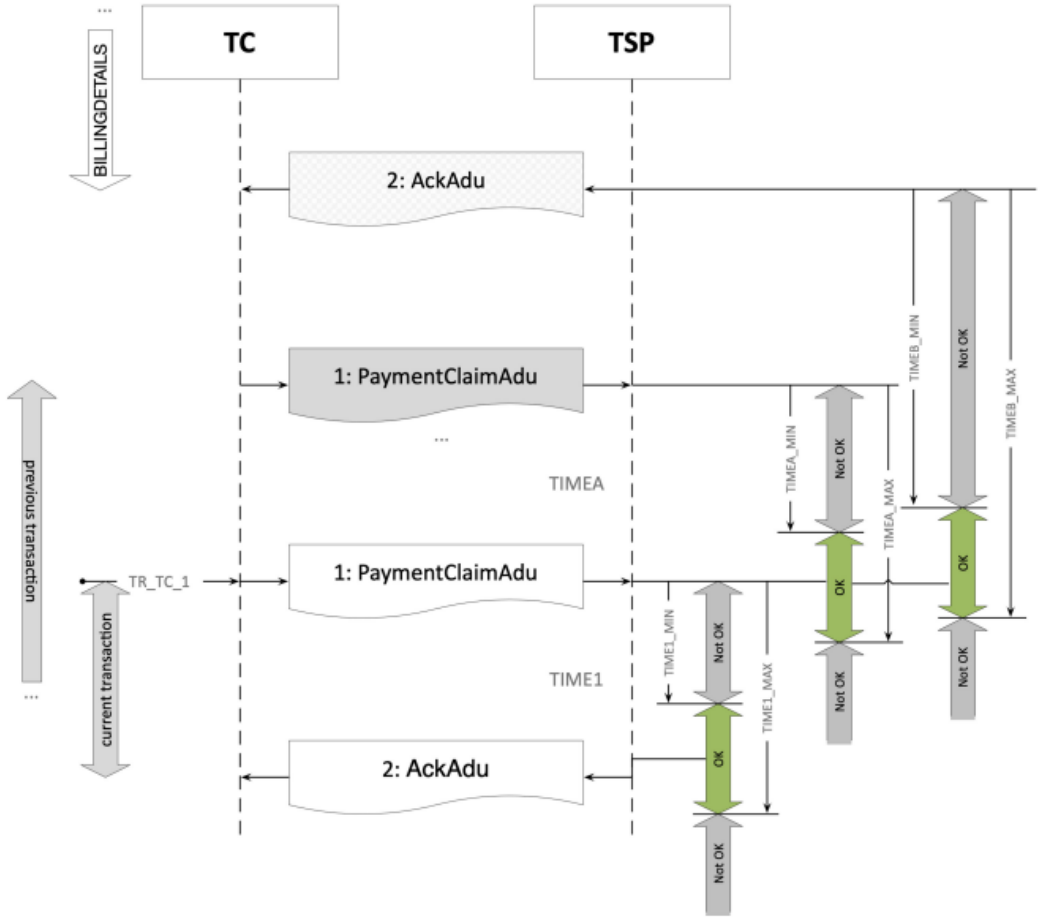
6.3.3 Communications Processes

The Payment Claim will be transferred using a REST Web-Service interface as defined in section 0, and will be Acknowledged (see 6.6).

The *AckAdu* is used to acknowledge the *PaymentClaimADU.*

The following additional specifications apply to this interface – Figure 16 from prEN16986:2023 is reproduced below for reference:

Figure 30. Copy of Figure 16 from EN 16986



1) Payment Claims will be pushed to the TSP at contractually agreed intervals. This is currently expected to be once per calendar month.
2) Payment claims will be synchronously acknowledged, i.e., a Payment Claim must be acknowledged before another can be sent, though timeouts will apply.
3) It is possible that Payment Claims will be too large to be transferred in a single message, in which case they must be sent as multiple chunks (see description of the chunking process in "Message Formats" below).
4) If no chunking is required, each Payment Claim list message will be acknowledged with an *Acknowledge TC* Adu.
5) In the case where chunking is required, assume a message has "n" chunks:
   a) Chunks 1 to (n-1) are acknowledged at the API level only (i.e., no *Acknowledge TC* Adu is sent)
   b) Once all n chunks have been received, the entire list will be acknowledged with an *Acknowledge TC Adu*. Each chunk will be explicitly acknowledged within the *Acknowledge TC Adu* with a separate *AckTcAdu* data field.
6) TIMEA_MIN and TIMEA_MAX (prEN16986:2023 Table 71 and Figure 16) are undefined
7) TIMEB_MIN (prEN16986:2023 Table 71) is defined as 0
8) TIMEB_MAX (prEN16986:2023 Table 71) is undefined

9) TIME1_MIN (prEN16986:2023 Table 71) is defined as 0
10) TIME1_MAX (prEN16986:2023 Table 71) is defined as 24 hours

6.3.4 Message Formats

The Payment Claim is transmitted in an *InfoExchange* message with a *PaymentClaimADU* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986. The *referenceDetailsList* data element will be used, containing the *billingDetailsList* choice.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

The *referenceDetailsList* contains a list of all the Billing Details for which payment is being claimed. As this may be a very large list (one entry per customer per day, so for a large TSP this could amount to millions of entries), the interface includes a mechanism for dividing the message into multiple smaller parts or "chunks". Each chunk will contain a maximum of 10,000 Billing Details entries. Four additional data elements are included to supporting the chunking:

- *totalPaymentClaimReferenceListEntries* – Total number of entries for this Payment Claim
- *paymentClaimReferenceListEntriesFromRange* – The value either "1" (first payload) or next number after previous *paymentClaimReferenceListEntriesToRange* (2nd and subsequent payloads)
- *paymentClaimReferenceListEntriesToRange* – *paymentClaimReferenceListEntriesFromRange* + *countOfPaymentClaimReferenceListEntries* – 1
- *countOfPaymentClaimReferenceListEntries* – Count of referenceDetailList entries in this payload

In cases where the chunking mechanism is required, each *apduIdentifier* must be unique as required by the standards. However, the different chunks of the same payment claim must have the same *aduIdentifier*.

Example 1: Payment Claim references 1 Billing Details (no chunking required):

Table 12. Payment Claim references 1 Billing Details

Data Element	Value
apduIdentifier	12345
aduIdentifier	6789
totalPaymentClaimReferenceListEntries	1
paymentClaimReferenceListEntriesFromRange	1
paymentClaimReferenceListEntriesToRange	1
countOfPaymentClaimReferenceListEntries	1

Example 2: Payment Claim references 1000 billing details (no chunking required):

Table 13. Payment Claim references 1000 billing details

Data Element	Value

apduIdentifier	12345
aduIdentifier	6789
totalPaymentClaimReferenceListEntries	1000
paymentClaimReferenceListEntriesFromRange	1
paymentClaimReferenceListEntriesToRange	1000
countOfPaymentClaimReferenceListEntries	1000

Example 3: Payment Claim references 10,900 billing details, so two chunks required:

First message containing chunk 1

Table 14. Payment Claim references 10,900 billing details – First message containing chunk 1

Data Element	Value
apduIdentifier	12345
aduIdentifier	6789
totalPaymentClaimReferenceListEntries	10900
paymentClaimReferenceListEntriesFromRange	1
paymentClaimReferenceListEntriesToRange	10000
countOfPaymentClaimReferenceListEntries	10000

Second message containing chunk 2

Table 15. Payment Claim references 10,900 billing details - Second message containing chunk 2

Data Element	Value
apduIdentifier	23456
aduIdentifier	6789
totalPaymentClaimReferenceListEntries	10900
paymentClaimReferenceListEntriesFromRange	10001
paymentClaimReferenceListEntriesToRange	10900
countOfPaymentClaimReferenceListEntries	900

6.3.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 16. Error codes - Payment Claim interface

Data	Validation	Acknowledgement
Message as a whole	The related payment claim has been rejected by the TSP.	Ack ADU IssueCode 800 claimRejectedByTSP
adus: startDateTime	The start date and time in payment claim has been rejected.	Ack ADU IssueCode 803 paymentStartDateTimeRejected
adus: endDateTime	The end date and time in payment claim has been rejected.	Ack ADU IssueCode 804 paymentEndDateTimeRejected
adus: paymentFeeAmount	The amount in payment claim has been rejected.	Ack ADU IssueCode 806 paymentAmountRejected
PaymentClaimADU: paymentClaimStatus	The status in payment claim has been rejected.	Ack ADU IssueCode 807 paymentStatusRejected
BillingDetailsInfo: billingDetailsNum	The related APDU ID in payment claim has been rejected.	Ack ADU IssueCode 809 paymentRelatedApduIdRejected
adus: actionCode	Only the add action code is supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

6.4 EETS Report Abnormal OBE interface

Note: The implementation of this interface has been suspended until further notice. TSPs will not be required to implement this interface during the initial roll-out of the KmToll system. Sund & Bælt reserve the right to require that TSPs support this interface at some future date.

6.4.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS Report Abnormal OBE – vn*", where "*vn*" is the version number currently in use.

6.4.2 Purpose of the Interface

The interface allows the TC to report that the OBE in one of the TSP's EETS User's vehicle is behaving abnormally.

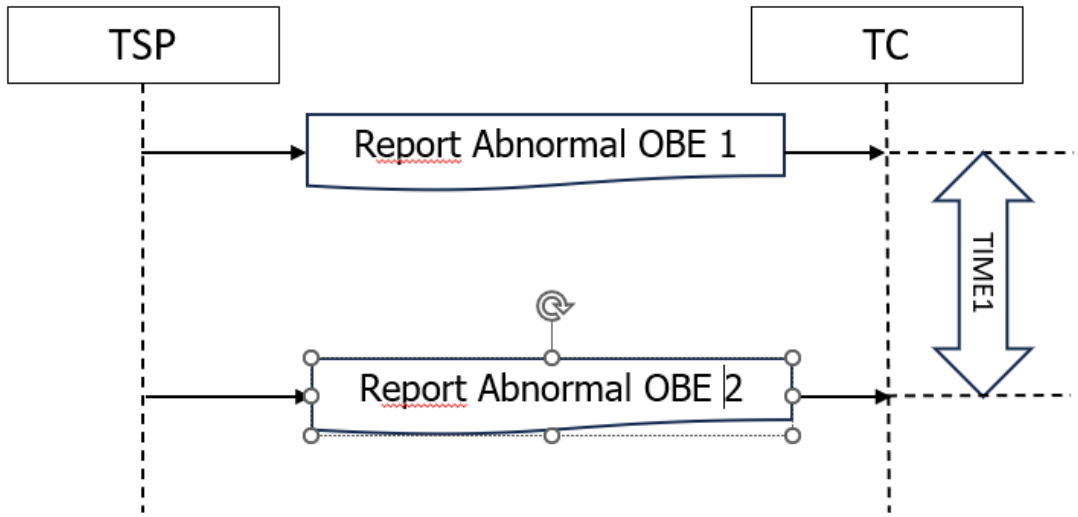
6.4.3 Communications Processes

The Report Abnormal OBE will be transferred using a REST Web-Service interface as defined in section 0, and will be Acknowledged (see 6.6).

The *ReportAbnormalObeAdu* will be acknowledged at the API level only, i.e., the *AckAdu* will not be used to acknowledge the *ReportAbnormalObeAdu.*

The following additional specifications apply to this interface:

Figure 31. Report Abnormal OBE message flows



1. Each *reportAbnormalObe* message need be acknowledged at the API (HTTPS) level only.
2. Multiple *reportAbnormalObe* messages can be sent in parallel.
3. TIME1_MIN is defined as 0

6.4.4 Message Formats

The Report Abnormal OBE data is transmitted in an *InfoExchange* message with a *ReportAbnormalObeAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

6.4.5 Error Handling

API errors will be handled as described in section 5.5.

6.5 EETS CCC Data Request interface

6.5.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS* CCC *Data Request – vn*", where "*vn*" is the version number currently in use.

6.5.2 Purpose of the Interface

The interface will be used by the TC to request Compliance Check data from the TSP. The TSP will be expected to reply with a message containing Compliance Check data to the CCC_Data_Response interface described below. The Compliance Check data will be similar to the CCC data transmitted from the RSE to the OBE in the conventional DSRC-based CCC transaction.

Note that only TSPs which intend to offer an OBE Type 2 to the EETS User will be required to implement this interface.

6.5.3 Communications Processes

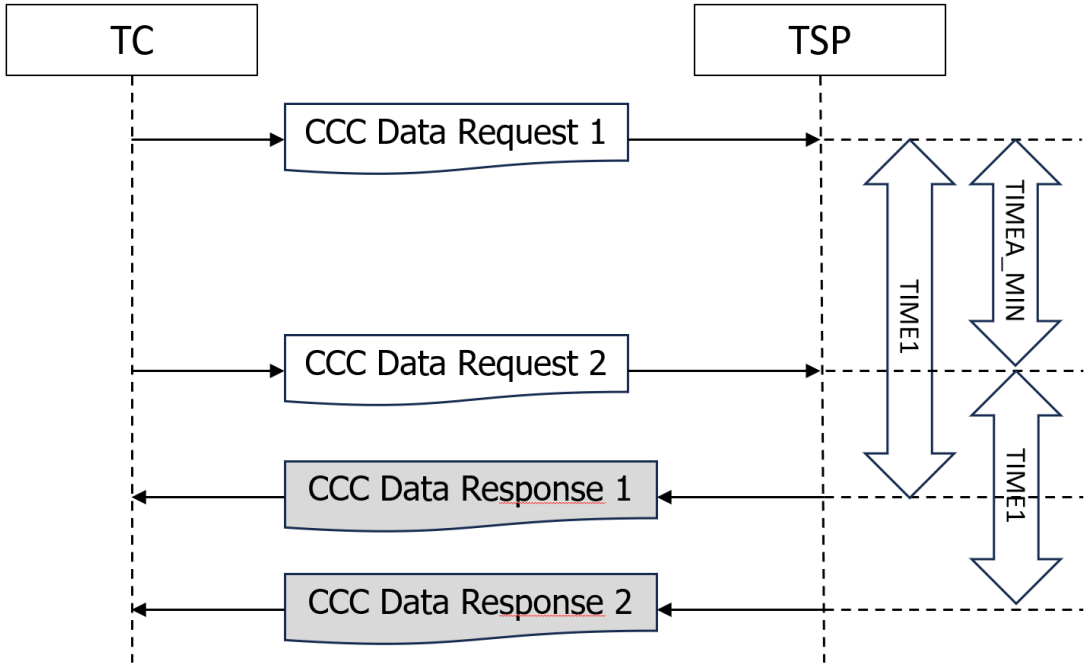
The CCC_Data_Request will be transferred using a REST Web-Service interface as defined in section 0. It will not be explicitly acknowledged at the ADU level as the response from the TSP will constitute an implicit Acknowledgement.

Note that the *CCCDataRequestADU* is not defined in EN 16986, but has been specifically created for this application. It is based on the *RequestAdu* structure defined in ISO 12855:2022

The *CCCDataResponseADU* is used to implicitly acknowledge the *CCCDataRequestADU.*

The communications sequence is shown below.

Figure 32. CCC Data Request flow



The TSP is required to respond to every CCC Data Request with a CCC Data Response.

TIMEA_MIN, the minimum time between successive CCC Data Requests is 0. Values for TIME1 are given in the description of the CCC Data Response interface, see section 6.14.

6.5.4 Message Formats

The CCC Data Request is transmitted in an *InfoExchange* message with a *cccDataRequestAdu* as defined in this document and this must in principle satisfy the restrictions described in the Section-Autonomous profile (with TC dominance) of specification EN 16986.

The following fields are defined for the *cccDataRequestAdu*:

Table 17. cccDataRequestAdu

Field	Comments
requestedADUType	Must be set to **128** to indicate that the request is for a CCC status.

userId	Identifies a user based on License Plate Number (LPN) which is made of • *countryCode* • *alphabetIndicator* • *licencePlateNumber* Further details of these fields can be found in the API specification
listOfParametersRequested	Holds the *userParameterRequest* field which must be set to **128** to indicate that it is the OBE's status that is to be reported.
userDetailsRequestReason	Must be set to **128** to indicate that the request is for CCC status.
userInfoValidityPeriod	Identifies the period for which the CCC information is requested • *beginOfPeriod*: VPR time / time of passage minus 20 seconds • *endOfPeriod*: VPR time / time of passage

Formal message and data formats can be downloaded from the API Management Developer Portal.

6.5.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 18. Error codes - CCC Data Request interface

Data	Validation	Acknowledgement
All data fields	The format of the message received was not understood by the recipient.	HTTP 400 Bad request
adus: actionCode	The sent action code is not supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

6.5.6 Sample Request

Sample 8 - cccDataRequestAdu for a VPR registered on 15.08.2024 08:23:17


```
"InfoExchange": {
   "InfoExchangeContent": {
      "apci": {
         "aidIdentifier": 17,
         "apduOriginator": {
            "countryCode": "1001011110",
            "providerIdentifier": 5
         },
         "informationSenderID": {
            "countryCode": "1001011110",
            "providerIdentifier": 5
```

```
         },
         "informationrecipientID": {
            "countryCode": "0011111100",
            "providerIdentifier": 116
         },
         "apduIdentifier": 151034950002001,
         "apduDate": "20240815082317Z"
      },
      "adus": {
         "cccDataRequestAdus": {
            "cccDataRequestAdu": {
               "requestedADUType": 128,
               "userId": {
                  "LPN": {
                     "countryCode": "1001011110",
                     "alphabetIndicator": "latinAlphabetNo1",
                     "licencePlateNumber": "41423331323332000000000000000"
                  }
               },
               "listOfParametersRequested": {
                  "UserParameterRequest": 128
               },
               "userDetailsRequestReason": 128,
               "userInfoValidityPeriod": {
                  "beginOfPeriod": "20240815082257Z",
                  "endOfPeriod": "20240815082317Z "
               }
            }
         }
      }
   }
}
```


6.6 EETS Acknowledgments TC Interface

6.6.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS Acknowledgments TC – vn*", where "*vn*" is the version number currently in use.

6.6.2 Purpose of the Interface

The EETS Acknowledgments TC interface is implemented by the TC and is used by the TSP to acknowledge the correct receipt of message by the TSP. It is never used in isolation and should be considered as a part of the messaging protocol.

Acknowledgements can be synchronous or asynchronous, depending on the message type being acknowledged.

In this version of the Specification, the EETS Acknowledgments TSP interface may only be used as part of the following transactions:

- BILLINGDETAILS_TC
- PAYMENTCLAIM

In future versions, it may be used in other transactions.

6.6.3 Communications Processes

The Acknowledgement will be transferred using a REST Web-Service interface as defined in section 0.

Acknowledgments may be synchronous or asynchronous, depending on the transaction. See the descriptions for Billing Details, Payment Claim, and Report Abnormal OBE interfaces for more details.

6.6.4 Message Formats

The EETS Acknowledgments TSP data is transmitted in an *InfoExchange* message with an *AckAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

The AduReasonCodes defined in Table 9 of ISO 12855:2022 must be supported.

The AduReasonCodes reported in the *issueCode* field which are defined in Table 11 of ISO 12855:2022 should be supported by the TSP. While all values in the range 0 – 65535 are allowed, the TSP can use the following specific values to report issues to the TC:

- Values 0 – 1 (generic ADU errors)
- Values 500 – 502 (abnormal OBE report errors)
- Values 700 – 716 (billing details related errors)
- Values 800 – 809 (payment claim related errors)
- Value 3000 (semantic error)
- Value 3010 (action code not supported error)

The following user defined values may also be used by the TSP to report issues - this list may be further expanded in future versions of this specification:

- Value 10700: Duplicate billing details aduIdentifier rejected. The AduIdentifier for the billing details has been previously used, and hence the billing details is rejected.
- Value 10701: chargeReportCounter is invalid for this user. A value used in the chargeReportCounter field has no corresponding value in a toll declaration, and hence the billing details is rejected.

Ranges marked as "reserved" will only be used when defined in future versions of the standards, or their use has been defined in bilateral agreements between the TC and the TSP.

Note further that it is not foreseen that the value 4000 (acceptedWithWarning) will be used, but it may be used in a future version of this interface.

6.6.5 Error Handling

API errors will be handled as described in section 5.5.

6.7 EETS Toll Declaration interface

6.7.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS Toll Declaration – vn*", where "*vn*" is the version number currently in use.

6.7.2 Purpose of the Interface

The EETS Toll Declaration interface is to be used by the TSP for the transmission of Toll Declarations generated by TSP on-board equipment from the TSP's central system to the TC.

The EETS Toll Declaration contains the GNSS position data generated by the On-Board Equipment (OBE) of the TSP, which is required by the TC for the calculation of tolls. The report also contains toll-relevant data about the vehicle, for example maximum permissible train `6.2, vehicle class, emissions class etc.

Note A Toll Declaration may only contain data for a single OBE

Note 2 Examine the restrictions in the next section carefully. It is important to understand that acknowledgement restrictions for Toll Declarations from the same OBE are different to acknowledgement restrictions for Toll Declarations from different OBE.

6.7.3 Communications Processes

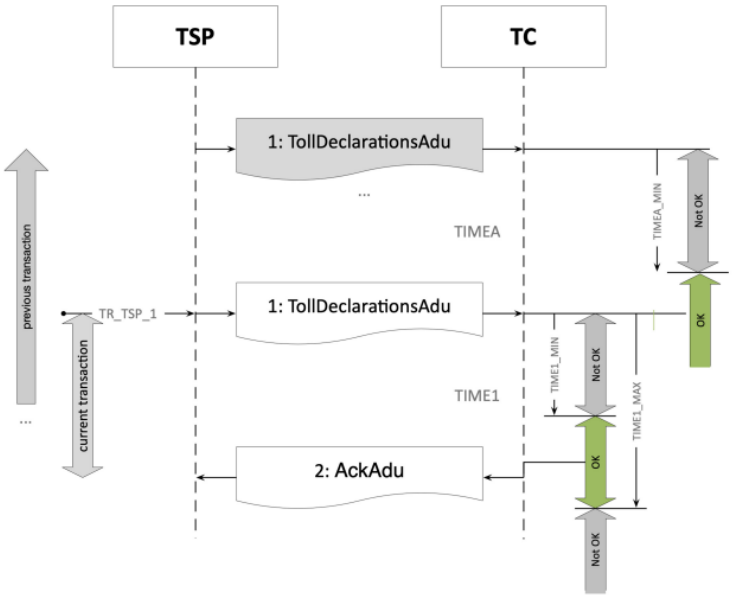
The Toll Declaration will be transferred using a REST Web-Service interface as defined in section 0, and will be Acknowledged (see 6.6).

Each InfoExchange message may only contain exactly one *TollDeclarationADU* and thus exactly one *ChargeReport.*

The *AckAdu* (see 6.6) is used to acknowledge the *TollDeclarationADU.*

The following additional specifications apply to this interface – Figure 21 from prEN16986:2023 is reproduced below for reference:

Figure 33. Copy of Figure 21 from EN 16986



1. Multiple Toll Declarations from different OBE can be started in parallel.
2. Toll Declarations from the same OBE MUST be sent in time sequence, as determined by the timestamps of the raw GNSS data points. Out of sequence Toll Declarations will not be processed.
3. Each Toll Declaration from the same OBE must be synchronously acknowledged, i.e., once a Toll Declaration for a specific OBE has been sent, a subsequent Toll Declaration for that OBE can only be sent after the initial one has been acknowledged.
4. If, after an agreed timeout period (see TIME1_MAX below) no acknowledgement is received, a service case should be raised. Only once this has been resolved should toll declaration sending resume.
5. TIMEA_MIN (prEN16986:2023 Table 132 and Figure 21) is defined as zero.
6. TIME1_MIN (prEN16986:2023 Table 132) is defined as zero.
7. TIME1_MAX (prEN16986:2023 Table 132) is defined as 5 minutes.

6.7.4 Message Formats

The Road Usage data is transmitted in an *InfoExchange* message with a *TollDeclarationAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

A toll declaration may only contain one *tollDeclarationAdu*, which in turn may only contain one *usageStatement*. Hence a toll declaration may only contain tolling information from one OBE.

Limits on measured position

The measured position is given in latitude and longitude, as described in the API Management Developer Portal. It should be noted that, further to the latitude and longitude values defined in ISO/TS 17573-3. additional limits have been placed on the range of latitude and longitude which may be reported. These limits are used to ensure that the measured position falls within a "box" defined by the geographic limits of the land area of Denmark, including the island of Bornholm, but excluding the Faroe Islands and Greenland.

The reported longitude must be in the range 7.8 degrees to 15.5 degrees, corresponding to the range 7800000 micro-degrees to 15500000 micro-degrees. Toll declarations containing locations outside these limits will be rejected.

The reported latitude must be in the range 54.4 degrees to 58.0 degrees, corresponding to the range 54400000 micro-degrees to 58000000 micro-degrees. Toll declarations containing locations outside these limits will be rejected.

Identification of OBE type

The KM-Toll system supports both OBE using dedicated hardware and OBE implemented as software on a nomadic platform (e.g., a smartphone), called Type 1 and Type 2 respectively. In order to identify the OBE type, a new data field called "*obeType*" is introduced as part of the "*obeId*" field. *obeType* is defined below:

Table 19. obeType

Data Element	Data type	Description
obeType	Integer	Identified the OBE type in use. It can take the values: "1" – OBE is of type 1 "2" – OBE is of type 2 Values between "3" and "127" are reserved for future use.

Additional GNSS data fields.

In addition to the data fields defined in ISO 12855:2022 and EN 16986, a new data field called "*additionalGnssData*" is introduced as part of the "*measuredRawData*" field. This will contain the following data elements:

Table 20. additionalGnssData

Data Elements		Data type	Description
additionalGnssData			Provides additional GNSS information as measured by the GNSS receiver.
	Speed	Integer	Measured speed on the GNSS receiver, in cm/s. Minimum speed is "0", maximum speed is 9999 cm/s. If this data is not available, the value "-1" should be entered. In the unlikely event that the measured speed is greater than 9999 cm/s, then the speed value must be recorded as "9999"
	heading	Integer	Measured heading of the GNSS receiver, in tenths of a degree. Allowed values are "0" to "3599", where "0" is due north and #3599" is a heading of 359.9 degrees. If this data is not available, the value "-1" should be entered.
	Integrity	string	Signal integrity as reported by the GNSS receiver. This is a single character, reported as a string of length 1, with the following allowed values: "A" – Autonomous mode "D" – Differential mode "E" – Dead reckoning for locates without subsequent editing "M" – Manual mode for locates supplemented subsequently "S" – Simulated mode with subsequently edited location information "N" – Non-valid location. This entry is not currently used in the toll calculation, but may be used in future versions of the system.
	noOfSatellites	Integer	Number of satellites used to obtain the geo-location. This must be a value between "0" and "99". "0" satellites should be reported if this data is not available, for example for simulated locations, dead reckoning etc. In the unlikely event that the number of satellites used is greater than 99, the number must be reported as "99". This entry is not currently used in the toll calculation, but may be used in future versions of the system.

	hDOP	Integer	The Horizontal Dilution of Precision, as reported by the GNSS receiver. This must be a value between "0" and "999". This value must be reported in tenths, i.e., if HDOP is 1, it should be reported as the value "10" If HDOP is unavailable, it should be reported as value "-1"
	hAcc	Integer	Horizontal Accuracy of the reported geo-location. This must be a value between "0" and "9999". This is an estimate of the 1-sigma error radius for the geo-location, i.e. this is the radius of the likely uncertainty error for a 68% (1-sigma) error probability. hAcc must be reported in cm. If hAcc is unavailable, it should be reported as value "-1"

The data fields speed, heading, hDOP, hAcc allow the use of a "data not available" indicator. It is not acceptable for this to be a default value; it is only to be used in specific circumstances where the data is temporarily unavailable for technical reasons.

Vehicle environmental characteristics

To comply with EU directives, it has been necessary to add a new attribute to the field *environmentalCharacteristics*, describing the vehicle's CO2 emissions class. This attribute is called *euCO2EmissionClass* and is an integer with a value between 1 and 5 inclusive, as shown below.

Table 21. euCO2EmissionClass

Attribute	Data type	Description
euCO2EmissionClass	Integer	EU emissions class according to EU directives Minimum value 1 Maximum vale 5

Note that three different attributes are defined within the *VehicleWeightLimits* field. All three attributes should be populated, using the rules as described in the API Management Developer Portal. Toll will be calculated from the F1 value as reported in the vehicleTechnicalMaxLadenWeight field.

6.7.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 22. Error codes Toll Declaration interface

Data	Validation	NACK/ACK
InformationoriginatorID	Validate against approved provider table and rise an alert if not found.	HTTP 400 Bad request or 401 Unauthorized
listOfRawUsageData: measuredPosition	Validate that longitude and latitude (approximate) are in Denmark	HTTP 400 Bad request
VehicleDescription: euCO2EmissionClass	Within 1 – 5	HTTP 400 Bad request or 401 Unauthorized

VehicleWeightLimits: vehicleTechnicalMaxLadenWeight	>=12000kg (value >=1200)	Ack ADU issueCode 604 tollDeclarationInvaldidChargeObjectRejected
rawDataList: TimewhenMeasured	Out of Sequence	Ack ADU issueCode 10600: Toll declaration ADU contains out-of-sequence GNSS data
rawDataList: TimewhenMeasured	Older than suitable geomodel	Ack ADU issueCode 10601: Geo-location data is rejected as it is too old
Adus: tollDeclarationId: issuerId DeclarationId	A duplicate of the tollDeclarationId has been identified.	Ack ADU IssueCode 600: tollDeclartionDuplicateTollDeclarationIdRejected
Adus: actionCode	Only the add action code is supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

6.8 EETS Exception Lists interface

6.8.1 Identification of the Interfaces

These interfaces are identified on the relevant portal by the names "*EETS Incremental Exception List - vn*", where "*vn*" is the version number of the interface currently in use. The KmToll will only support incremental exception lists, containing a maximum of one entry.

6.8.2 Purpose of the Interface

The interface is used by the TSP to transfer exception lists to the TC. Exception lists can be of two types: Incremental Whitelist and Incremental Blacklist. Only a single white- or blacklist entry can be transferred per transaction. Longer lists are simply transferred as a sequence of incremental updates.

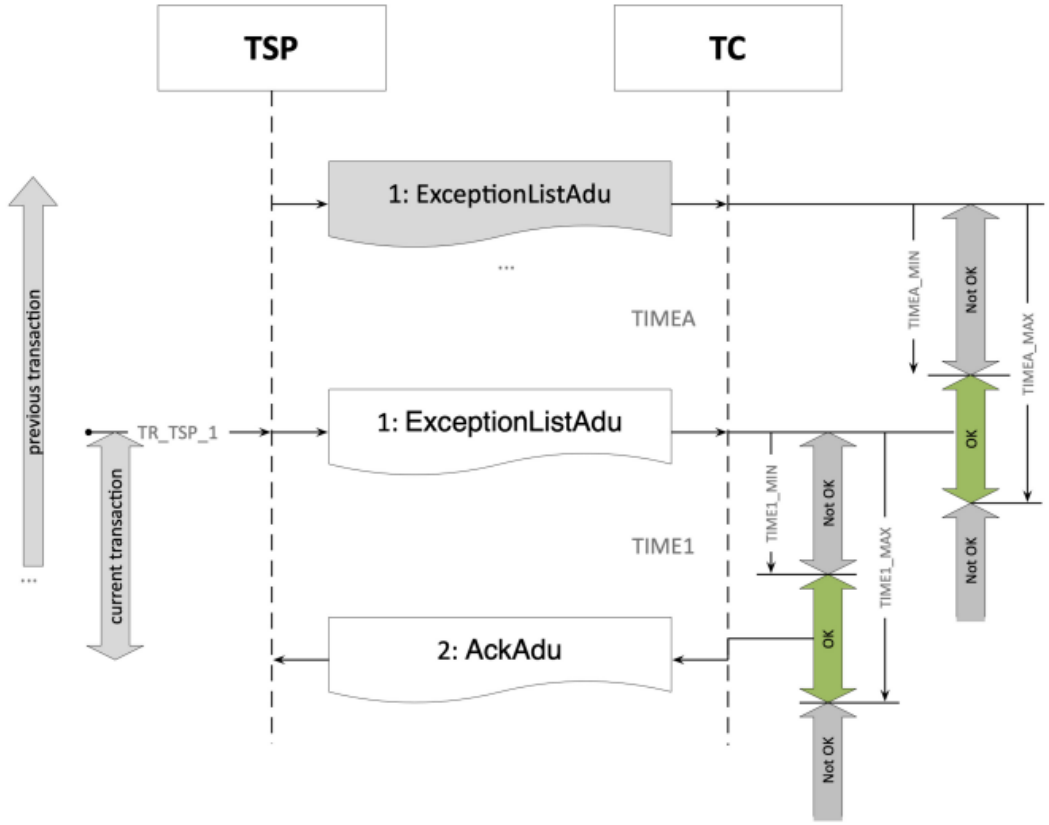
6.8.3 Communications Processes

The Exception Lists will be transferred using a REST Web-Service interface as defined in section 0.

The *AckAdu* is used to acknowledge the EETS Exception Lists, but with special considerations for full lists – see the Message Formats section below.

The following additional specifications apply to this interface – Figure 9 from prEN16986:2023 is reproduced below for reference:

Figure 34. Copy of Figure 9 from EN 16986



1) Incremental Exception Lists may be transferred as required.
2) Each Exception List message will be acknowledged with an *Acknowledge TSP Adu*.
3) TIMEA_MIN (prEN16986:2023 Table 35) is defined as 0 for incremental exception lists
4) TIMEA_MAX (prEN16986:2023 Table 35) is undefined for incremental exception lists
5) TIME1_MIN (prEN16986:2023 Table 35) is defined as 0 for incremental exception lists
6) TIME1_MAX (prEN16986:2023 Table 35) is defined as 10 minutes for incremental exception lists

6.8.4 Message Formats

The Exception Lists are transmitted in an *InfoExchange* message with a *ExceptionListAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

Identification of OBE type

The KmToll Scheme supports both OBE using dedicated hardware and OBE implemented as software on a nomadic platform (e.g., a smartphone), called Type 1 OBE and Type 2 OBE respectively. To identify the OBE type, a new data field called "*obeType*" is introduced as part of the "*obeId*" field. *obeType* is defined below:

Table 23. obeType

Data Element	Data type	Description
obeType	Integer	Identifies the OBE type in use. It can take the values: "1" – OBE is of type 1 "2" – OBE is of type 2 Values between "3" and "127" are reserved for future use.

6.8.5 Limitations

Some data fields in the ADU have restrictions placed on the values allowed. These are listed below.

Table 24. Data field in ADU with restrictions

Data Field	Allowed Values
exceptionListType	11 (incremental blacklist) 12 (incremental whitelist)
reasonCode	0 ("notToBeDisclosed", use only for blacklist) 8 ("whiteListedUser, use for only whitelist)
statusType	0 or 1 (used for blacklist) 3 (used for whitelist)
actionCode	0 (add), 1 (revoke) and 3 (adjust)

Coding of Acknowledgements (see also section 6.1 EETS Acknowledgments TSP Interface)

Limitations to the Acknowledge APDU reason code to be used to accept or reject an APDU.

Table 25. Values used to accept or reject Acknowledge APDU

ApduReasonCodes values	Semantics	ApduAckCode values
apduOk	The APDU and ADU was accepted.	2
apduNotOk	The APDU and ADU was rejected.	3
originatorRejected	The APDU was rejected because the Apdu Originator is not known, or no valid contract exists.	7

ADU Reason Codes to be supported:

Limitations to the Acknowledge ADU issue code to be used to reject an entry.

Table 26. Values used to reject an Acknowledge ADU

AduReasonCode values	Validation	IsssueCode values

exception-ListsSta-tusTypeRejected	The type of status in exception list has been rejected.	403
exception-ListsReasonTy-peRejected	The reason code in exception list has been rejected.	404

6.8.6 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 27. Error codes - Exception Lists interface

Data	Validation	Acknowledgement
adus: exceptionListType	The type of exception list has been rejected.	HTTP 400 Bad request
adus: statusType	The type of status in exception list has been rejected.	Ack ADU IsssueCode 403 exceptionListsStatusTypeRejected
adus: reasonCode	The reason code in exception list has been rejected.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active	HTTP 400 Bad request or 401 Unauthorized
adus: actionCode	Only the add, revoke and adjust action code is supported.	HTTP 400 Bad request

6.9 EETS Payment Announcement interface

6.9.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS* Payment Announcement – *vn*", where "*vn*" is the version number currently in use.

6.9.2 Purpose of the Interface

The interface is used by the TSP to inform the TC that a payment has been made to the TC. This payment will normally be made in response to a Payment Claim (see 6.3).

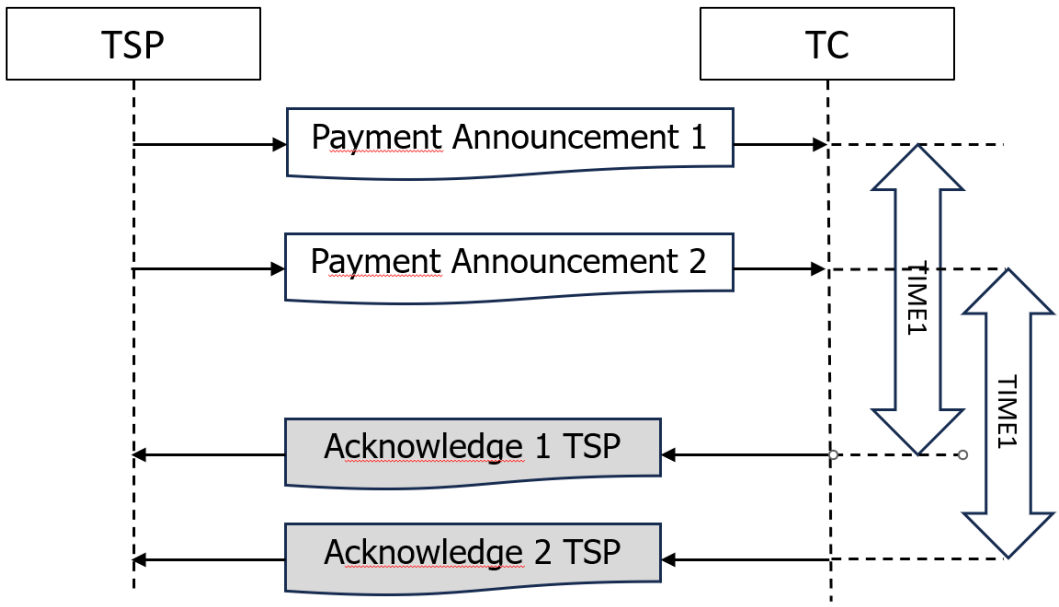
6.9.3 Communications Processes

The Payment Announcement will be transferred using a REST Web-Service interface as defined in section 0, and will be Acknowledged (see 6.6).

The *AckAdu* is used to acknowledge the *PaymentAnnouncementAdu.*

The following additional specifications apply:

Figure 35. Payment Announcement flows



1. The TSP shall issue a Payment Announcement within an agreed period after making a payment to the TC. This time is contractually defined.
2. The TC will acknowledge the Payment Announcement within TIME1. This is currently defined as 24 hours.
3. The Payment Announcement will be asynchronously acknowledged, i.e., multiple Payment Announcements may be sent in parallel.

6.9.4 Message Formats

The Payment Announcement data is transmitted in an *InfoExchange* message with a *PaymentAnnouncementAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

Data fields removed from or added to those defined in EN 16986:

1. As ISO 12855:2022 has no pre-defined AduReasonCodes to indicate rejection of the payment announcement, a user defined code of value 10900 has been defined for this purpose. If a future version of ISO 12855 defines specific codes within the pre-defined range of 900 – 999, these may be used in a future version of this specification.

6.9.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 28. Error codes - Payment Announcement interface

Data	Validation	Acknowledgement
referenceDetailsList: referencedAduIdentifier	Payment Announcement ADU not accepted. The data in the Payment	IssueCode 10900 IssueText: "Payment-Announcement referenceAduIdentifier [referenceA-duIdentifier] is unknown at Toll charger"
paymentFeeAmount	Announced payment amount not matching Payment claim amount	IsssueCode 10900 IssueText: PaymentFeeAmount value [paymentFeeAmount] is not identical to referenceAduIdentifier [referenceA-duIdentifier] paymentFeeAmount value [FU.ClaimProposals].[ClaimPro-posalAmount]
paymentReference	Announced payment ref-erence not matching pay-ment claim	IsssueCode 10900 IssueText: PaymentReference value [paymentReference] is not identical to referenceAduIdentifier [referenceA-duIdentifier] paymentReference value [FU.ClaimProposals].[aduIdentifier]
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized
adus: actionCode	Only the add action code is supported.	HTTP 400 Bad request

6.10 EETS Contract Issuer List interface

6.10.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS* Contract Issuer List – *vn*", where "*vn*" is the version number currently in use.

6.10.2 Purpose of the Interface

The interface is used by a TSP to provide a TC with contractual information about their issued OBE.

The S&B Kilometer Tolling scheme uses the EETS Contract Issuer List API to keep track of the TSPs OBE types and associated trust objects in order for the RSE to communicate with the OBEs using DSRC.

The Contract Issuer List is used for enforcement purposes only and is not validated when charging the applicable toll on specific OBEs.

6.10.3 Communications Processes

The Contract Issuer List will be transferred using a REST Web-Service interface as defined in section 0, and will only be acknowledged at the API level.

Registration of Contract Issuer List

When a new TSP enters the S&B Kilometer Tolling toll domain all supported OBE types must be registered in the S&B Kilometer Tolling scheme.

To facilitate this the TSP must use the EETS Contract Issuer List API to send the relevant information according to the *contractIssuerListAdu* data structure defined in ISO 12855 and in EN 16986 and in this document.

The following samples reflect registration of the contract issuer list:

- Sample 9 – Register contractIssuerList containing 1 contractIssuerListEntry (OBU Type 1)
- Sample 10 – Register contractIssuerList containing 2 contractIssuerListEntry (OBU Type 1 and 2)

Update of Contract Issuer List

If a TSP gets a new type of OBE or a specific type of OBE should no longer be considered in the Toll Domain, the TSP must use the EETS Contract Issuer List API to update the list in the S&B Kilometer Tolling scheme.

To update the list, a new full list must be sent which will replace the current list at the time defined in *validFrom* field of the *versionAndValidity* field of the *contractIssuerListAdu*.

The following samples reflect update of the contract issuer list:

- Sample 11 – Update to contractIssuerList created in Sample 9 (Adding a new OBU Type 1 to the list)
- Sample 12 – Update to contractIssuerList created in Sample 10 (Removal of OBU Type 2)

6.10.4 Message Formats

The Contract Issuer List data is transmitted in an *InfoExchange* message with a *ContractIssuerListAdu* in accordance with standard ISO/DIS 12855:2022 and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

Formal message and data formats can be downloaded from the API Management Developer Portal. This will also contain any restrictions or changes to the formal message format definitions in EN 16986.

Data fields removed from, added to, or changed from those defined in EN 16986:

ContractIssuerList

The *contractIssuerList* contains one *contractIssuerListEntry* for each type of OBE that the TSP supports

EfcContextMark

Each entry in the *contractIssuerList* contains an *efcContextMark* element. The *efcContextMark* describes the contract between the TSP and OBE for the RSE to be able to read the details of the OBE using the DSRC protocol.

The following fields are included in the *efcContextMark*:

Table 29. efcContextMark

Field	Comments
contractProvider	Identifies the TSP that the OBE is registered for.
typeOfContract	Identifies the type of contract for the OBE that has been set by the TSP.

contextVersion	Identifies the specific version of the contract. The value of contextVersion is at the discretion of the TSP and is currently not used by S&B.

EfcContextMarkVersion

Each entry in the list contains an efcContextMarkVersion element.

The *efcContextMarkVersion* defines the compatibility of the format of the *efcContextMark* and is coded as defined in *ISO16986(2023) EfcInfoExchangeSectionAutonomousProfileV2.2.asn*

The *efcContextMarkVersion* is currently not validated in the S&B Kilometer Tolling scheme and can thereby be set to any of the allowed values

Additional ContractIssuerListEntry Fields

Additional fields of the *contractIssuerListEntry* are defined as follows:

Table 30. contractIssuerListEntry

Field	Comments
equipmentClass	Identifies the equipment class of the OBE as defined in ISO 14906
manufacturerId	Identifies the manufacturer of the OBE as defined in ISO 17573-3
uniquePartOfPan	Identifies the leftmost unique digits of PAN used by the TSP
typeOfEfcApplication	Identifies the type of OBE used in the S&B Kilometer Tolling scheme. Must comply to one of the following: • **1** - GNSS Type 1 OBE (including a DSRC element) • **2** - GNSS Type 2 OBE (not including a DSRC element)
securityLevel	Security level to use for the defined EFC application • 0 -- no access credentials • 1 -- security level 1 • 2 -- security level 2 ***Must always** be set to **0**
acCrKeyReference	Number of key in the keyset to use for access credentials ***Not used** by S&B Kilometer Tolling scheme – must be set to **0**
authKeyReference	Number of key in the keyset to use for TSP authenticator ***Not used** by S&B Kilometer Tolling scheme – must be set to **0**

versionAndValidityEntry	Identifies the validity of the *contractIssuerListEntry* ***Note:** This field only covers the validity of the specific entry in the list. The validity of the actual is defined in the versionAndValidity field described in section **Fejl! Henvisningskilde ikke fundet. Fejl! Henvisningskilde ikke fundet.**

VersionAndValidity

The *versionAndValidty* element identifies the validatity of the full *contractIssuerListAdu*.

The *versionAndValidty* element must be used by the enforcement system to identify which contract issue list is the current active list for a given TSP.

***Note**: This field covers the validity of the full Contract Issuer List. Each entry in the list also has a validity which is defined in the *versionAndValidityEntry* field described in **Additional ContractIssuerListEntry Fields** section above

6.10.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 31. Error codes - Contract Issuer List interface

Data	Validation	Acknowledgement
All data fields	The format of the message received was not understood by the recipient.	HTTP 400 Bad request
adus: actionCode	The add action code is the only one supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

6.10.6 Sample Requests

Sample 9 – Register contractIssuerList containing 1 contractIssuerListEntry (OBU Type 1)

```
"contractIssuerListAdu": {
   "aduIdentifier": 10000002341,
   "contractIssuerList": {
      "contractIssuerListEntry": [
         {
            "efcContextMark": {
               "contractProvider": {
                  "countryCode": "1100011000",
                  "providerIdentifier": 16360
               },
               "typeOfContract": "1A2B",
               "contextVersion": 1
            },
            "efcContextMarkVersion": 1,
```

```
            "equipmentClass": 17,
            "manufacturerId": 63,
            "uniquePartOfPAN": "FF2000",
            "typeOfEfcApplication": 1,
            "securityLevel": 0,
            "acCrKeyReference": 0,
            "authKeyReference": 0,
            "versionAndValidityEntry": {
               "version": "1",
               "validFrom": "20240801000000Z"
            }
         }
      ]
   },
   "versionAndValidity": {
      "version": "1",
      "validFrom": "20240801000000Z"
   },
   "actionCode": 0
}
```

Sample 10 – Register contractIssuerList containing 2 contractIssuerListEntry (OBU Type 1 and 2)

```
"contractIssuerListAdu": {
   "aduIdentifier": 10000002342,
   "contractIssuerList": {
      "contractIssuerListEntry": [
         {
            "efcContextMark": {
               "contractProvider": {
                  "countryCode": "1100011000",
                  "providerIdentifier": 16360
               },
               "typeOfContract": "1A2B",
               "contextVersion": 1
            },
            "efcContextMarkVersion": 1,
            "equipmentClass": 17,
            "manufacturerId": 63,
            "uniquePartOfPAN": "FF2000",
            "typeOfEfcApplication": 1,
            "securityLevel": 0,
            "acCrKeyReference": 0,
            "authKeyReference": 0,
            "versionAndValidityEntry": {
               "version": "1",
               "validFrom": "20240801000000Z"
            }
         },
         {
            "efcContextMark": {
               "contractProvider": {
                  "countryCode": "1100011000",
                  "providerIdentifier": 16360
```

```
               },
               "typeOfContract": "1A3C",
               "contextVersion": 5
            },
            "efcContextMarkVersion": 40,
            "equipmentClass": 25,
            "manufacturerId": 63,
            "uniquePartOfPAN": "FF2000",
            "typeOfEfcApplication": 2,
            "securityLevel": 0,
            "acCrKeyReference": 0,
            "authKeyReference": 0,
            "versionAndValidityEntry": {
               "version": "1",
               "validFrom": "20240801000000Z"
            }
         }
      ]
   },
   "versionAndValidity": {
      "version": "1",
      "validFrom": "20240801000000Z"
   },
   "actionCode": 0
}
```


Sample 11 – Update to contractIssuerList created in Sample 9 (Adding a new OBU Type 1 to the list)


```
"contractIssuerListAdu": {
   "aduIdentifier": 10000002343,
   "contractIssuerList": {
      "contractIssuerListEntry": [
         {
            "efcContextMark": {
               "contractProvider": {
                  "countryCode": "1100011000",
                  "providerIdentifier": 16360
               },
               "typeOfContract": "1A2B",
               "contextVersion": 1
            },
            "efcContextMarkVersion": 1,
            "equipmentClass": 17,
            "manufacturerId": 63,
            "uniquePartOfPAN": "FF2000",
            "typeOfEfcApplication": 1,
            "securityLevel": 0,
            "acCrKeyReference": 0,
            "authKeyReference": 0,
            "versionAndValidityEntry": {
               "version": "1",
               "validFrom": "20240801000000Z"
            }
         },
```

```
         {
            "efcContextMark": {
               "contractProvider": {
                  "countryCode": "1100011000",
                  "providerIdentifier": 16360
               },
               "typeOfContract": "1A2B",
               "contextVersion": 1
            },
            "efcContextMarkVersion": 1,
            "equipmentClass": 25,
            "manufacturerId": 56,
            "uniquePartOfPAN": "FF2000",
            "typeOfEfcApplication": 1,
            "securityLevel": 0,
            "acCrKeyReference": 0,
            "authKeyReference": 0,
            "versionAndValidityEntry": {
               "version": "1",
               "validFrom": "20250101000000Z"
            }
         }
      ]
   },
   "versionAndValidity": {
      "version": "1",
      "validFrom": "20250101000000Z"
   },
   "actionCode": 0
}
```


Sample 12 – Update to contractIssuerList created in Sample 10 (Removal of OBU Type 2)


```
"contractIssuerListAdu": {
   "aduIdentifier": 10000002344,
   "contractIssuerList": {
      "contractIssuerListEntry": [
         {
            "efcContextMark": {
               "contractProvider": {
                  "countryCode": "1100011000",
                  "providerIdentifier": 16360
               },
               "typeOfContract": "1A2B",
               "contextVersion": 1
            },
            "efcContextMarkVersion": 1,
            "equipmentClass": 17,
            "manufacturerId": 63,
            "uniquePartOfPAN": "FF2000",
            "typeOfEfcApplication": 1,
            "securityLevel": 0,
            "acCrKeyReference": 0,
            "authKeyReference": 0,
```

```
            "versionAndValidityEntry": {
               "version": "1",
               "validFrom": "20240801000000Z"
            }
         }
      ]
   },
   "versionAndValidity": {
      "version": "1",
      "validFrom": "20250101000000Z"
   },
   "actionCode": 0
}
```


6.11 EETS CCC Data Response interface

6.11.1 Identification of the Interface

This interface is identified on the relevant portal by the name "*EETS* CCC Data Response – *vn*", where "*vn*" is the version number currently in use.

6.11.2 Purpose of the Interface

The TSP will use the interface to respond to a Compliance Check data request from the TSP. The Compliance Check data will be similar to the CCC data transmitted from the RSE to the OBE in the conventional DSRC-based CCC transaction.

Note that only TSPs which intend to offer an OBE Type 2 to their customers will be required to support this interface.

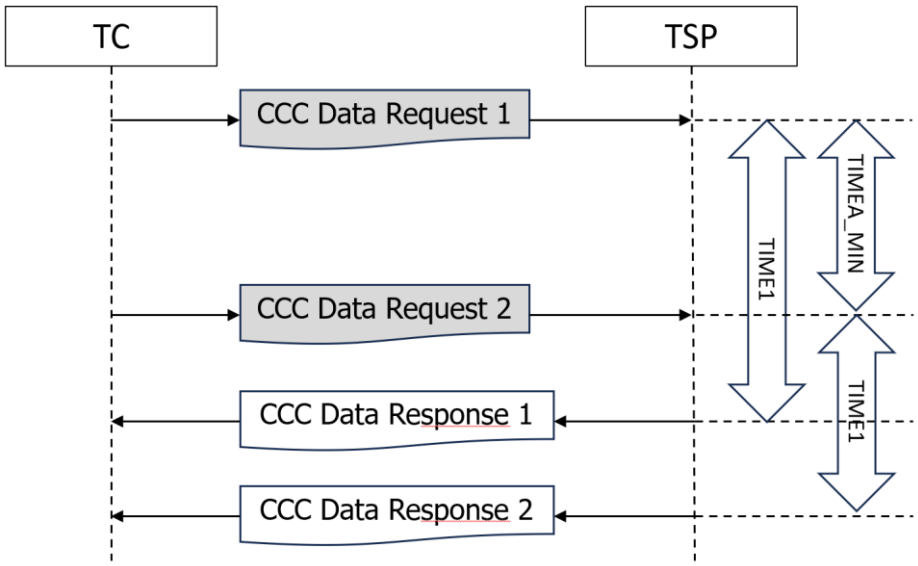
6.11.3 Communications Processes

The CCC Data Response will be transferred using a REST Web-Service interface as defined in section 0. Acknowledgement will only take place at the API level, i.e., it will not be explicitly acknowledged with an AckAdu.

Note that the *CccDataResponseAdu* is not defined in EN 16986, but has been specifically created for this application. It is based on the *ReportCccEventAdu* structure defined in ISO 12855:2022.

The communications sequence is shown below.

Figure 36. Sequence diagram for CCC Data Request transaction



The TSP is required to respond to the CCC Data Request with a CCC Data Response.

TIME1_MIN, the minimum time to respond to a CCC Data Request is 0.

TIME1_MAX, the maximum time to respond to a CCC Data Request is 15 minutes. If after 15 minutes the TSP does not have the status information available (for example, there has been no connectivity to the OBE), the TSP must respond with a *CCC Data Response* with the *cccStatusUnavailable* flag set to 1.

In the case where the TSP has responded with the *cccStatusUnavailable* flag set to 1, the TC will re-request the data up to 24 hours later.

In the case where the TSP has more than one GNSS result for *userInfoValidityPeriod* provided in the *cccDataRequestAdu* only the latest GNSS result, closest to the provided *endOfPeriod*, should be re-turned in the *cccDataResponseAdu*.

For example, if the TSP has captured the following GNSS results from the OBE:

- 2024-08-15 08:23:06
- 2024-08-15 08:23:11
- 2024-08-15 08:23:16
- 2024-08-15 08:23:21
- 2024-08-15 08:23:26

And the *cccDataRequestAdu* has the following period defined in the *userInfoValidityPeriod* field:

- beginOfPeriod: 2024-08-15 08:22:57
- endOfPeriod: 2024-08-15 08:23:17

The following GNSS result must be used in the *cccDataResponseAdu:*

- 2024-08-15 08:23:16

In the case where the TSP has no GNSS results for the *userInfoValidityPeriod* provided in the *cccDataRequestAdu* the *cccDataResponseAdu* should not be populated in the EETS CCC Data Response request.

6.11.4 Message Formats

The CCC Data Response is transmitted in an *InfoExchange* message with a *CCCDataResponseADU* as defined in this document and this must in principle satisfy the restrictions described in the SectionAutonomous profile (with TC dominance) of specification EN 16986.

The CCC Data Response will contain data regarding the status of the OBE. The following data will be reported:

cccStatusUnavailable

The *cccStatusUnavailable* must hold one of the two following values:

- **0** – GNSS result available and populated in *cccDataResponseAdu*
- **1** – No GNSS result available, and the *cccDataResponseAdu* is not included in the response

cccDataResponseAdu

The *cccDataResponseAdu* is only included in the response if *cccStatusUnavailable* has the value 0.

The following fields are defined for the *cccDataResponseAdu:*

Table 32. cccDataResponseAdu

Field	Comments
timeOfCccStatus	Date and Time of the GNSS result returned
locationOfCccStatus	Identifies the GNSS coordinates of the OBE at time of status
cccStatusData	Described in section 0 **cccStatusData**

cccStatusData

The *cccStatusData* contains all the information to be reported for the OBE.

The following fields are defined for the *cccStatusData*:

Table 33. cccStatusData

Field	Comments
cccContext	Identifies the EFC Context Mark for the OBE which is made up of • *contractProvider* (TSP) • *typeOfContract* • *contextVersion* Further details can be found in the API specification

vehicleLpn	Identifies the vehicle license plate number which is made up of • *countryCode* • *alphabetIndicator* • *licencePlateNumber* Further details can be found in the API specification
vehicleClass	Identifies the vehicleClass as defined in EN 15509:2023 Table A2, attribute 17
vehicleWeightLimits	Identifies the weight limits of the vehicle which is made up of • *vehicleTechicalMaxLadenWeight* • *vehicleMaxLadenWeight* • *vehicleTrainMaximumWeight* Further details can be found in the API specification
vehicleSpecificCharacteris-tics	Identifies relevant characteristics of the vehicle which is made up of • ***environmentalCharacteristics*** *(euroValue, copValue, euCO2EmissionsClass)* • ***engineCharacteristics*** *(ISO/TS 17573-03:2021, table 13)* • ***futureCharacteristics*** *(ISO/TS 17573-03:2021, table 3)* Further details can be found in the API specification
ObeId	Identifies the OBE that is being reported on. The ObeId is made up of: • *manufacturerId* • *equipmentObuId* • *obeType* *obeType* must comply to one of the following: • **1** - GNSS Type 1 OBE (including a DSRC element) • **2** - GNSS Type 2 OBE (not including a DSRC element)

obeStatusHistory	Identifies the status of the OBE at the time of reporting. The status history is made up of • ***statusIndicator*** – Indicates the status of the OBE at the time of reporting. One of the following values are expected o **0**: **NO-GO** – OBE is considered "red" or not operational o **1**: **GO** – OBE is considered "green" or fully operational • ***timeWhenChanged*** – Time when *statusIndicator* was changed to its current status. • ***timeWhenActivated*** – Last time the OBE was started being operational (OBE is considered "green"). E.g. when the app started tracking GNSS points. • ***timeWhenObePowered*** – Last time the OBE was powered on, or app was started.

Formal message and data formats can be downloaded from the API Management Developer Portal.

6.11.5 Error Handling

API errors will be handled as described in section 5.5.

The table below shows error codes that are used in validation of this interface.

Table 34. Error codes - CCC Data Response interface

Data	Validation	Acknowledgement
All data fields	The format of the message received was not understood by the recipient.	HTTP 400 Bad request
adus: actionCode	The sent action code is not supported.	HTTP 400 Bad request
apci: apduOriginator	Validate that originator is active in active provider table	HTTP 400 Bad request or 401 Unauthorized

6.11.6 Sample Requests

Sample 13 - cccDataResponseAdu for cccDataRequestAdu from Sample 8 with GNSS result available


```
"InfoExchange": {
   "InfoExchangeContent": {
      "apci": {
         "aidIdentifier": 17,
         "apduOriginator": {
            "countryCode": "0011111100",
            "providerIdentifier": 116
         },
         "informationSenderID": {
            "countryCode": "0011111100",
```

```
            "providerIdentifier": 116
         },
         "informationrecipientID": {
            "countryCode": "1001011110",
            "providerIdentifier": 5
         },
         "apduIdentifier": 151034950004212,
         "inResponseToApduId": {
            "apduOriginator": {
               "countryCode": "1001011110",
               "providerIdentifier": 5
            },
            "apduIdentifier": 151034950002001
         },
         "apduDate": "20240815083112Z"
      },
      "adus": {
         "cccStatusUnavailable": 0,
         "cccDataResponseAdus": {
            "cccDataResponseAdu": {
               "timeOfCccStatus": "20240815083111Z",
               "locationOfCccStatus": {
                  "PointType": {
                     "absolutePointCoordinates": {
                        "longitude": 11433544,
                        "latitude": 55362310
                     }
                  }
               },
               "cccStatusData": [{
                  "cccContext": {
                     "contractProvider": {
                        "countryCode": "0011111100",
                        "providerIdentifier": 116
                     },
                     "typeOfContract": "81F2",
                     "contextVersion": 1
                  },
                  "vehicleLpn": {
                     "countryCode": "1001011110",
                     "alphabetIndicator": "latinAlphabetNo1",
                     "licencePlateNumber": "41423331323332000000000000000"
                  },
                  "vehicleClass": 32,
                  "vehicleWeightLimits": {
                     "vehicleTechicalMaxLadenWeight": 2500,
                     "vehicleMaxLadenWeight": 2500,
                     "vehicleTrainMaximumWeight": 5000
                  },
                  "vehicleSpecificCharacteristics": {
                     "environmentalCharacteristics": {
                        "euroValue": 6,
                        "copValue": 1,
                        "euCO2EmissionClass": 1
                     },
```

```
                     "engineCharacteristics": 4,
                     "futureCharacteristics": 0
                  },
                  "ObeId": {
                     "manufacturerId": 63,
                     "equipmentObuId": "041000055C",
                     "obeType": 2
                  },
                  "obeStatusHistory": {
                     "statusIndicator": 1,
                     "timeWhenChanged": "20240815075234Z",
                     "timeWhenActivated": "20240815075234Z",
                     "timeWhenObePowered": "20240815075123Z"
                  }
               }]
            }
         }
      }
   }
}
```


Sample 14 - cccDataResponseAdu for cccDataRequestAdu from Sample 8 with no GNSS result


```
"InfoExchange": {
   "InfoExchangeContent": {
      "apci": {
         "aidIdentifier": 17,
         "apduOriginator": {
            "countryCode": "0011111100",
            "providerIdentifier": 116
         },
         "informationSenderID": {
            "countryCode": "0011111100",
            "providerIdentifier": 116
         },
         "informationrecipientID": {
            "countryCode": "1001011110",
            "providerIdentifier": 5
         },
         "apduIdentifier": 151034950004213,
         "inResponseToApduId": {
            "apduOriginator": {
               "countryCode": "1001011110",
               "providerIdentifier": 5
            },
            "apduIdentifier": 151034950002002
         },
         "apduDate": "202408150831127Z"
      },
      "adus": {
         "cccStatusUnavailable": 1
      }
   }
}
```

6.12 Trust Objects (TRUSTOBJECT transaction)

Trust objects will not be transmitted over a REST interface. The transfer mechanism will be bilaterally agreed between the TC and each TS. As these transfers are expected to be infrequent, a manual exchange mechanism (e.g., upload via secure web interface) is envisaged.

6.13 Actor Table (EXCHANGEACTORTABLE transaction)

Actor Tables will not be transmitted over a REST interface. The TC will specify the transfer mechanism. As these transfers are expected to be infrequent, a manual exchange mechanism (e.g. secure email, secure file share etc.) is envisaged.

6.14 Compliance Check Communication (CCC) transaction (DSRC)

6.14.1 Purpose of the Interface

The interface is used to enable compliance checking of the OBE of passing vehicles passing a roadside enforcement (RSE) point by allowing the RSE to directly interrogate the OBE using DSRC.

6.14.2 Communications Processes

The communications between the RSE and the OBE will use Dedicated Short-Range Communications (DSRC) according to the ISO 12813:2019 and associated standard.

The following additional specifications apply to this interface:

6.14.3 Message Formats

Message formats will be as defined in ISO 12813:2019.

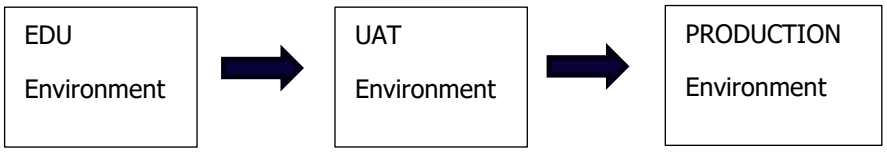
6.14.4 Error Handling

As defined in ISO 12813:2019 and normative references therein.

7 Environments & Data Expectation

The TC has different environments for different purpose. The following process diagram below shows the normal flow and the order of environments that will be opened gradually when the process of going from a sandbox environment to full production-ready APIs.

Figure 37. Environments used for development, test and production



To link between the TC and TSP environments, it is crucial that TSP at least has at least a test and a production environment. In the test environment, or in the TC's EDU and UAT environments, no live production data (or GDPR-relevant data) are sent.

EDU Environment (KMToll-EDU):

Purpose:

This environment is used for the accreditation purpose and used as a sandbox to play with the APIs provided by TC. The Developer Portal also has APIs defined in OpenAPI format that the TSP should use to develop their own APIs.

The endpoints can be tested for format validation and connectivity test. The initial release for v1 of the APIs will not contain the Oauth 2.0 requirements; this will be added in v2. The idea of this environment is "connectivity test" and a format validation check against payload to/from TC to TSPs. This environment is a downscaled environment, and not scaled for any kind of performance test.

Data:

No real production data should be sent to these endpoints.

URLs:

Developer Portal: https://portal-edu.vejafgifter.dk

API Example: https://api-edu.vejafgifter.dk/ack-tc/v1/

UAT Environment (KMToll-UAT):

Purpose:

This environment is used for the integration flow test. The idea of this environment is "integration test" and is connected to multiple systems. This environment has similarity environment as production and is scaled for performance test. The test performed on this environment must be coordinated tests which needs approval from TC.

Data:

No real production data should be sent to these endpoints, and only coordinated tests are allowed.

URLs:

Developer Portal: https://portal-uat.vejafgifter.dk

API Example: https://api-uat.vejafgifter.dk/ack-tc/v1/

PRODUCTION Environment (KMToll-PROD):

Purpose:

This environment is the actual production environment and has integration towards all of the TC's sub-suppliers and 3rd parties.

Data:

Only real production data may be sent to these endpoints.

URLs:

Developer Portal: https://portal.vejafgifter.dk

API Example: https://api.vejafgifter.dk/ack-tc/v1/