ITSS practice group

–

ITSS

Standard Specification

ITSS Interface IF1

(Telematics Application – Customer System)

Version 1.2 final

Copyright Notice

The information made available within this document:

- could include technical or other mistakes, inaccuracies or typographical errors
- may be out of date and the ITSS practice group and its contributors make no commitment to update such materials
- is not intended to be used in safety relevant applications
- is not approved for use in safety-relevant applications

In no event shall the ITSS practice group or its contributors be liable to you or any third parties for any special, punitive, incidental, indirect or consequential damages of any kind, or any damages whatsoever, including, without limitation, those resulting from loss of use, data or profits, whether or not ITSS practice group or its contributors has been advised of the possibility of such damages, and on any theory of liability, arising out of or in connection with the use of the information contained herein.

The telematics application provides telematics data to the customer ERP system; the customer ERP system itself is responsible for the information exchange between the entities as described in the TAF TSI. Therefore, the regulations given in the TAF TSI are not relevant for the ITSS interface IF1 standard specification. The information exchange in compliance with regulation TAF TSI is in the sole responsibility of the customer.

The information provided herein can be used to communicate over different communication links and technologies. Depending on the provider and tariff additional communication costs may occur for each communication link.

Preface

ITSS Interface IF1 describes the data exchange between a Telematics Application and any customer system, as shown in “Diagram 1: Generic System Architecture” marked with the ‘1’ in a red circle.

Content

–

System Architecture Overview

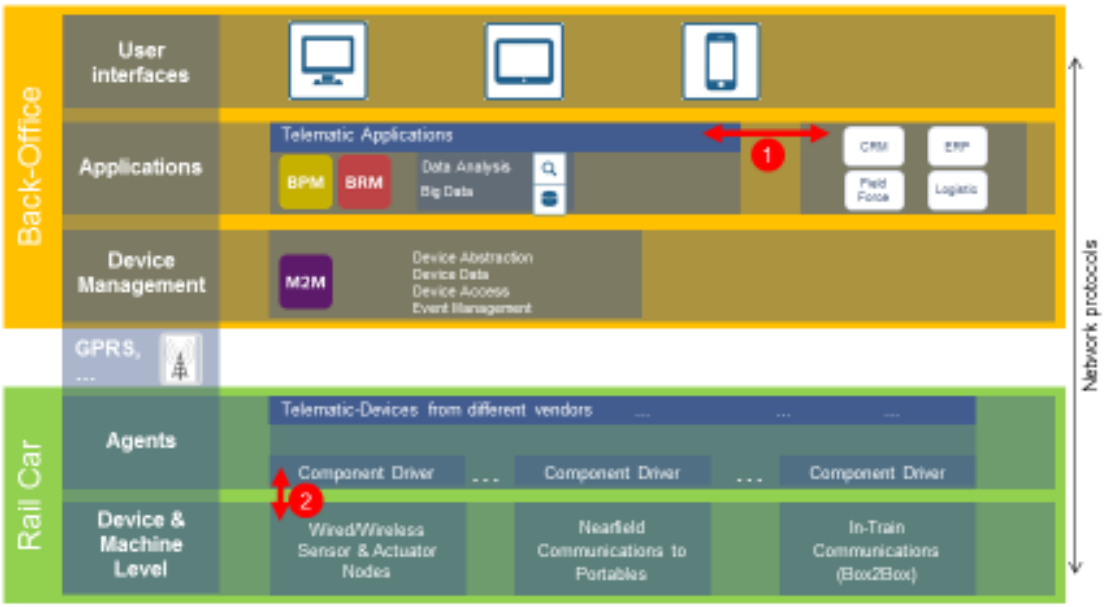


Diagram 1: Generic System Architecture

Interface IF1:

Every system taking part in the communication over ITSS interface IF1 has a unique system ID and a user and password for basic authentication via HTTP headers. The customer system (e.g. customer ERP) has a customer system ID and user and password for basic authentication. The Telematics Application in turns has a telematics system ID and user and password for basic authentication.

Both system IDs should be different and user and password selected and changed based on a secure rule set but the standard imposes no restriction on the IDs and user and password for basic authentication. Furthermore, this standard does not provide any rules for the generation of those IDs and passwords. It is the sole responsibility of the communication parties to choose system ID and user and passwords in a secure manner.

The only security regulation this standard defines is that the communication parties need to agree on those system IDs and user and passwords and that the system ID and user and password need to be validated by both systems prior to each data exchange. If a security violation is detected, e.g. the authentication header does not match for the given system ID or the system ID is unknown, the receiving system has to discard all received data silently. A security violation warning might be raised by the detecting system but this is out of the scope of this standard.

Although it is advisable to make those IDs unique, no restrictions are given from the standard. It is the responsibility of the communication parties to manage the IDs for all their communication requirements.

The standard does not provide any means for the allocation of unique system IDs.

Communication parties for interface IF1:

ITSS Interface IF1 describes the communication between a telematics application and a customer system.

– A telematics application can communicate with more than one customer system and a customer system can communicate to multiple telematics applications.

The customer is able to connect with as many systems to the telematics application as required, e.g. customer ERP, customer logistics, customer CRM. The following sections refer to all those systems in general as the customer system.

Process description of pairing telematics devices with transport devices

The customers have arbitrary requirements with respect to where the pairing (coupling) of the telematics devices with the transport devices (e.g. wagons) is done. Some customers require that the transport device ID is not disclosed to the telematics application provider, this requires that the pairing is maintained in the customer system. Other customers leave the pairing to the telematics application provider accessing the telematics data either by transport device ID or telematics device ID.

This standard specification supports both approaches of pairing telematics devices with transport devices; either pairing is done in the customer system (option A) or pairing is done in the telematics application (option B).

Depending of which option has been selected in the mutual agreement between customer and telematics application provider before implementation of the interface, either the customer system is responsible of maintaining the correct matching of telematics device ID and transport device ID (option A) or the telematics application (option B).

Option A:

Requests from the customer system to the telematics application must include the telematics device ID for proper identification. Responses and events from the telematics application to the customer system must also include the telematics device ID.

Option B:

For proper object identification for option B, requests from the customer system to the telematics application must include either the transport device ID (preferred) or the matching telematics device ID, if the customer system has gained knowledge of the proper pairing. Responses from the telematics application to the customer system must include the ID as provided by the calling system. This can either be the transport device ID or the telematics device ID. The telematics application can provide both IDs in the response, using its internal matching to find the other ID. This functionality is not required by the standard but recommended to create uniform information in responses and events. Events issued from the telematics application to the customer system must always include the telematics device ID and may also include the transport device ID to give the customer system all information for a proper identification of the object causing the event.

Push-Notification (Event) reception

The telematics application can notify the Customer System via events about new values or problems connected to a transport device. This notification uses REST over HTTP and communicates with the Customer System using a dedicated customer specific URI. The communicating parties, that is, the telematics application owner and the Customer System owner, need to agree on the URI for those event notifications. For security the same principles as described in the chapter Data security are applied, that is, the telematics application uses the Customer System ID and user and password for basic authentication as agreed upon by both communicating parties.

–

If the telematics application does not know an event notification URI for a given System ID no events will be delivered.

The URI construction must include the part “itss” and follow the versioning as described in the chapter Versioning of the REST web services. Therefore, in all event descriptions for this document an URI in the following form is used:

https://{customerURI}/itss/1.2/shockDetected

The part {customerURI} is the part that must be agreed upon by both communicating parties. All other parts of the URI are regulated by this specification.

A notification happens whenever the telematics application creates or receives new information from a telematics device or a sensor. If a customerURI is known to the telematics application but the delivery of a notification fails due to any possible reason, the telematics application will store the notification for at least 24 hours. As soon as the connection becomes available again, all buffered notifications will be transmitted to the customer system.

Notifications older than 24 hours can be silently removed from the buffer by the telematics application and will not be transmitted to the customer system. The data, e.g. sensor values, will still be stored by the telematics application and can be explicitly requested by the customer system.

Notifications from the telematics application to the customer system cannot be stopped by the customer system. Once the customerURI is known to the telematics application any notification for the specific customer system is buffered and pushed to this system.

It can be mutually agreed between the communicating partners, which event messages are to be sent as push notifications.

Strategy for missing data

If the telematics application is not able to deliver all data a request contains and the value is marked as not required (required: false), the telematics application does not include the element concerned and tries to satisfy the request or event with as much data as possible. A prominent example for a case of missing data is the GNSS position connected to an event. If the telematics device is not delivering this position to the telematics application the telematics application still delivers the event to the Customer System without the “GNSS_Position” tag.

The result of this strategy is that requests might be answered with a valid response even if this response does not contain any relevant information, e.g. *lastPosition* request without known GNSS position of the transport device. It is not allowed to indicate this condition with HTTP error codes, e.g. 204 (no content).

Versioning of the REST web services

If the specification (API or parameters) for a web service changes in a future version this will be represented by a part of the URI, which is used to call the respective service.

https://telematik.xyz.com/itss/**1.0**/lastPosition?ITSS_TransportDeviceID=3180%204674%20%20001-1&ITSS_CustomerSystemID=custSys4711&ITSS_PassPhrase=open%20sesame

The version number is placed between the base URI of the specific server and the name of the webservice. Moreover, the version number used in the URI corresponds directly to the version number of the specification. Hence, the full API is accessible via the versioned requests even if a specific request/response did not change from one version to another. The following example should clarify the API versioning:

From version 1.0 to version 1.1 the specification of the lastPosition changed. The specification for mileageTimeInterval did not change. Both versions would be accessible via:

Version 1.0 (request parameter and response according to ITSS specification V1.0):

https://telematik.xyz.com/itss/**1.0**/lastPosition?ITSS_TransportDeviceID=3180%204674%20%20001-1&ITSS_CustomerSystemID=custSys4711&ITSS_PassPhrase=open%20sesame

https://telematik.xyz.com/itss/request/**1.0**/mileage?ITSS_TransportDeviceID=3180%204674%20001-1&fromUTCtimestamp=1436700000&toUTCtimestamp=1436792541&ITSS_CustomerSystemID=custSys4711&ITSS_PassPhrase=open%20sesame

Version 1.1 (request parameter and response according to ITSS specification V1.1):

https://telematik.xyz.com/itss/**1.1**/lastPosition?ITSS_TransportDeviceID=3180%204674%20%20001-1&ITSS_CustomerSystemID=custSys4711&ITSS_PassPhrase=open%20sesame

https://telematik.xyz.com/itss/request/**1.1**/mileage?ITSS_TransportDeviceID=3180%204674%20001-1&fromUTCtimestamp=1436700000&toUTCtimestamp=1436792541&ITSS_CustomerSystemID=custSys4711&ITSS_PassPhrase=open%20sesame

Version 1.2 (request parameter and response according to ITSS specification V1.2):

https://telematik.xyz.com/itss/**1.2**/lastPosition?ITSS_TransportDeviceID=3180%204674%20%20001-1&ITSS_CustomerSystemID=custSys4711

https://telematik.xyz.com/itss/request/**1.2**/mileage?ITSS_TransportDeviceID=3180%204674%20001-1&fromUTCtimestamp=1436700000&toUTCtimestamp=1436792541&ITSS_CustomerSystemID=custSys4711

–

although the request and response for the mileage calls are identical in both cases.

If the telematics application supports multiple versions of the ITSS interface IF1 a customer system can freely use requests across those supported versions.

As the telematics application does not provide a version request the telematics applicationcommunicating parties shall agree on the versions used for the respective application themselves.

Released versions

Version 1.0: First release of the ITSS specification IF1

Version 1.1: Updated release of the ITSS specification IF1 with additional requests and notifications. For details see section Change log.

Version 1.2: Removed the pass phrases from the communication between telematics application and customer system. Instead, added basic authentication for security as a mandatory feature. Added assembled notification and geofence configuration. More details at section Change log.

Request last known position

1. Description

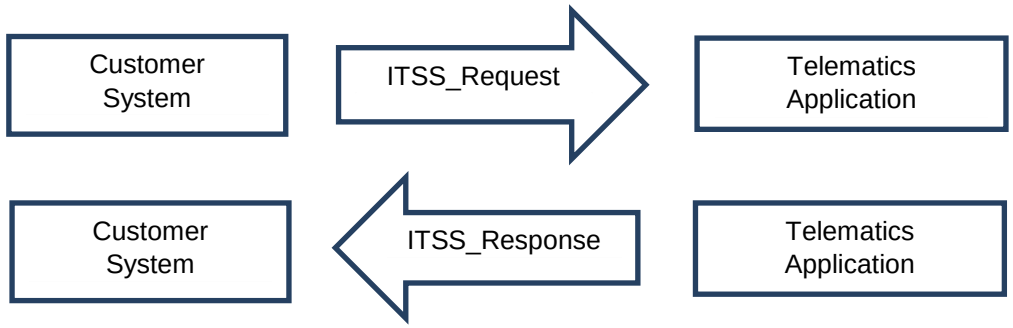
The customer wants to get information about the last known position (geo coordinates) of a specific transport device.

2. Method (Request / Response)

The customer system requests the last known position of a transport device identified by the

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID

The telematics application processes the request and responds with the required information.



Access method: Synchronous

lastPosition

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_CustomerSystemID

lastPosition response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- GNSS_Position
- ITSS_LocationInfo (optional)
- ITSS_TelematicsApplicationID

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/lastPosition?ITSS_TransportDeviceID={deviceId}& ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/lastPosition?ITSS_TelematicsDeviceID={deviceId}& ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json
BODY: json Schema	

```
{
    “title”: “lastPosition”,
    "$schema": "http://json-schema.org/draft-04/schema#",
    “type”:“object”,
    “properties”:
    {
        “ITSS_TransportDeviceID”:      { "type": "string", required:false  },
        “ITSS_TelematicsDeviceID”:     { "type": "string", required:true  },
        “GNSS_Position” :
        {
            “type”: “object”, required: false
            "properties":
            {
                "GNSS_UTCtimestamp ":  { "type": "number", required:true  },
                ”GNSS_Latitude":       { "type": "number", required:true },
                "GNSS_Longitude ":     { "type": "number", required:true },
                "GNSS_Speed_kmph ":    { "type": "number", required:false },
                "GNSS_Heading_deg ":   { "type": "number", required:false },
                “GNSS_Altitude “:      { “type”: “number”, required:false },
                "GNSS_Accuracy ":      { "type": "number", required:false },

                “ITSS_LocationInfo”:          { "type": "object", required:false,
                "properties":
                {
                    “Location_ZIP”:           { "type": "string", required: false },
                    “Location_City”:          { "type": "string", required: false },
                    “Location_Street”:        { "type": "string", required: false },
                    “Location_Description”:   { "type": "string", required: false },
                    “Location_Country”:       { "type": "string", required: false },
                    “Location_UIC_Code”:      { "type": "string", required: false },
                    “Location_GeoZone”:  { "type": "string", required: false }
                }

            }
        },
        “ITSS_TelematicsApplicationID”:    { "type": "string", required:true  }
```

	} }

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“GNSS_Position” :
	{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1,
	“ITSS_LocationInfo”: {
	“Location_ZIP”: 38126,
	“Location_City”: “Braunschweig”,
	“Location_Street“: “Berliner Platz”,
	“Location_Description": “Braunschweig Hbf”,
	“Location_Country”: “Germany”,
	“Location_UIC_Code”: “051”,
	“Location_GeoZone”: “DE”
	}
	},
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}
	–

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Assembled Notification

1. Description

The customer expects to be notified of any kind of event for a transport device / telematics device, where several notifications are caused by one single event (e.g. a new geoposition)

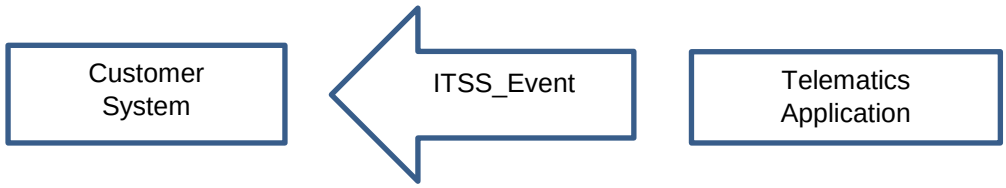
This notification allows to provide all described notifications to be assembled and pushed to the customer system in one single post.

Additional detailed information can be found in the description of the single notifications that are included in this assembled notification.

–

2. Method (Event based notification)

The telematics application sends an event notification containing all relevant information that has been triggered by the single event.



Access method: Event message

assembledNotification:

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- UTCTimeStamp
- GNSS_Position (optional)
- mileage (optional)
- loadingState (optional)
- payload (optional)
- ITSS_SensorValueList (optional)
- ITSS_GeofenceEventList (optional)
- ITSS_MovementState (optional)
- derailment_triggered (optional)
- X-Axis_triggered, Y-Axis_triggered, Z-Axis_triggered (optional)
- X-Axis, Y-Axis, Z-Axis (optional)
- ITSS_TelematicsApplicationID

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/assembledNotification
BODY: json Schema	{ "title": "assembledNotification", "$schema": "http://json-schema.org/draft-04/schema#", "type":"object", "properties": { "ITSS_TransportDeviceID": { "type": "string", required:false }, "ITSS_TelematicsDeviceID": { "type": "string", required:true }, "UTCtimestamp": { “type”: “number”, required:true }, "GNSS_Position" : { "type": "object", required: false "properties": { "GNSS_UTCtimestamp": { "type": "number", required:true }, "GNSS_Latitude": { "type": "number", required:true }, "GNSS_Longitude": { "type": "number", required:true }, "GNSS_Speed_kmph": { "type": "number", required:false }, "GNSS_Heading_deg": { "type": "number", required:false }, "GNSS_Altitude": { "type": "number", required:false }, "GNSS_Accuracy": { "type": "number", required:false }, "ITSS_LocationInfo": { "type": "object", required:false, "properties": { "Location_ZIP": { "type": "string", required: false }, "Location_City": { "type": "string", required: false }, "Location_Street": { "type": "string", required: false }, "Location_Description": { "type": "string", required: false }, "Location_Country": { "type": "string", required: false }, "Location_UIC_Code": { "type": "string", required: false },

```
                    "Location_GeoZone":        { "type": "string", required: false }
                }
            }
        },
        "mileage":                              { "type": "number", required:false},
        "loadingState":                         { "type": "string", required: false},
        "payload":                              { “type”: “number”, required: false},
        "ITSS_SensorValueList" :                { "type": "array", required:false }
        {
            "ITSS_SensorValue":                 { "type": "object", required: false }
            "properties":
            {
                "SamplingUTCTimestamp":         { "type": "number", required: true },
                "ITSS_SensorId":                { "type": "string", required: true },
                "Value":                        { "type": "float", required: true },
                "ITSS_SensorType":              { "type": "string", required: true },
                "ITSS_SensorPosition":          { "type": "string", required: true }
            }
        },
        "ITSS_GeofenceEventList":               {"type": "array", required: false,
            "items":
            {
                "type": "object", required: false
                "properties":
                {
                    "UTCtimestamp":                     { "type": "number", required: true },
                    "ITSS_Geofence"
                    {
                        "type": "object", required: true,
                        "properties":
                        {
                            "GeofenceID":               { "type": "string", required: true },
                            "GeofenceName":             { "type": "string", required: false }
                        }
                    }
                    "ITSS_GeofenceEventTrigger":        { "type": "string", required: true }
                }
            }
        },
        "ITSS_MovementState":                   { "type": "string", required: false },
        "derailment_triggered":                 { "type": "boolean", required: false },
        "X-Axis_triggered":                     { "type": "boolean", required: false },
        "Y-Axis_triggered":                     { "type": "boolean", required: false },
```

	"Z-Axis_triggered": { "type": "boolean", required: false }, "X-Axis": { "type": "number", required: false }, "Y-Axis": { "type": "number", required: false }, "Z-Axis": { "type": "number", required: false }, "ITSS_TelematicsApplicationID": { "type": "string", required:true} } }

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“UTCtimestamp”: 1436712339.124,
	“GNSS_Position” :
	{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1,
	“ITSS_LocationInfo”: {
	“Location_ZIP”: 38126,
	“Location_City”: “Braunschweig”,
	“Location_Street“: “Berliner Platz”,
	“Location_Description": “Braunschweig Hbf”,
	“Location_Country”: “Germany”,
	“Location_UIC_Code”: “051”,
	“Location_GeoZone”: “DE”
	}
	},
	“derailment_triggered”: true
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Notification about the last position

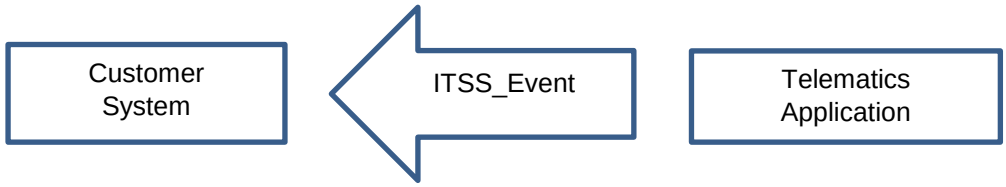
1. Description

The customer expects to be notified of a new known position of a transport device.

This information is provided only for diagnostic purposes. At no time, it shall be used to derive safety related information or actions.

2. Method (Event based notification)

The telematics application sends an event notification containing the new position as a GNSS_Position. If the transport device is in motion, the transmitted GNSS_Position may differ from the real position of the transport device when the customer system receives this notification.



Access method: Event message

lastPosition

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- GNSS_Position
- ITSS_LocationInfo (optional)
- ITSS_TelematicsApplicationID

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/lastPosition
BODY: json Schema	{ “title”: “lastPosition”, "$schema": "http://json-schema.org/draft-04/schema#", “type”:“object”, “properties”:

<table>
<tr><td></td><td>
{
“ITSS_TransportDeviceID”: { "type": "string", required:false },
“ITSS_TelematicsDeviceID”: { "type": "string", required:true },
“GNSS_Position” :
{
“type”: “object”, required: false
"properties":
{
"GNSS_UTCtimestamp ": { "type": "number", required:true },
”GNSS_Latitude": { "type": "number", required:true },
"GNSS_Longitude ": { "type": "number", required:true },
"GNSS_Speed_kmph ": { "type": "number", required:false },
"GNSS_Heading_deg ": { "type": "number", required:false },
“GNSS_Altitude “: { “type”: “number”, required:false },
"GNSS_Accuracy ": { "type": "number", required:false },
“ITSS_LocationInfo”: { "type": "object", required:false,
"properties":
{
“Location_ZIP”: { "type": "string", required: false },
“Location_City”: { "type": "string", required: false },
“Location_Street”: { "type": "string", required: false },
“Location_Description”: { "type": "string", required: false },
“Location_Country”: { "type": "string", required: false },
“Location_UIC_Code”: { "type": "string", required: false },
“Location_GeoZone”: { "type": "string", required: false }
}
}
},
“ITSS_TelematicsApplicationID”: { "type": "string", required:true }
}
}
</td></tr>
<tr><td>BODY example</td><td>
{
“ITSS_TransportDeviceID”: “3180 4674 001-1”,
“ITSS_TelematicsDeviceID”: “MANUF000000751”,
“GNSS_Position” :
{
“GNSS_UTCtimestamp”: 1436712345.154,
“GNSS_Latitude”: 52.264304,
“GNSS_Longitude”: 10.525537,
“GNSS_Speed_kmph”: 48.87,
“GNSS_Heading_deg”: 350.1,
“ITSS_LocationInfo”: {
“Location_ZIP”: 38126,
“Location_City”: “Braunschweig”,
“Location_Street“: “Berliner Platz”,
“Location_Description": “Braunschweig Hbf”,
“Location_Country”: “Germany”,
“Location_UIC_Code”: “051”,
“Location_GeoZone”: “DE”
}
},
“ITSS_TelematicsApplicationID”: “TeleApp0815”

}
</td></tr>
</table>

Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Request positions for a time interval

1. Description

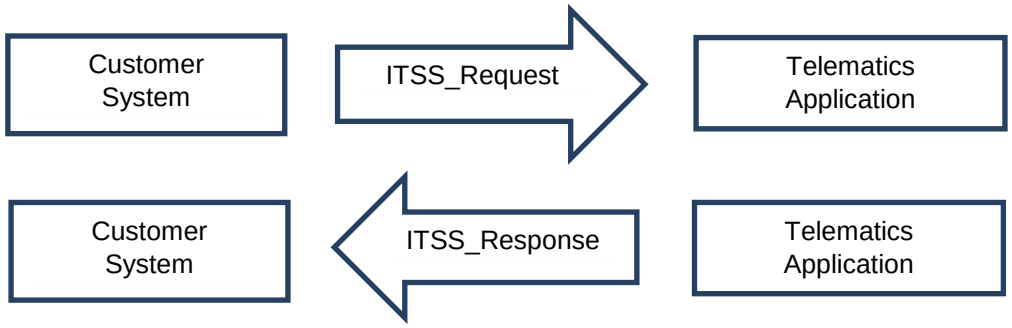
The customer system requests information about the position of a transport device in a given time interval (e.g. transportation trace).

2. Method (Request / Response)

The customer system requests a list of positions in the given time interval for one transport device identified by

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID

The telematics application responds with the requested list of positions. The GNSS_PositionList might be empty, if no position is contained in the specified time interval.



Access method: Synchronous

positionsTimeInterval

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- From_UTCtimestamp: UTCtimestamp – interval begin date (inclusive)
- To_UTCtimestamp: UTCtimestamp interval end date (exclusive)
- ITSS_CustomerSystemID

positionsTimeInterval response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- GNSS_PositionList [
  - GNSS_Position,
  - ITSS_LocationInfo (optional)]
- ITSS_TelematicsApplicationID

The returned GNSS_PositionList is ordered starting at the From_UTCtimestamp (first array entry) to the To_UTCtimestamp (last array entry). The timestamps returned can differ from the

requested interval such that the first array entry might start after the requested start time (From_UTCtimestamp) and the last array entry might stop before the requested stop time (To_UTCtimestamp). The telematics application uses the data available from the transport device and as such might not have data at all for the requested time interval. This also implies that the interval between two GNSS_PositionList entries in the returned data might differ from entry to entry including large gaps.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/positionsTimeInterval?ITSS_TransportDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/positionsTimeInterval?ITSS_TelematicstDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json
BODY: json Schema	{ “title”: “positionsTimeInterval”, "$schema": "http://json-schema.org/draft-04/schema#", “type”:“object”, “properties”: { “ITSS_TransportDeviceID”: { "type": "string", required:false }, “ITSS_TelematicsDeviceID”: { "type": "string", required:true }, “GNSS_Position_List” : { “type”: “array”, “items”: { "description": " GNSS_Position", "type": "object", "properties": { "GNSS_UTCtimestamp ": { "type": "number", required:true }, ”GNSS_Latitude": { "type": "number", required:true }, "GNSS_Longitude ": { "type": "number", required:true }, "GNSS_Speed_kmph ": { "type": "number", required:false }, "GNSS_Heading_deg ": { "type": "number", required:false }, “GNSS_Altitude “: { “type”: “number”, required:false }, "GNSS_Accuracy ": { "type": "number", required:false }, “ITSS_LocationInfo”: { "type": "object", required:false, "properties": {

	“Location_ZIP”: { "type": "string", required: false }, “Location_City”: { "type": "string", required: false }, “Location_Street”: { "type": "string", required: false }, “Location_Description”: { "type": "string", required: false }, “Location_Country”: { "type": "string", required: false }, “Location_UIC_Code”: { "type": "string", required: false }, “Location_GeoZone”: { "type": "string", required: false } } } } }, “ITSS_TelematicsApplicationID”: { "type": "string", required:true } } }

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“GNSS_Position_List” : [
	{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1
	},{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1,
	“ITSS_LocationInfo”: {
	“Location_ZIP”: 38126,
	“Location_City”: “Braunschweig”,
	“Location_Street“: “Berliner Platz”,
	“Location_Description": “Braunschweig Hbf”,
	“Location_Country”: “Germany”,
	“Location_UIC_Code”: “051”,
	“Location_GeoZone”: “DE”
	}
	}],
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	
	}

Response on empty list	
HTTP Status	200
MIME Type	application/json
BODY:	{ “GNSS_Position_List” : [] }
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept

Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Request the mileage of a transport device

1. Description of the use case

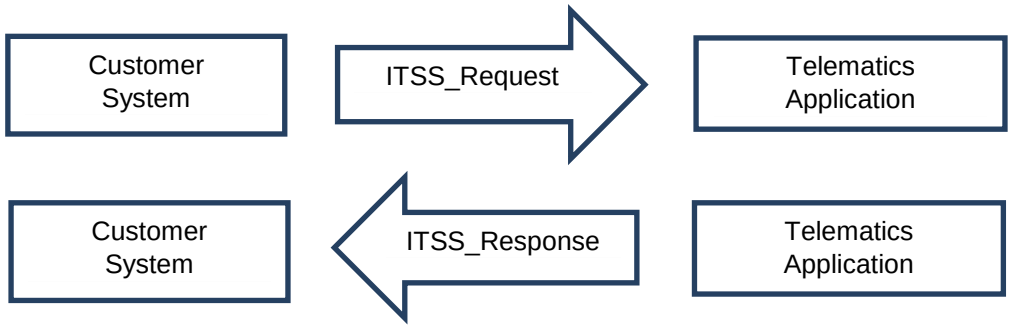
The customer system requests the mileage of a transport device within a specified time interval.

2. Method (Request / Response)

The customer system requests the mileage of one specific transport device travelled within a given time interval. The transport device is identified by

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID

The telematics application responds with the mileage travelled in the specified time interval. The mileage reported is related solely to the mileage travelled while the specific transport device has been equipped with a telematics device being known to the telematics application.



Access Method: Synchronous

mileageTimeInterval

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- From_UTCtimestamp: UTCtimestamp – interval begin date (inclusive)
- To_UTCtimestamp: UTCtimestamp – interval end date (exclusive)
- ITSS_CustomerSystemID

mileageTimeInterval response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- From_UTCtimestamp: UTCtimestamp
- To_UTCtimestamp: UTCtimestamp
- mileage: number in meters
- ITSS_TelematicsApplicationID

The returned mileage represents the information available to the telematics application for the requested time interval. This does not imply the accurate start at the From_UTCtimestamp and end at the To_UTCtimestamp. The mileage returned is generated from available information and as such might start after the requested start time (From_UTCtimestamp) and might stops before the requested stop time (To_UTCtimestamp). The returned timestamps (From_UTCtimestamp and To_UTCtimestamp) indicate the start and stop of the interval as used in the calculation of the mileage. The telematics application uses the data available from the transport device and as such might not have data at all for the requested time interval.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/mileageTimeInterval?ITSS_TransportDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/mileageTimeInterval?ITSS_TelematicstDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json

BODY:
json Schema
	{
	“title”: “mileageTimeInterval”,
	"$schema": "http://json-schema.org/draft-04/schema#",
	“type”:“object”,
	“properties”:
	{
	“ITSS_TransportDeviceID”: { "type": "string", required:false },
	“ITSS_TelematicsDeviceID”: { "type": "string", required:true },
	“mileage”: { "type": "number", required:true },
	“ITSS_TelematicsApplicationID”: { "type": "string", required:true }
	}
	}

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“mileage”: 10000000,
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Notification about the mileage

1. Description

The customer expects to be notified of a new known mileage of a transport device.

2. Method (Event based notification)

The telematics application sends an event notification containing the new mileage.

Access method: Event message

mileage

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- mileage: number in meters
- UTCTimeStamp
- ITSS_TelematicsApplicationID

The mileage represents the last known information available to the telematics application and as that is always an accumulated, absolute mileage.

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/mileage
BODY: json Schema	{ “title”: “mileage”, "$schema": "http://json-schema.org/draft-04/schema#", “type”:“object”, “properties”: { “ITSS_TransportDeviceID”: { "type": "string", required:false }, “ITSS_TelematicsDeviceID”: { "type": "string", required:true }, “mileage”: { "type": "number", required:true },

	“UTCtimestamp”: { “type”: “number”, required: true }, “ITSS_TelematicsApplicationID”: { "type": "string", required:true } } }

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“mileage”: 10000000,
	“UTCtimestamp”: 1436712339.124,
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Request the loading state

1. Description of the use case

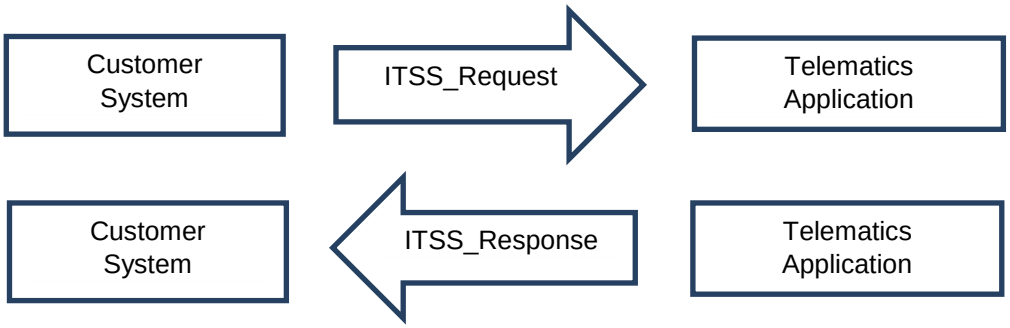
The customer wants to know the last known loading state of a transport device.

2. Method (Request / Response)

The customer system requests the last known loading state of one transport device related to the given

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID.

– The telematics application responds with the last known loading state of the specific transport device.



Access Method: Synchronous

loadingState

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_CustomerSystemID

loadingState response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- UTCtimestamp
- GNSS_Position
- ITSS_LocationInfo (optional)
- ITSS_LoadingState
- payload: number in kg (optional)
- ITSS_TelematicsApplicationID

The UTCtimestamp defines the time when the reported loading state was evaluated. It can differ from the GNSS_UTCtimestamp as the GNSS_UTCtimestamp states the time of the GNSS position evaluation.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/loadingState?ITSS_TransportDeviceID={deviceId}& ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/loadingState?ITSS_TelemacticsDeviceID={deviceId}& ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json
BODY: json Schema	{ “title”: “loadingState”, "$schema": "http://json-schema.org/draft-04/schema#", “type”:“object”, “properties”: { “ITSS_TransportDeviceID”: { "type": "string", required:false }, “ITSS_TelematicsDeviceID”: { "type": "string", required:true }, “UTCtimestamp”: { “type”: “number”, required: true } “GNSS_Position” : { “type”: “object”, required: false "properties": { "GNSS_UTCtimestamp ": { "type": "number", required:true }, ”GNSS_Latitude": { "type": "number", required:true }, "GNSS_Longitude ": { "type": "number", required:true }, "GNSS_Speed_kmph ": { "type": "number", required:false }, "GNSS_Heading_deg ": { "type": "number", required:false }, “GNSS_Altitude “: { “type”: “number”, required:false }, "GNSS_Accuracy ": { "type": "number", required:false }, “ITSS_LocationInfo”: { "type": "object", required:false, "properties": { “Location_ZIP”: { "type": "string", required: false }, “Location_City”: { "type": "string", required: false }, “Location_Street”: { "type": "string", required: false }, “Location_Description”: { "type": "string", required: false }, “Location_Country”: { "type": "string", required: false }, “Location_UIC_Code”: { "type": "string", required: false }, “Location_GeoZone”: { "type": "string", required: false } } } }, “loadingState”: { "type": "string", required: true }, “payload”: { “type”: “number”, required: false },

	“ITSS_TelematicsApplicationID”: { "type": "string", required: true } } }

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“UTCtimestamp”: 1436712339.124,
	“GNSS_Position” :
	{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1,
	“ITSS_LocationInfo”: {
	“Location_ZIP”: 38126,
	“Location_City”: “Braunschweig”,
	“Location_Street“: “Berliner Platz”,
	“Location_Description": “Braunschweig Hbf”,
	“Location_Country”: “Germany”,
	“Location_UIC_Code”: “051”,
	“Location_GeoZone”: “DE”
	}
	},
	“loadingState”: “loaded”,
	“payload”: 20145
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Notification about the loading state

1. Description

The customer expects to be notified of a new known loading state of a transport device.

2. Method (Event based notification)

The telematics application sends an event notification containing the new loading state.

Access method: Event message

loadingState

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- UTCtimestamp
- GNSS_Position
- ITSS_LocationInfo (optional)
- ITSS_LoadingState
- payload: number in kg (optional)
- ITSS_TelematicsApplicationID

The UTCtimestamp defines the time when the reported loading state was evaluated. It can differ from the GNSS_UTCtimestamp as the GNSS_UTCtimestamp states the time of the GNSS position evaluation.

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/loadingState
BODY: json Schema	{ “title”: “loadingState”, "$schema": "http://json-schema.org/draft-04/schema#", “type”:“object”, “properties”: { “ITSS_TransportDeviceID”: { "type": "string", required:false },

	“ITSS_TelematicsDeviceID”: { "type": "string", required:true },
	“UTCtimestamp”: { “type”: “number”, required: true }
	“GNSS_Position” :
	{
	“type”: “object”, required: false
	"properties":
	{
	"GNSS_UTCtimestamp ": { "type": "number", required:true },
	”GNSS_Latitude": { "type": "number", required:true },
	"GNSS_Longitude ": { "type": "number", required:true },
	"GNSS_Speed_kmph ": { "type": "number", required:false },
	"GNSS_Heading_deg ": { "type": "number", required:false },
	“GNSS_Altitude “: { “type”: “number”, required:false },
	"GNSS_Accuracy ": { "type": "number", required:false },
	“ITSS_LocationInfo”: { "type": "object", required:false,
	"properties":
	{
	“Location_ZIP”: { "type": "string", required: false },
	“Location_City”: { "type": "string", required: false },
	“Location_Street”: { "type": "string", required: false },
	“Location_Description”: { "type": "string", required: false },
	“Location_Country”: { "type": "string", required: false },
	“Location_UIC_Code”: { "type": "string", required: false },
	“Location_GeoZone”: { "type": "string", required: false }
	}
	}
	},
	“loadingState”: { "type": "string", required: true },
	“payload”: { “type”: “number”, required: false },
	“ITSS_TelematicsApplicationID”: { "type": "string", required:true }
	}
	}

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“UTCtimestamp”: 1436712339.124,
	“GNSS_Position” :
	{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1,
	“ITSS_LocationInfo”: {
	“Location_ZIP”: 38126,
	“Location_City”: “Braunschweig”,
	“Location_Street“: “Berliner Platz”,
	“Location_Description": “Braunschweig Hbf”,
	“Location_Country”: “Germany”,
	“Location_UIC_Code”: “051”,
	“Location_GeoZone”: “DE”
	}
	},
	“loadingState”: “loaded”,
	“payload”: 20145
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Request all known devices

1. Description of the use case

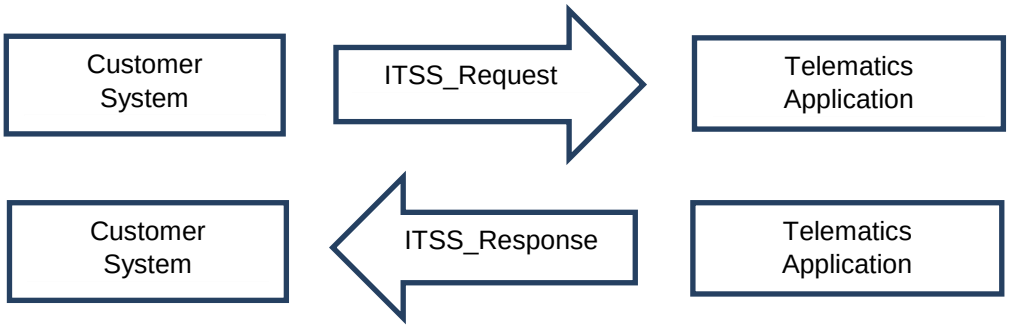
The customer system wants to get a list of all ITSS_TelematicsDeviceIDs known to the telematics application. If the telematics application also knows the mapping of ITSS_TelematicsDeviceID and ITSS_TransportDeviceID the request will return both IDs. Otherwise the returned list only contains the ITSS_TelematicsDeviceID.

2. Method (Request / Response)

The customer system requests the list of all known devices related to the given

- ITSS_CustomerSystemID

The telematics application responds with the list of known devices for this customer system and, if available, the mapping to the ITSS_TransportDeviceID.



Access Method: Synchronous

allDevices

- ITSS_CustomerSystemID

allDevices response

- UTCTimeStamp
- ITSS_DeviceList [
  - ITSS_TransportDeviceID (optional),
  - ITSS_TelematicsDeviceID]
- ITSS_TelematicsApplicationID

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/allDevices?ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json
BODY: json Schema	{ “title”: “allDevices”, "$schema": "http://json-schema.org/draft-04/schema#", “type”:“object”, “properties”: { “UTCtimestamp”: { “type”: “number”, required: true }, “ITSS_DeviceList” : [{ “ITSS_TransportDeviceID”: { "type": "string", required: false }, “ITSS_TelematicsDeviceID”: { "type": "string", required: true } }], “ITSS_TelematicsApplicationID”: { "type": "string", required: true } } }
BODY example	{ “UTCtimestamp”: 1436712339.124, "ITSS_DeviceList": [{“ITSS_TransportDeviceID”: “3180 4674 001-1”, “ITSS_TelematicsDeviceID”: “MANUF000000751”}, {“ITSS_TransportDeviceID”: “3180 4674 001-2”, “ITSS_TelematicsDeviceID”: “MANUF000000752”}, {“ITSS_TransportDeviceID”: “3180 4674 001-3”, “ITSS_TelematicsDeviceID”: “MANUF000000753”}], "ITSS_TelematicsApplicationID": "TeleApp0815" }
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Request sensor values for a time interval

1. Description of the use case

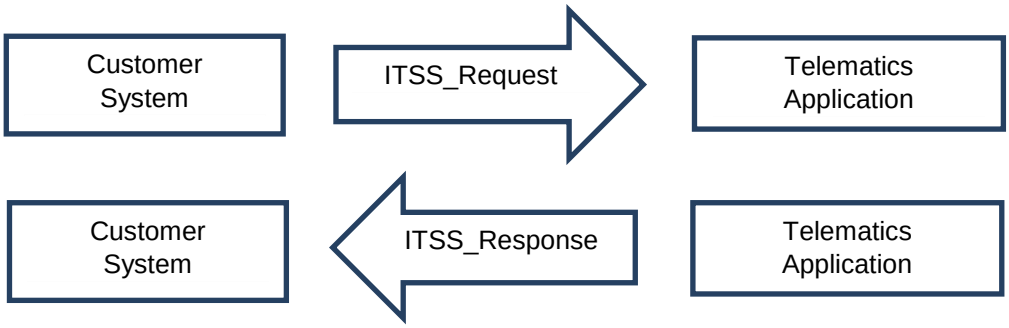
The customer requests information about the sensor values of a telematics device in a given time interval.

2. Method (Request / Response)

The customer system requests a list of sensor values in the given time interval for one telematics device identified by

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID

The telematics application responds with the requested list of sensor values. The ITSS_SensorValueList might be empty, if no sensor values are contained in the specified time interval.



Access Method: Synchronous

sensorValuesTimeInterval

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- From_UTCtimestamp: UTCtimestamp – interval begin date (inclusive)
- To_UTCtimestamp: UTCtimestamp interval end date (exclusive)
- ITSS_CustomerSystemID

sensorValuesTimeInterval response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_SensorValueList [
  - ITSS_SensorValue]
- ITSS_TelematicsApplicationID

The returned ITSS_SensorValueList contains none or one or multiple entries of ITSS_SensorValues (see section Glossary: for details), ordered starting at the From_UTCtimestamp (first array entry) to the To_UTCtimestamp (last array entry). The timestamps returned can differ from the requested interval such that the first array entry might start after the requested start time (From_UTCtimestamp) and the last array entry might stop before the requested stop time (To_UTCtimestamp). The telematics application uses the data available from the telematics device and as such might not have data at all for the requested time interval. This also implies that the interval between two ITSS_SensorValueList entries in the returned data might differ from entry to entry including large gaps.

If a sensor sends data directly to the telematics application, e.g. by using some kind of gateway which is not a telematics device, the ITSS_TelematicsDeviceID must be set equal to the ITSS_SensorId. That is, a sensor which communicates directly with a telematics application not using any kind of telematics device between is considered a telematics device in itself and must have an ITSS_TelematicsDeviceID equal to its ITSS_SensorId.

The request returns the values for all sensors connected or associated with the given ITSS_TelematicsDeviceID.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/SensorValuesTimeInterval?ITSS_TransportDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/SensorValuesTimeInterval?ITSS_TelematicsDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json
BODY: json Schema	{ “title”: “sensorValuesTimeInterval”, "$schema": "http://json-schema.org/draft-04/schema#", “type”: “object”, “properties”: { "ITSS_TransportDeviceID": { "type": "string", required: false },

	"ITSS_TelematicsDeviceID": { "type": "string", required: true },
	“ITSS_SensorValueList” : { "type": "array", required: true },
	[{
	“ITSS_SensorValue”: { "type": "object", required: false }
	“properties”:
	{
	"SamplingUTCTimestamp": { "type": "number", required: true },
	"ITSS_SensorId": { "type": "string", required: true },
	"Value": { "type": "float", required: true },
	"ITSS_SensorType": { "type": "string", required: false },
	"ITSS_SensorPosition": { "type": "string", required: false }
	}
	}]
	“ITSS_TelematicsApplicationID”: { "type": "string", required: true }
	}
	}

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“ITSS_SensorValueList”:[{
	"SamplingUTCTimestamp ": 1436712345.154,
	"ITSS_SensorId": "MANUF000001234500010",
	"Value": "48.87",
	"ITSS_SensorType": "temperature",
	"ITSS_SensorPosition": "axleBearingL1
	}, {
	"SamplingUTCTimestamp": 1436722345.154,
	"ITSS_SensorId": "MANUF000001234500123",
	"Value": "175000.0",
	"ITSS_SensorType": "pressure",
	"ITSS_SensorPosition": "tank"
	}],
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

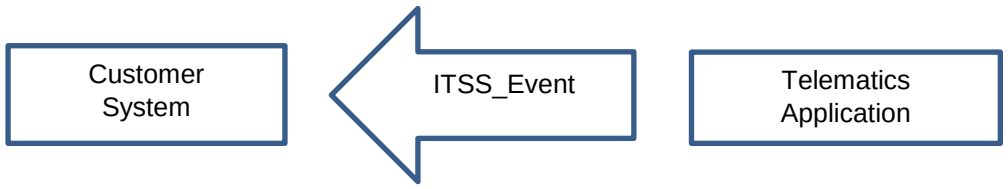
Notification about one or more new sensor values

1. Description

The customer expects to be notified if a sensor reports new data. The sensor data is provided only for diagnostic purposes. At no time, it shall be used to derive safety related information or actions.

2. Method (Event based notification)

The telematics application sends an event notification containing one or more ITSS_SensorValue when new sensor data is available.



Access method: Event message

sensorValues

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- ITSS_SensorValueList [
  - ITSS_SensorValue]
- ITSS_TelematicsApplicationID

The transmitted ITSS_SensorValueList contains one or multiple entries of ITSS_SensorValues (see section Glossary: for details). The values are transmitted as they arrive in the telematics application, hence the order of the ITSS_SensorValue within an ITSS_SensorValueList may not be ordered according to their timestamps.

If a sensor sends data directly to the telematics application, e.g. by using some kind of gateway which is not a telematics device, the ITSS_TelematicsDeviceID must be set equal to the ITSS_SensorId. That is, a sensor which communicates directly with a telematics application not using any kind of telematics device between is considered a telematics device in itself and must have an ITSS_TelematicsDeviceID equal to its ITSS_SensorId.

JSON Schema and example for a notification related to an ITSS_TelematicsDeviceID

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/sensorValues
BODY: json Schema	{ "title": "sensorValues", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { "ITSS_TransportDeviceID": { "type": "string", required: false }, "ITSS_TelematicsDeviceID": { "type": "string", required: true }, "ITSS_SensorValueList" : { "type": "array", required: true }, [{ "ITSS_SensorValue": { "type": "object", required: false } "properties": { "SamplingUTCTimestamp": { "type": "number", required: true }, "ITSS_SensorId": { "type": "string", required: true }, "Value": { "type": "float", required: true }, "ITSS_SensorType": { "type": "string", required: true }, "ITSS_SensorPosition": { "type": "string", required: true } } }] "ITSS_TelematicsApplicationID": { "type": "string", required: true } } }
BODY example	{ "ITSS_TransportDeviceID": "3180 4674 001-1", "ITSS_TelematicsDeviceID": "MANUF000000751", "ITSS_SensorValueList":[{ "SamplingUTCTimestamp ": 1436712345.154, "ITSS_SensorId": "MANUF000001234500010", "Value": "48.87", "ITSS_SensorType": "temperature", "ITSS_SensorPosition": "axleBearingL1 }, { "SamplingUTCTimestamp": 1436722345.154, "ITSS_SensorId": "MANUF000001234500123", "Value": "175000.0", "ITSS_SensorType": "pressure", "ITSS_SensorPosition": "tank" }], "ITSS_TelematicsApplicationID": "TeleApp0815" }
Response on success	
HTTP Status	201
MIME Type	Text plain

Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Request last known geofencing state

1. Description

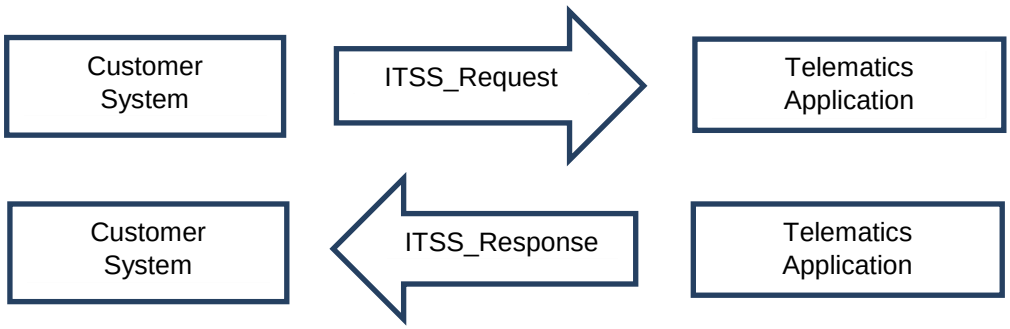
The customer wants to get information about the last known geofencing state of a specific transport device. The geofencing state is a list of all geofences the device is currently located within.

2. Method (Request / Response)

The customer system requests the last known geofencing state of a transport device identified by the

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID

The telematics application processes the request and responds with the required information.



Access method: Synchronous

lastGeofencingState

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_CustomerSystemID

lastGeofencingState response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- UTCtimestamp
- ITSS_GeofenceList
- GNSS_Position (optional)
- ITSS_TelematicsApplicationID

If the device is currently not located within a geofence an empty ITSS_GeofenceList is returned.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/lastGeofencingState?ITSS_TransportDeviceID={deviceId}& ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/lastGeofencingState?ITSS_TelematicsDeviceID={deviceId}& ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json

BODY:
json Schema
	{
	"title": "lastGeofencingState",
	"$schema": "http://json-schema.org/draft-04/schema#",
	"type": "object",
	"properties":
	{
	"ITSS_TransportDeviceID": { "type": "string", required: false },
	"ITSS_TelematicsDeviceID": { "type": "string", required: true },
	"UTCtimestamp": { "type": number", required: true },
	"ITSS_GeofenceList":
	[{
	"type": array, required: true,
	"items"
	{
	"type": "object", required: false,
	"properties":
	{
	"GeofenceID": { "type": "string", required: true },
	"GeofenceName": { "type": "string", required: false }
	}
	}
	}]
	"GNSS_Position" :
	{
	"type": "object", required: false,
	"properties":
	{
	"GNSS_UTCtimestamp": { "type": "number", required: true },
	"GNSS_Latitude": { "type": "number", required: true },
	"GNSS_Longitude": { "type": "number", required: true },
	"GNSS_Speed_kmph": { "type": "number", required: false },
	"GNSS_Heading_deg": { "type": "number", required: false },
	"ITSS_LocationInfo":
	{
	"type": "object", required: false,
	"properties":
	{
	"Location_ZIP": { "type": "string", required: false },
	"Location_City": { "type": "string", required: false },
	"Location_Street": { "type": "string", required: false },
	"Location_Description": { "type": "string", required: false },
	"Location_Country": { "type": "string", required: false },

	"Location_UIC_Code": { "type": "string", required: false }, "Location_GeoZone": { "type": "string", required: false }, "Location_POI_ID" : { "type": "string", required: false } } } } }, "ITSS_TelematicsApplicationID": { "type": "string", required: true } } }

BODY example	{
	"ITSS_TransportDeviceID": "3180 4674 001-1",
	"ITSS_TelematicsDeviceID": "MANUF000000751",
	"UTCtimestamp": 1436712345.15,
	"ITSS_GeofenceList": [
	{
	"GeofenceID": "123456789",
	"GeofenceName": "Very important location"
	},
	{
	"GeofenceID": "99999999",
	"GeofenceName": "Another important location"
	}
	],
	"GNSS_Position":
	{
	"GNSS_UTCtimestamp": 1436790123.154,
	"GNSS_Latitude": 52.264304,
	"GNSS_Longitude": 10.525537,
	"GNSS_Speed_kmph": 48.87,
	"GNSS_Heading_deg": 350.1,
	"ITSS_LocationInfo":
	{
	"Location_ZIP": "38126",
	"Location_City": "Braunschweig",
	"Location_Street": "Berliner Platz",
	"Location_Description": "Braunschweig Hbf",
	"Location_Country": "Germany",
	"Location_UIC_Code": "051",
	"Location_GeoZone": "DE"
	}
	},
	"ITSS_TelematicsApplicationID": "TeleApp0815"
	}

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Notification about a geofence event

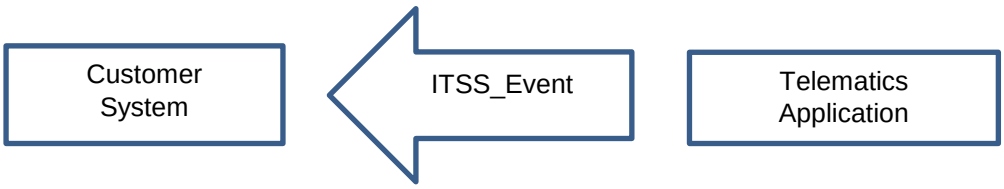
1. Description

The customer expects to be notified of a new geofence event of a transport device.

A geofence event occurs either when a device enters (ITSS_GeofenceEventTrigger = "enter") or leaves a geofence (ITSS_GeofenceEventTrigger = "exit").

2. Method (Event based notification)

The telematics application sends an event notification containing the geofencing event.



Access method: Event message

geofenceState

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- UTCtimestamp: the time when this event has been evaluated
- ITSS_GeoFence
- ITSS_GeofenceEventTrigger: "enter" or "exit"
- GNSS_Position (optional): position that has been used to evaluate the geofence state
- ITSS_TelematicsApplicationID

The UTCtimestamp defines the time when the reported geofence event was evaluated. Note that this timestamp may not report the exact time when the border of the geofence was actually crossed by the transport device. It may also differ from the GNSS_UTCtimestamp of an optionally included GNSS_Position which only states when this GNSS position has been acquired.

If the geofence evaluation generates events for more than one geofence, e.g. one "exit" event and one "enter" event, every event is communicated separately.

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/geofenceState

BODY: json Schema	<pre>{
	"title": "geofenceState",
	"$schema": "http://json-schema.org/draft-04/schema#",
	"type": "object",
	"properties":
	{
	"ITSS_TransportDeviceID": { "type": "string", required: false },
	"ITSS_TelematicsDeviceID": { "type": "string", required: true },
	"UTCtimestamp": { "type": "number", required: true },
	"ITSS_Geofence"
	{
	"type": "object", required: true,
	"properties":
	{
	"GeofenceID": { "type": "string", required: true },
	"GeofenceName": { "type": "string", required: false },
	}
	}
	"ITSS_GeofenceEventTrigger": { "type": "string", required: true },
	"GNSS_Position":
	{
	"type": "object", required: false,
	"properties":
	{
	"GNSS_UTCtimestamp ": { "type": "number", required: true },
	"GNSS_Latitude": { "type": "number", required: true },
	"GNSS_Longitude ": { "type": "number", required: true },
	"GNSS_Speed_kmph": { "type": "number", required: false },
	"GNSS_Heading_deg": { "type": "number", required: false },
	"ITSS_LocationInfo":
	{
	"type": "object", required: false,
	"properties":
	{
	"Location_ZIP": { "type": "string", required: false },
	"Location_City": { "type": "string", required: false },
	"Location_Street": { "type": "string", required: false },
	"Location_Description": { "type": "string", required: false },
	"Location_Country": { "type": "string", required: false },
	"Location_UIC_Code": { "type": "string", required: false },
	"Location_GeoZone": { "type": "string", required: false },
	"Location_POI_ID": { "type": "string", required: false }
	}
	}
	}
	},
	"ITSS_TelematicsApplicationID": { "type": "string", required: true }
	}
	}</pre>

BODY example	<pre>{
	"ITSS_TransportDeviceID": "3180 4674 001-1",
	"ITSS_TelematicsDeviceID": "MANUF000000751",
	"UTCtimestamp": 1436712339.124,
	"ITSS_Geofence"
	{
	"GeofenceID": "123456789",
	"GeofenceName": "Very important place"
	}
	"ITSS_GeofenceEventTrigger": "enter",
	"GNSS_Position":
	{
	"GNSS_UTCtimestamp": 1436712345.154,
	"GNSS_Latitude": 52.264304,
	"GNSS_Longitude": 10.525537,
	"GNSS_Speed_kmph": 48.87,
	"GNSS_Heading_deg": 350.1,
	"ITSS_LocationInfo":
	{
	"Location_ZIP": "38126",
	"Location_City": "Braunschweig",
	"Location_Street": "Berliner Platz",
	"Location_Description": "Braunschweig Hbf",
	"Location_Country": "Germany",
	"Location_UIC_Code": "051",
	"Location_GeoZone": "DE"
	}
	},
	"ITSS_TelematicsApplicationID": "TeleApp0815"
	}</pre>

Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Request geofence events for a time interval

1. Description

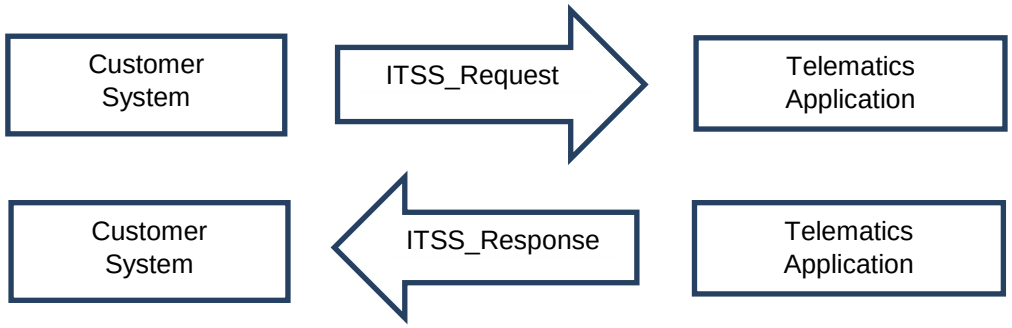
The customer requests information about the geofence events of one transport device in a given time interval (e.g. transportation trace).

2. Method (Request / Response)

The customer system requests a list of geofence events in the given time interval for one transport device identified by

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID

The telematics application responds with the requested list of geofence events. The GeofenceEventList might be empty, if no geofence event is contained in the specified time interval.



Access method: Synchronous

geofenceEventsTimeInterval

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- From_UTCtimestamp: UTCtimestamp – interval begin date (inclusive)
- To_UTCtimestamp: UTCtimestamp interval end date (exclusive)
- ITSS_CustomerSystemID

geofenceEventsTimeInterval response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_GeofenceEventList
- ITSS_TelematicsApplicationID

The returned ITSS_GeofenceEventList is ordered starting at the From_UTCtimestamp (first array entry) to the To_UTCtimestamp (last array entry). The timestamps returned can differ from

the requested interval such that the first array entry might start after the requested start time (From_UTCtimestamp) and the last array entry might stop before the requested stop time (To_UTCtimestamp). The telematics application uses the data available from the transport device and as such might not have data at all for the requested time interval.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceEventsTimeInterval?ITSS_TransportDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/geofenceEventsTimeInterval?ITSS_TelematicsDeviceID={deviceId}&From_UTCtimestamp={timeStamp}&To_UTCtimestamp={timeStamp}&ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json
BODY: json Schema	{ "title": "geofenceEventsTimeInterval", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { "ITSS_TransportDeviceID": { "type": "string", required: false }, "ITSS_TelematicsDeviceID": { "type": "string", required: true }, "ITSS_GeofenceEventList": [{ "type": "array", required: true, "items": { "type": "object", required: false "properties": { "UTCtimestamp": { "type": "number", required: true }, "ITSS_Geofence" { "type": "object", required: true, "properties": { "GeofenceID": { "type": "string", required: true }, "GeofenceName": { "type": "string", required: false } } } "ITSS_GeofenceEventTrigger": { "type": "string", required: true } "GNSS_Position" : { "type": "object", required: false "properties": {

<table>
<tr><td></td><td>
<pre>
"GNSS_UTCtimestamp": { "type": "number", required: true },
"GNSS_Latitude": { "type": "number", required: true },
"GNSS_Longitude": { "type": "number", required: true },
"GNSS_Speed_kmph": { "type": "number", required: false },
"GNSS_Heading_deg": { "type": "number", required: false },
"ITSS_LocationInfo":
{
    "type": "object", required: false,
    "properties":
    {
        "Location_ZIP": { "type": "string", required: false },
        "Location_City": { "type": "string", required: false },
        "Location_Street": { "type": "string", required: false },
        "Location_Description": { "type": "string", required: false },
        "Location_Country": { "type": "string", required: false },
        "Location_UIC_Code": { "type": "string", required: false },
        "Location_GeoZone": { "type": "string", required: false },
        "Location_POI_ID" : { "type": "string", required: false }
    }
}
}
}
}
}
}]
"ITSS_TelematicsApplicationID": { "type": "string", required: true }
}
}
</pre>
</td></tr>
<tr><td>BODY example</td><td>
<pre>
{
"ITSS_TransportDeviceID": "3180 4674 001-1",
"ITSS_TelematicsDeviceID": "MANUF000000751",
"ITSS_GeofenceEventList":
[{
"UTCtimestamp": 1436712339.124,
"ITSS_Geofence":
{
"GeofenceID": "123456789",
"GeofenceName": "Very important place"
},
"ITSS_GeofenceEventTrigger": "enter"
},
{
"UTCtimestamp": 14367123654.765,
"ITSS_Geofence":
{
"GeofenceID": "123456789",
"GeofenceName": "Very important place"
},
"ITSS_GeofenceEventTrigger": "exit"
}],
"ITSS_TelematicsApplicationID": "TeleApp0815"
}
</pre>
</td></tr>
<tr><td colspan="2">Response on empty list</td></tr>
<tr><td>HTTP Status</td><td>200</td></tr>
<tr><td>MIME Type</td><td>application/json</td></tr>
<tr><td>BODY:</td><td>{ "ITSS_GeofenceEventList" : [] }</td></tr>
</table>

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Request the movement state

1. Description of the use case

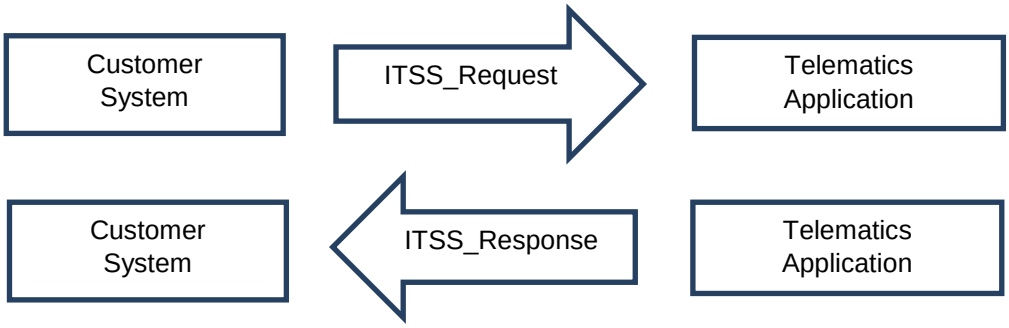
The customer wants to know the last known movement state of a transport device.

2. Method (Request / Response)

The customer system requests the last known movement state of a transport device related to the given

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID.

The telematics application responds with the last known movement state of the specific transport device.



Access Method: Synchronous

movementState

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_CustomerSystemID

movementState response

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- UTCtimestamp
- GNSS_Position (optional)
- ITSS_MovementState: "moving" / "standing" / "parking" / "unknown"
- ITSS_TelematicsApplicationID

The UTCtimestamp defines the time when the reported movement state was evaluated. It can differ from the GNSS_UTCtimestamp as the GNSS_UTCtimestamp states the time of the GNSS position acquisition. This specification does not define how the movement state is evaluated by the telematics application or telematics device.

JSON schema and example:

Request	
HTTP Type	GET
MIME Type	Text plain
Request Path	https://telematik.xyz.com/itss/1.2/movementState?ITSS_TransportDeviceID={deviceId}&ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/movementState?ITSS_TelematicsDeviceID={deviceId}&ITSS_CustomerSystemID={custId}
Response on success	
HTTP Status	200
MIME Type	application/json

BODY: json Schema	<pre>{
	"title": "movementState",
	"$schema": "http://json-schema.org/draft-04/schema#",
	"type": "object",
	"properties":
	{
	"ITSS_TransportDeviceID": { "type": "string", required: false },
	"ITSS_TelematicsDeviceID": { "type": "string", required: true },
	"UTCtimestamp": { "type": "number", required: true },
	"GNSS_Position":
	{
	"type": "object", required: false,
	"properties":
	{
	"GNSS_UTCtimestamp": { "type": "number", required: true },
	"GNSS_Latitude": { "type": "number", required: true },
	"GNSS_Longitude": { "type": "number", required: true },
	"GNSS_Speed_kmph": { "type": "number", required: false },
	"GNSS_Heading_deg": { "type": "number", required: false },
	"ITSS_LocationInfo":
	{
	"type": "object", required: false,
	"properties":
	{
	"Location_ZIP": { "type": "string", required: false },
	"Location_City": { "type": "string", required: false },
	"Location_Street": { "type": "string", required: false },
	"Location_Description": { "type": "string", required: false },
	"Location_Country": { "type": "string", required: false },
	"Location_UIC_Code": { "type": "string", required: false },
	"Location_GeoZone": { "type": "string", required: false },
	"Location_POI_ID" : { "type": "string", required: false }
	}
	}
	}
	},
	"ITSS_MovementState": { "type": "string", required: true },
	"ITSS_TelematicsApplicationID": { "type": "string", required: true }
	}
	}</pre>

BODY example	<pre>{
	"ITSS_TransportDeviceID": "3180 4674 001-1",</pre>

	<pre>"ITSS_TelematicsDeviceID": "MANUF000000751",
	"UTCtimestamp": 1436712339.124,
	"GNSS_Position":
	{
	"GNSS_UTCtimestamp": 1436712353.3,
	"GNSS_Latitude": 52.264304,
	"GNSS_Longitude": 10.525537,
	"GNSS_Speed_kmph": 48.87,
	"GNSS_Heading_deg": 350.1,
	"ITSS_LocationInfo":
	{
	"Location_ZIP": "38126",
	"Location_City": "Braunschweig",
	"Location_Street": "Berliner Platz",
	"Location_Description": "Braunschweig Hbf",
	"Location_Country": "Germany",
	"Location_UIC_Code": "051",
	"Location_GeoZone": "DE"
	}
	},
	"ITSS_MovementState": "moving",
	"ITSS_TelematicsApplicationID": "TeleApp0815"
	}</pre>

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

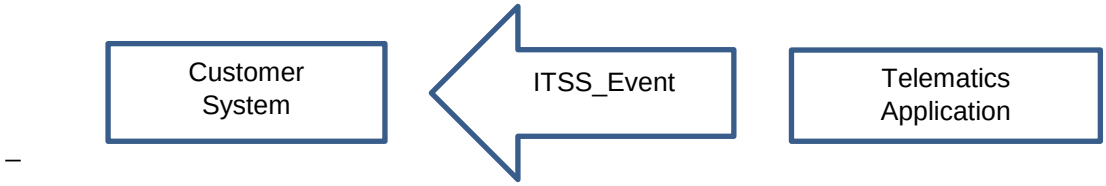
Notification about the movement state

1. Description

The customer expects to be notified of a new known movement state of a transport device.

2. Method (Event based notification)

The telematics application sends an event notification containing the new movement state.



Access method: Event message

movementState

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- UTCtimestamp
- GNSS_Position (optional)
- ITSS_MovementState: "moving" / "standing" / "parking" / "unknown"
- ITSS_TelematicsApplicationID

The UTCtimestamp defines the time when the reported movement state was evaluated. It can differ from the GNSS_UTCtimestamp as the GNSS_UTCtimestamp states the time of the GNSS position acquisition.

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/movementState
BODY: json Schema	{ "title": "movementState", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { "ITSS_TransportDeviceID": { "type": "string", required: false }, "ITSS_TelematicsDeviceID": { "type": "string", required: true }, "UTCtimestamp": { "type": "number", required: true }, "GNSS_Position":

	{
	"type": "object", required: false
	"properties":
	{
	"GNSS_UTCtimestamp": { "type": "number", required: true },
	"GNSS_Latitude": { "type": "number", required: true },
	"GNSS_Longitude": { "type": "number", required: true },
	"GNSS_Speed_kmph": { "type": "number", required: false },
	"GNSS_Heading_deg": { "type": "number", required: false },
	"ITSS_LocationInfo":
	{
	"type": "object", required: false,
	"properties":
	{
	"Location_ZIP": { "type": "string", required: false },
	"Location_City": { "type": "string", required: false },
	"Location_Street": { "type": "string", required: false },
	"Location_Description": { "type": "string", required: false },
	"Location_Country": { "type": "string", required: false },
	"Location_UIC_Code": { "type": "string", required: false },
	"Location_GeoZone": { "type": "string", required: false },
	"Location_POI_ID" : { "type": "string", required: false }
	}
	}
	}
	},
	"ITSS_MovementState": { "type": "string", required: true },
	"ITSS_TelematicsApplicationID": { "type": "string", required: true }
	}
	}

BODY example	{
	"ITSS_TransportDeviceID": "3180 4674 001-1",
	"ITSS_TelematicsDeviceID": "MANUF000000751",
	"UTCtimestamp": 1436712339.124,
	"GNSS_Position":
	{
	"GNSS_UTCtimestamp": 1436712345.154,
	"GNSS_Latitude": 52.264304,
	"GNSS_Longitude": 10.525537,
	"GNSS_Speed_kmph": 48.87,
	"GNSS_Heading_deg": 350.1,
	"ITSS_LocationInfo":
	{
	"Location_ZIP": "38126",
	"Location_City": "Braunschweig",
	"Location_Street": "Berliner Platz",
	"Location_Description": "Braunschweig Hbf",
	"Location_Country": "Germany",
	"Location_UIC_Code": "051",
	"Location_GeoZone": "DE"
	}
	},
	"ITSS_MovementState": "moving",
	"ITSS_TelematicsApplicationID": "TeleApp0815"
	}

Response on success	
HTTP Status	201
MIME Type	Text plain

Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Notification of a detected derailment

1. Description

The customer expects to be notified immediately, if a derailment has been detected by the telematics device.

This non-vital information is provided for diagnostic purposes only. At no time, it shall be used to derive safety related information or actions.

2. Method (Event based notification)

- The telematics application sends an event notification containing the timestamp when the derailment has been detected. If the GNSS_Position of that event is available it shall be included. If the transport device is in motion, the acquired GNSS_Position may differ from the position where the derailment has been actually detected.

Access method: Event message

derailmentDetected

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- UTCtimestamp – when derailment has been detected
- GNSS_Position (if available)
- ITSS_LocationInfo (optional)
- ITSS_TelematicsApplicationID

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json

Request Path	https://{**customerURI**}/itss/1.2/derailmentDetected
BODY: json Schema	{ "title": "derailmentDetected", "$schema": "http://json-schema.org/draft-04/schema#", "type":"object", "properties": { "ITSS_TransportDeviceID": { "type": "string", required:false }, "ITSS_TelematicsDeviceID": { "type": "string", required:true }, "UTCtimestamp": { "type": "number", required: true } "GNSS_Position" : { "type": "object", required: false "properties": { "GNSS_UTCtimestamp ": { "type": "number", required:true }, "GNSS_Latitude": { "type": "number", required:true }, "GNSS_Longitude ": { "type": "number", required:true }, "GNSS_Speed_kmph ": { "type": "number", required:false }, "GNSS_Heading_deg ": { "type": "number", required:false }, "GNSS_Altitude ": { "type": "number", required:false }, "GNSS_Accuracy ": { "type": "number", required:false }, "ITSS_LocationInfo": { "type": "object", required:false, "properties": { "Location_ZIP": { "type": "string", required: false }, "Location_City": { "type": "string", required: false }, "Location_Street": { "type": "string", required: false }, "Location_Description": { "type": "string", required: false }, "Location_Country": { "type": "string", required: false }, "Location_UIC_Code": { "type": "string", required: false }, "Location_GeoZone": { "type": "string", required: false } } } }, "ITSS_TelematicsApplicationID": { "type": "string", required:true } } }
BODY example	{ "ITSS_TransportDeviceID": "3180 4674 001-1", "ITSS_TelematicsDeviceID": "MANUF000000751", "UTCtimestamp": 1436712339.124, "GNSS_Position" : { "GNSS_UTCtimestamp": 1436712345.154, "GNSS_Latitude": 52.264304, "GNSS_Longitude": 10.525537, "GNSS_Speed_kmph": 48.87, "GNSS_Heading_deg": 350.1, "ITSS_LocationInfo": { "Location_ZIP": 38126, "Location_City": "Braunschweig", "Location_Street": "Berliner Platz", "Location_Description": "Braunschweig Hbf", "Location_Country": "Germany", "Location_UIC_Code": "051", "Location_GeoZone": "DE" }

	}, “ITSS_TelematicsApplicationID”: “TeleApp0815” }
Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Notification of a detected shock

1. Description

The customer expects to be informed about a shock that has been detected by the telematics device which is installed at the transport device. The detected shock occurred to the transport device as well as to the transported freight. As a shock above a certain level may result in damage to freight and / or transport device, the occurrence has to be signalled to the customer system.

For commercial reason, it is necessary to get the amplitude of the detected shock as well as the location where it occurred.

2. Method (Event based notification)

The telematics application sends an event notification containing the timestamp when the shock has been detected. If the GNSS_Position of that event is available it shall be included. If the transport device is in motion, the acquired GNSS_Position may differ from the position where the shock has been actually detected.

Access Method: Event message

shockDetected

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- UTCtimestamp
- GNSS_Position – after the shock has been detected (if available)
- ITSS_LocationInfo (optional)
- X-Axis_triggered: boolean (true, if axis triggered the shock detection)
- Y-Axis_triggered: boolean (true, if axis triggered the shock detection)
- Z-Axis_triggered: boolean (true, if axis triggered the shock detection)
- X-Axis: number in milli g (g = 9.81 m / s²)
- Y-Axis: number in milli g (g = 9.81 m / s²)
- Z-Axis: number in milli g (g = 9.81 m / s²)
- ITSS_TelematicsApplicationID

Remark: If the transport device is a railway wagon, then the reference for all axes is defined according to the wagon coordinate system in “Diagram 2: wagon coordinate system”

JSON schema and example:

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://{**customerURI**}/itss/1.2/shockDetected

BODY: json Schema	<pre>{
	“title”: “shockDetected”,
	"$schema": "http://json-schema.org/draft-04/schema#",
	“type”: “object”,
	“properties”:
	{
	“ITSS_TransportDeviceID”: { "type": "string", required:false },
	“ITSS_TelematicsDeviceID”: { "type": "string", required:true },
	“UTCtimestamp”: { “type”: “number”, required: true }
	“GNSS_Position” :
	{
	“type”: “object”, required: false
	"properties":
	{
	"GNSS_UTCtimestamp ": { "type": "number", required:true },
	”GNSS_Latitude": { "type": "number", required:true },
	"GNSS_Longitude ": { "type": "number", required:true },
	"GNSS_Speed_kmph ": { "type": "number", required:false },
	"GNSS_Heading_deg ": { "type": "number", required:false },
	“GNSS_Altitude “: { “type”: “number”, required:false },
	"GNSS_Accuracy ": { "type": "number", required:false },
	“ITSS_LocationInfo”: { "type": "object", required:false,
	"properties":
	{
	“Location_ZIP”: { "type": "string", required: false },
	“Location_City”: { "type": "string", required: false },
	“Location_Street”: { "type": "string", required: false },
	“Location_Description”: { "type": "string", required: false },
	“Location_Country”: { "type": "string", required: false },
	“Location_UIC_Code”: { "type": "string", required: false },
	“Location_GeoZone”: { "type": "string", required: false }
	}
	}
	},
	“X-Axis_triggered”: { "type": "boolean", required: true }
	“Y-Axis_triggered”: { "type": "boolean", required: true }
	“Z-Axis_triggered”: { "type": "boolean", required: true }
	“X-Axis”: { "type": "number", required: true }
	“Y-Axis”: { "type": "number", required: true }
	“Z-Axis”: { "type": "number", required: true }
	“ITSS_TelematicsApplicationID”: { "type": "string", required:true }
	}
	}</pre>

BODY example	{
	“ITSS_TransportDeviceID”: “3180 4674 001-1”,
	“ITSS_TelematicsDeviceID”: “MANUF000000751”,
	“UTCtimestamp”: 1436712339.124,
	“GNSS_Position” :
	{
	“GNSS_UTCtimestamp”: 1436712345.154,
	“GNSS_Latitude”: 52.264304,
	“GNSS_Longitude”: 10.525537,
	“GNSS_Speed_kmph”: 48.87,
	“GNSS_Heading_deg”: 350.1,
	“ITSS_LocationInfo”: {
	“Location_ZIP”: 38126,
	“Location_City”: “Braunschweig”,
	“Location_Street“: “Berliner Platz”,
	“Location_Description": “Braunschweig Hbf”,
	“Location_Country”: “Germany”,
	“Location_UIC_Code”: “051”,
	“Location_GeoZone”: “DE”
	}
	},
	“X-Axis_triggered”: true,
	“Y-Axis_triggered”: false,
	“Z-Axis_triggered”: false,
	“X-Axis”: 2600,
	“Y-Axis”: 100,
	“Z-Axis”: 100,
	“ITSS_TelematicsApplicationID”: “TeleApp0815”
	}

Response on success	
HTTP Status	201
MIME Type	Text plain
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Create

1. Description of the use case

The customer system wants to create new geofences in the telematics application geofence repository for an ITSS_CustomerSystemID.

2. Method (Request)

The customer system requests to create new geofences. These geofences will be added to potentially existing geofences. There will be no implicit delete of existing geofences.

- Each geofence has an identifier that is unique among all geofences of a customer and a name (can be used for displaying). The attempt of creating an existing geofence will return an error. Every error leads to a rejecting of the whole list; hence no geofence will be created.

  The list contains one or more elements. If the list is empty, GeofenceCreate returns an error.

  This specification defines three types of geofences: circles, polygons and lines.

  1. Circles are defined by a point specified by longitude and latitude and a radius in meters. Positions with a distance to the given point below this radius are regarded as inside the geofence.
  2. Polygons are defined by an array of three or more (up to 32) points (latitude and longitude). Polygons need not be convex but may also be concave, but they must not be overturned. Positions within the polygon are regarded as inside the geofence.
  3. Lines are defined by an array of two or more points (latitude and longitude) and the maximum normal distance from this line. All positions with a normal distance to that line below this threshold are regarded as inside the geofence.

The geofence functions “circles” and “polygons” must be supported by the telematics system. The geofence function “lines” can be supported by the telematics system.

The following definitions are generally used within ITSS geofence methods:

```
"definitions":
{
    "ITSS_GeoPoint":
    {
        "type": "object",
        "properties":
        {
            "Longitude": { "type": "number" },
            "Latitude": { "type": "number" }
        },
        "required": [ "Latitude", "Longitude" ]
    },
    "ITSS_GeoLine":
    {
        "type": "object",
        "properties":
        {
            "Line":
            {
                "type": "array",
                "minItems": 2,
                "items": { "$ref": "#/definitions/ITSS_GeoPoint" }
            },
            "Distance": { "type": "number" }
        },
        "required": [ "Line", "Distance" ]
    },
    "ITSS_GeoCircle":
    {
        "type": "object",
        "properties":
        {
            "Center": { "$ref": "#/definitions/ITSS_GeoPoint" },
            "Radius": { "type": "number" }
        },
        "required": [ "Center", "Radius" ]
    },
    "ITSS_GeoPolygon":
    {
        "type": "object",
        "properties":
        {
            "Polygon":
            {
                "type": "array",
                "minItems": 3,
                "items": { "$ref": "#/definitions/ITSS_GeoPoint" }
            }
        },
        "required": [ "Polygon" ]
    }
}
```

```
"definitions":
{
    "DeviceID":
    {
        "oneOf":
        [
```

```
                    { "ITSS_TransportDeviceID": { "type": "string" },
                    { "ITSS_TelematicsDeviceID": { "type": "string" }
                ],
                "required": true
            },
            “ITSS_GeofenceIDList”:
            {
                "type": "array", required: true,
                "items":
                [{
                 "GeofenceID":                { "type": "string", required: true }

                }]

        }
    }
```

–

geofenceCreate

- ITSS_CustomerSystemID
- GeofenceList

JSON schema and example for a request

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://telematik.xyz.com/itss/1.2/geofenceCreate
BODY: json Schema	{ "title": "geofenceCreate", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { “ITSS_CustomerSystemID”: { "type": "string", required: true }, "GeofenceList": [{ "type": "array", "items": { "ITSS_GeofenceDef" { "type": "object", required: true, "properties": {

	"GeofenceID": { "type": "string", required: true },
	"GeofenceName": { "type": "string", required: false },
	"ITSS_GeofenceEvent": { "type": "string", "required": false }
	}
	},
	"GeoObject":
	{
	"anyOf":
	[
	{ "$ref": "#/definitions/ITSS_GeoCircle" },
	{ "$ref": "#/definitions/ITSS_GeoLine" },
	{ "$ref": "#/definitions/ITSS_GeoPolygon" }
	],
	"required": true
	}
	}
	}]
	}
	}

BODY:
example
	{
	“ITSS_CustomerSystemID”: ”CUSTSYS001”,
	"GeofenceList":
	[{
	“ITSS_GeofenceDef”:
	{
	"GeofenceID": "123456788",
	"GeofenceName": "Very important Circle",
	“ITSS_GeofenceEvent”: “on_both”
	},
	"GeoObject":
	{
	"Circle": { "Longitude": 16.300, "Latitude": 48.300 },
	"Radius": 200
	},
	}, {
	" ITSS_GeofenceDef ":
	{
	"GeofenceID": "123456799",
	"GeofenceName": "Important Circle",
	“ITSS_GeofenceEvent”: “on_enter”
	},
	“GeoObject":
	{
	"Circle": { "Longitude": 16.322, "Latitude": 49.300 },
	"Radius": 130
	}
	}]
	}

Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: empty	
Response on request error	
HTTP Status	400

MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Geofence Update

1. Description

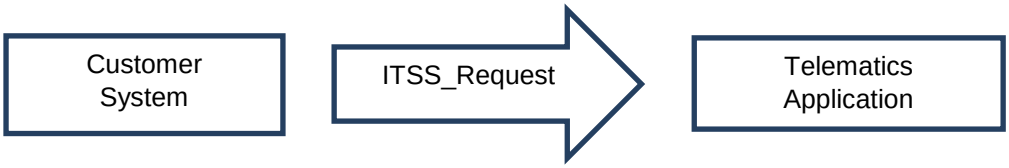
The customer system wants to update a list of already defined geofences in the telematics application geofence repository for an ITSS_CustomerSystemID.

2. Method (Request)

The customer system requests to update a list of existing definitions of geofences.

There will be no implicit creation of non-existing geofences.

Every geofence has an identifier that is unique among all geofences of a customer and is used to identify the geofence that has to be updated. The updating of not existing geofences will return an error. In case of an error the whole list will be rejected.



Access Method: Synchronous

geofenceUpdate

- ITSS_CustomerSystemID
- GeofenceList

JSON schema and example for a request

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://telematik.xyz.com/itss/1.2/geofenceUpdate
BODY: json Schema	{ "title": "geofenceUpdate", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { “ITSS_CustomerSystemID”: { "type": "string", required: true }, "GeofenceList": [{ "type": "array", "items": {

	"ITSS_GeofenceDef"
	{
	"type": "object", required: true,
	"properties":
	{
	"GeofenceID": { "type": "string", required: true },
	"GeofenceName": { "type": "string", required: false },
	"ITSS_GeofenceEvent": { "type": "string", "required": false }
	}
	},
	"GeoObject":
	{
	"anyOf":
	[
	{ "$ref": "#/definitions/ITSS_GeoCircle" },
	{ "$ref": "#/definitions/ITSS_GeoLine" },
	{ "$ref": "#/definitions/ITSS_GeoPolygon" }
	],
	"required": true
	}
	}
	}]
	}

BODY: example	{
	“ITSS_CustomerSystemID”: ”CUSTSYS001”,
	"GeofenceList":
	[{
	“ITSS_GeofenceDef”:
	{
	"GeofenceID": "123456788",
	"GeofenceName": "Very important Circle",
	“ITSS_GeofenceEvent”: “on_both”
	},
	"GeoObject":
	{
	"Circle": { "Longitude": 16.300, "Latitude": 48.300 },
	"Radius": 200
	},
	}, {
	" ITSS_GeofenceDef ":
	{
	"GeofenceID": "123456799",
	"GeofenceName": "Important Circle",
	“ITSS_GeofenceEvent”: “on_enter”
	},
	"GeoObject":
	{
	"Circle": { "Longitude": 16.322, "Latitude": 49.300 },
	"Radius": 130
	}
	}]
	}

Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: empty	
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Read

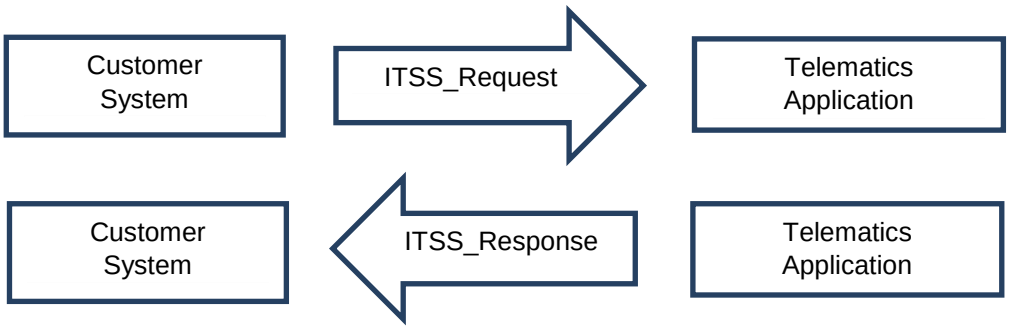
1. Description

The customer system wants to read the definition of a set of geofences for an ITSS_CustomerSystemID.

2. Method (Request)

The customer system requests the geographic definition of a list of geofences.

Every geofence has an identifier that is unique among all geofences of a customer.



Access Method: Synchronous

geofenceRead

- ITSS_CustomerSystemID
- ITSS_GeofenceIDList

geofenceReadResponse

- geofenceList

JSON schema and example for a request

Request	
HTTP Type	POST
MIME Type	text/plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceRead
BODY: json schema	{ "title": "geofenceRead", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { “ITSS_CustomerSystemID”: { "type": "string", required: true }, “ITSS_GeofenceIDList”: { "$ref": "#/definitions/ITSS_GeofenceIDList", "required": true } }
BODY: example	{ “ITSS_CustomerSystemID”: ”CUSTSYS001”, “ITSS_GeofenceIDList”: [{ "GeofenceID": “123456788” }, { “GeofenceID”: “123455558” }] }
Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: schema	{ "title": "geofenceRead", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { "GeofenceList": [{ "type": "array", "items": { "ITSS_GeofenceDef" { "type": "object", required: true,

	"properties":
	{
	"GeofenceID": { "type": "string", required: true },
	"GeofenceName": { "type": "string", required: false },
	"ITSS_GeofenceEvent": { "type": "string", "required": false }
	}
	},
	"GeoObject":
	{
	"anyOf":
	[
	{ "$ref": "#/definitions/ITSS_GeoCircle" },
	{ "$ref": "#/definitions/ITSS_GeoLine" },
	{ "$ref": "#/definitions/ITSS_GeoPolygon" }
	],
	"required": true
	}
	}
	}]
	}
	}

BODY: example	{
	"GeofenceList":
	[{
	“ITSS_GeofenceDef”:
	{
	"GeofenceID": "123456788",
	"GeofenceName": "Very important Circle",
	“ITSS_GeofenceEvent”: “on_both”
	},
	"GeoObject":
	{
	"Circle": { "Longitude": 16.300, "Latitude": 48.300 },
	"Radius": 200
	},
	" ITSS_GeofenceDef ":
	{
	"GeofenceID": "123455558",
	"GeofenceName": "Important Circle",
	“ITSS_GeofenceEvent”: “on_enter”
	},
	"GeoObject":
	{
	"Circle": { "Longitude": 16.322, "Latitude": 49.300 },
	"Radius": 130
	}
	}]
	}

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain

BODY:	{error description}

–

Geofence Delete

1. Description

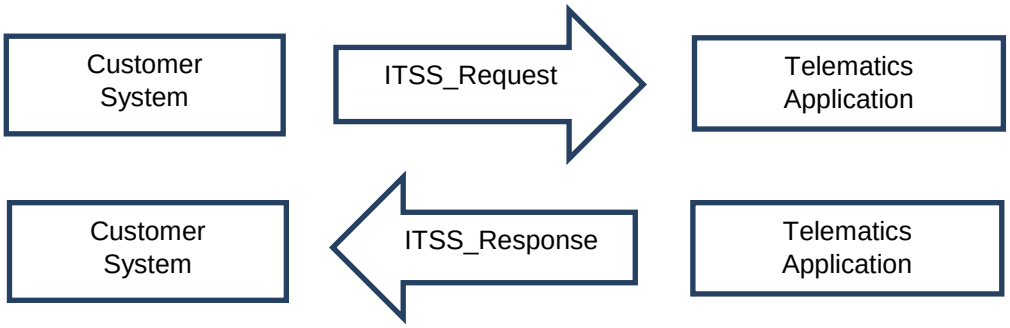
The customer system wants to delete a set of geofences from the telematics application geofence repository for an ITSS_CustomerSystemID.

Deleting a geofence will automatically delete all assignments to any telematic device of this geofence.

2. Method (Request)

The customer system requests deletion of a list of geofence definitions .

- Every geofence has an identifier that is unique among all geofences of a customer and is used to identify the geofences that have to be deleted. Non existing geofences will be ignored.



Access Method: Synchronous

geofenceDelete

- ITSS_CustomerSystemID
- ITSS_GeofenceIDList

JSON schema and example for a deletion request

Request	
HTTP Type	POST
MIME Type	text/plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceDelete
BODY: json schema	{ "title": "geofenceDelete", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object",

	"properties":
	{
	“ITSS_CustomerSystemID”: { "type": "string", required: true },
	“ITSS_GeofenceIDList”: { "$ref": "#/definitions/ITSS_GeofenceIDList",
	"required": true }
	}
	}

BODY:
example
	{
	“ITSS_CustomerSystemID”: ”CUSTSYS001”,
	“ITSS_GeofenceIDList”:
	[{
	"GeofenceID": “123456788”
	},
	{
	“GeofenceID”: “123455558”
	}]
	}

Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: empty	
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

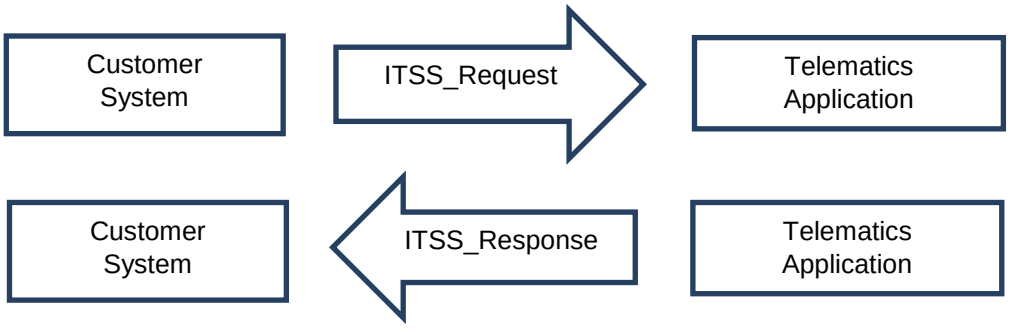
Geofence Read All

1. Description

The customer system wants to get the definition of all geofences of the ITSS_CustomerSystemID.

2. Method (Request)

The customer system requests all geofences.



Access Method: Synchronous

geofenceReadAll

- ITSS_CustomerSystemID

geofenceReadAllResponse

- geofenceList

JSON schema and example for a request

Request	
HTTP Type	GET
MIME Type	text/plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceReadAll?ITSS_CustomerSystemID={custId}
BODY: empty	
Response on success	
HTTP Status	200

MIME Type	Text plain
BODY: schema	{ "title": "geofenceReadAll", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { "GeofenceList": [{ "type": "array", "items": { "ITSS_GeofenceDef" { "type": "object", required: true, "properties": { "GeofenceID": { "type": "string", required: true }, "GeofenceName": { "type": "string", required: false }, "ITSS_GeofenceEvent": { "type": "string", "required": false } }, "GeoObject": { "anyOf": [{ "$ref": "#/definitions/ITSS_GeoCircle" }, { "$ref": "#/definitions/ITSS_GeoLine" }, { "$ref": "#/definitions/ITSS_GeoPolygon" }], "required": true } } }] } }
BODY: example	{ "GeofenceList": [{ “ITSS_GeofenceDef”: { "GeofenceID": "123456788", "GeofenceName": "Very important Circle", “ITSS_GeofenceEvent”: “on_both” }, "GeoObject": { "Circle": { "Longitude": 16.300, "Latitude": 48.300 }, "Radius": 200 }, }, { " ITSS_GeofenceDef ": { "GeofenceID": "123455558", "GeofenceName": "Important Circle", “ITSS_GeofenceEvent”: “on_enter” }, "GeoObject": { "Circle": { "Longitude": 16.322, "Latitude": 49.300 },

	"Radius": 130 } }] }
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Delete All

Description of the use case

The customer wants to delete all geofences for an ITSS_CustomerSystemID.

Method (Request)

The customer system requests to delete all geofences.

deleteGeofenceAll

- ITSS_CustomerSystemID

JSON schema and example for a request

Request	
HTTP Type	GET
MIME Type	text/plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceDeleteAll
BODY: empty	
Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: empty	
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Create Assignment

1. Description of the use case

The customer system wants to assign geofences from the telematics application geofence repository to a telematics device or transport device.

2. Method (Request)

The customer system requests to assign geofences to a single telematics device. These geofences will be added to potentially assigned geofences. There will be no implicit removal of assigned geofences.

- Each geofence has an identifier that is unique among all geofences of a customer. The attempt of assign a not existing geofence will return an error. Every error results in rejecting the whole list; hence no geofence will be assigned.

  The list contains one or more elements. If the list is empty, GeofenceCreateAssignment returns an error.

With successful assignment of the geofence, event notifications are automatically activated (if such an event is defined).

The following definitions are generally used within ITSS geofence methods:

geofenceCreateAssignment

- ITSS_CustomerSystemID
- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_GeofenceIDList

JSON schema and example for a request

Request	
HTTP Type	POST

MIME Type	application/json
Request Path	https://telematik.xyz.com/itss/1.2/geofenceCreateAssignment

BODY: json Schema	{
	"title": "geofenceCreateAssignment",
	"$schema": "http://json-schema.org/draft-04/schema#",
	"type": "object",
	"properties":
	{
	“ITSS_CustomerSystemID”: { "type": "string", required: true },
	"DeviceID": { "$ref": "#/definitions/DeviceID", "required": true },
	ITSS_GeofenceIDList”: { "$ref": "#/definitions/ITSS_GeofenceIDList",
	"required": true }
	}
	}

BODY: example	{
	“ITSS_CustomerSystemID”: ”CUSTSYS001”,
	“ITSS_TelematicsDeviceID”: ” MANUF000000751”,
	"ITSS_GeofenceIDList":
	[{
	"GeofenceID": "123456788"
	}, {
	"GeofenceID": "123456789"
	}]
	}

Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: empty	
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Read Assignments

1. Description

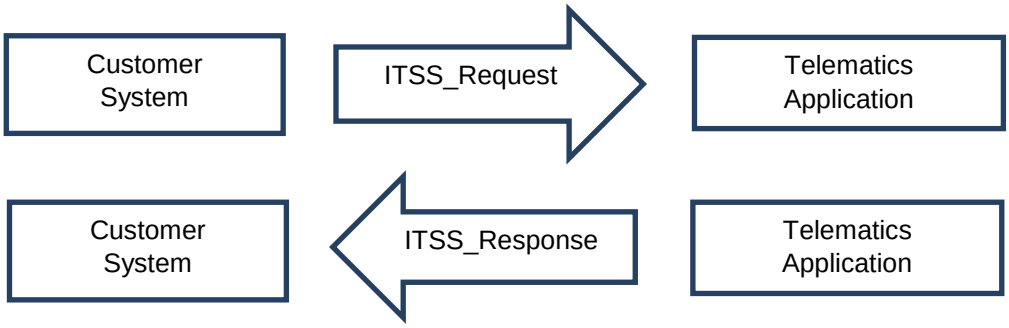
The customer system wants to read all the geofences assigned to a telematics device or transport device.

2. Method (Request)

The customer system requests the assignments of a list of geofences to a single telematics or transport device.

Every geofence has an identifier that is unique among all geofences of a customer.

–



Access Method: Synchronous

geofenceReadAssignment

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_CustomerSystemID

geofenceReadAssignmentResponse

- ITSS_TelematicsDeviceID
- ITSS_TransportDeviceID (optional)
- ITSS_GeofenceIDList

JSON schema and example for a request

Request	
HTTP Type	POST
MIME Type	text/plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceReadAssignment

BODY:
json schema
	{
	"title": "geofenceReadAssignment",
	"$schema": "http://json-schema.org/draft-04/schema#",
	"type": "object",
	"properties":
	{
	“ITSS_CustomerSystemID”: { "type": "string", required: true },
	"DeviceID": { "$ref": "#/definitions/DeviceID", "required": true }
	}
	}

BODY:
example
	{
	“ITSS_CustomerSystemID”: ”CUSTSYS001”,
	“ITSS_TelematicsDeviceID”: ” MANUF000000751”
	}

Response on success	
HTTP Status	200
MIME Type	Text plain

BODY:
schema
	{
	"title": "geofenceRead",
	"$schema": "http://json-schema.org/draft-04/schema#",
	"type": "object",
	"properties":
	{
	“ITSS_TransportDeviceID”: { "type": "string", required:false },
	“ITSS_TelematicsDeviceID”: { "type": "string", required:true },
	ITSS_GeofenceIDList”: { "$ref": "#/definitions/ITSS_GeofenceIDList",
	"required": true }
	}
	}

BODY:
example
	{
	“ITSS_TelematicsDeviceID”:” MANUF000000751”,
	"ITSS_GeofenceIDList":
	[{
	"GeofenceID": "123456788"
	}, {
	"GeofenceID": "123456789"
	}]
	}

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Delete Assignment

1. Description of the use case

The customer system wants to delete the assignment of one or more geofences to a telematics device or transport device.

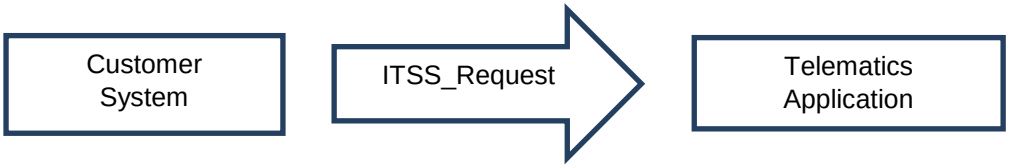
2. Method (Request)

The customer system requests to delete the assignment of geofences to a single telematics device. These geofences will be removed from the assignment.

- Each geofence has an identifier that is unique among all geofences of a customer. The attempt of unassigning a not existing geofence will return an error. Every error leads to a rejection of the whole list; hence no assignments will be deleted.

  The list contains one or more elements. If the list is empty, GeofenceDeleteAssignment returns an error.

  With successful removal from the assignment of the geofence, event notifications are automatically deactivated for this geofence on this device.



geofenceDeleteAssignment

- ITSS_CustomerSystemID
- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_GeofenceIDList

JSON schema and example for a request

Request	
HTTP Type	POST
MIME Type	application/json
Request Path	https://telematik.xyz.com/itss/1.2/geofenceDeleteAssignment
BODY: json Schema	{ "title": "geofenceDeleteAssignment", "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { “ITSS_CustomerSystemID”: { "type": "string", required: true }, "DeviceID": { "$ref": "#/definitions/DeviceID", "required": true }, ITSS_GeofenceIDList”: { "$ref": "#/definitions/ITSS_GeofenceIDList", "required": true } } }
BODY: example	{ “ITSS_CustomerSystemID”: ”CUSTSYS001”, “ITSS_TelematicsDeviceID”: ” MANUF000000751”, "ITSS_GeofenceIDList": [{ "GeofenceID": "123456788" }, { "GeofenceID": "123456789" }] }
Response on success	
HTTP Status	200
MIME Type	Text plain
BODY: empty	

Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

Geofence Delete All Assignments

Description of the use case

The customer wants to delete all assignments of geofences assigned to a telematics device or transport device.

Method (Request)

The customer system requests to delete the assignments of all geofences of a single device.

geofenceDeleteAllAssignments

- ITSS_TelematicsDeviceID or ITSS_TransportDeviceID
- ITSS_CustomerSystemID

JSON schema and example for a request

Request	
HTTP Type	GET
MIME Type	text/plain
Request Path	https://telematik.xyz.com/itss/1.2/geofenceDeleteAllAssingments?ITSS_TransportDeviceID={deviceId}&ITSS_CustomerSystemID={custId} or https://telematik.xyz.com/itss/1.2/geofenceDeleteAllAssingments?ITSS_TelematicsDeviceID={deviceId}&ITSS_CustomerSystemID={custId}
BODY: empty	
Response on success	
HTTP Status	200
MIME Type	Text plain

BODY: empty	
Response on request error	
HTTP Status	400
MIME Type	application/json
BODY:	JSON formatted error description see section Error concept
Response on undefined error	
HTTP Status	All other HTTP Status codes
MIME Type	text/plain
BODY:	{error description}

–

Webservice Technology

The REST specification provides a standard way for web clients to communicate with servers through REpresentational State Transfer (REST) technology.

REST supports optional parameter (unlike SOAP). It is easy to implement and widespread.

Optional parameters that are not supplied by the system are simply left out in the response.

The client issues text-based JSON requests to the server through structured URLs.

The server responds with a text-based reply in JSON format that complies to the request.

- Both communication parties must provide a web service that complies with the specification.

Webservice methods and invocation

The following section describes the web service methods provided by the interface between the telematics application and the customer system.

- The interface uses JSON over HTTP for data communication (see http://www.json.org).
- All web service methods in the interface are callable (according to REST) via an URI plus method name and additional parameters
  http://telematik.xyz.com/itss/<x>.<y>/<request>?<key>=<value>&<key>=<value>
- The following HTTP headers are mandatory based on the method:
  - For GET requests: none
  - For POST requests: Content-Type: application/json
- If invocation of the web service is successful the following responses can be expected based on the method:
  - For GET requests: HTTP status 200 is returned. The response body contains the expected JSON object as described above.
  - For POST requests: HTTP status 201 is returned. The response body will be empty and can be ignored.
- If invocation of the web service is not successful but the web service is able to respond, then status 400 is returned. The response body contains a JSON object with an detailed error description according to the section Error concept.
- If the web service does not respond or if it returns status code 503 (“Service unavailable”) the client should retry after a waiting period (e.g. ten minutes).

Data security

The interface between the telematics application and the customer system supports just one access right. Restrictions on objects and data (the role and right management) have to be implemented solely in the customer's system.

In order to protect against not authorized access, every message between the communication parties has to be authenticated by using system-ID and HTTP basic authentication.

In order to protect the telematics application against wrong accesses of valid customers, the system ID is the unique identifier of a specific customer system.

- To prevent third parties from eavesdropping, only SSL/TLS secured communication will be allowed.

Error concept

This chapter describes the error handling and returned error codes and messages. All errors and messages are generated by the telematics application.

As multiple fields of data are included in one API request a HTTP response code is not sufficient anymore to clearly represent the status of the request. Therefore, the error concept knows two levels of errors. The first level is HTTP based and addresses general errors like 'Server Down', or 'Resource Not Found' via the standard HTTP error codes. Those codes are described for each API request directly at the request description.

The second error level handles problems in the context of a request itself, that is, the request is received by the telematics application but cannot be executed due to for example parameter errors. This condition is generally indicated by an HTTP error code of 403 as described in the API request descriptions.

If the parameter validation of one parameter fails, the telematics application continues to process all other parameters before an error is returned. For each error an error code and an error description is returned. The request is not executed and no data from the request is stored by the telematics application.

All errors are returned with a single general error code and include a detailed error code that provides additional information, e.g. a field breakdown. This additional error field provides detailed error information for every failed parameter:

Important hint: all error responses MUST contain detailed error information with one exception - in case of an authentication failure (error code 0100) the detailed information should not be provided.

The details of the error are specified in a JSON object, see below:

Possible Error	JSON response format

General error	<pre>{
	  "code" : 1234,
	  "message" : "Something bad happened"
	}</pre>

Detailed error	<pre>{
	  "code" : 101,
	  "message" : "Parameter validation failed",
	  "errors" : [
	  {
	  "code" : 5432,
	  "message" : "The format of the id is wrong"
	  },
	  {
	  "code" : 5622,
	  "message" : "The requested time is unknown"
	  }
	  ]
	}</pre>

The following section lists all possible **general error codes**:

Error code	Message	Description
0100	Authentication failed.	The customer system could not be authenticated by the telematics application. The errors array **might** contain additional information. **Hint:** although the standard allows adding additional information to this error message consider the security problems raised by returning detailed authorization failure information.
0101	Parameter validation failed.	The parameter check for one or many parameters of the request failed. The errors array **must** contain additional information.

The following section lists all possible **detailed error codes**:

Error code	Message	Description
1001	The given system ID could not be found	The telematics application does not know the system ID provided.
1002	The system ID is missing	The customer system did not provide a system ID.
2001	The ITSS_TelematicsDeviceID could not be found	The requested telematics device is not known to the telematics application.
2002	The ITSS_TelematicsDeviceID format is not specification compliant	The format of the provided telematics device ID does not follow the rules of the specification.
2003	The ITSS_TransportDeviceID could not be found	The requested transport device is not known to the telematics application.
2004	The ITSS_TransportDeviceID format is not specification compliant	The format of the provided transport device ID does not follow the rules of the specification.
2005	The ITSS_TelematicsDeviceID and the ITSS_TransportDeviceID are not paired.	In case the customer system provides both IDs within a request and the telematics application knows the device pairing, the pairing is checked and an error returned if the pairing is not identical.
2006	The From_UTCtimestamp is invalid.	Either the format or the time stated by the time stamp is incorrect, e.g. time stamp smaller 0.
2007	The To_UTCtimestamp is invalid.	Either the format or the time stated by the time stamp is incorrect, e.g. time stamp smaller 0.
2008	The time interval between From_UTCtimestamp and To_UTCtimestamp is invalid.	The time interval between the two timestamps cannot be applied by the telematics application, e.g. is negative, that is, the To_UTCtimestamp is smaller than the From_UTCtimestamp.
2009	The parameter ITSS_TelematicsDeviceID is missing.	The customer system did not provide an ITSS_TelematicsDeviceID.
2010	The parameter ITSS_TransportDeviceID is missing.	The customer system did not provide an ITSS_TransportDeviceID.
2011	The parameter From_UTCtimestamp is missing.	The customer system did not provide a From_UTCtimestamp.

2012	The parameter To_UTCtimestamp is missing.	The customer system did not provide a To_UTCtimestamp.
2013	One of the parameters is not supported.	The customer system sent a parameter which is not supported by the telematics application.
3001	The response contains too much data and cannot be processed by the telematics application.	The request generated so much data that the telematics application reaches internal capacity limits.

–

Abbreviations

The following table contains the descriptions of the abbreviations used in this document

Abbreviation	Description
DIUM	DIUM - Uniform distance table for international freight traffic: List of railway stations - List of handover/delivery points used by the railways
GNSS	Global Navigation Satellite System; e.g.: GPS-NAVSTAR; GLONASS, Galileo, BeiDou …
ITSS	ITSS stands for „Industrieplattform Telematik und Sensorik im Schienengüterverkehr“; which describes a practice group defining standards related to telematics and sensorics in railway business
TIS	TIS stands for “Technischer Innovationskreis Schienengüterverkehr“; a group discussing and creating innovations related to railway business
TLS	Transport Layer Security; successor of SSL, used to provide communications security.
UTC	Universal time coordinated; is often used as time / date reference system in computer systems.
VPN	Virtual Private Network; is used to securely extend a private network across the internet. Remote computers (and services) appear as if they were part of the private network.
WGS 84	World Geodetic System as of 1984; is often used as reference system by GNSS systems
API	Application Programming Interface, a software interface to access functions across system/module boundaries.

Glossary:

The following table contains the terms used in this document

Term	Description
ITSS_ManufacturerID	ID of manufacturer as provided by the ITSS practice group upon request. 5 digits of printable ASCII codes from ‘0’ to ‘9’ and from ‘A’ to ‘Z’ The ITSS_ManufacturerID is defined by the ITSS practice group.
ITSS_TelematicsDeviceID	20 printable ASCII codes from ‘0’ to ‘9’ and from ‘A’ to ‘Z’ built from ITSS_ManufacturerID (5 digits) concatenated with manufacturer specific device ID (15 digits)
ITSS_TransportDeviceID	• UIC wagon number • Wheelset ID • GIAI number • UIC Traction Unit (Locomotives, multiple units, …) number • Vehicle ID • Container ISO number • Container NonISO (Swap Bodies, ULDs, …) ID • Generic Object ID An ITSS_TransportDeviceID is limited to 50 printable ASCII codes in the range of hexadecimal 0x21 to 0x7E with the exception of hexadecimal 0x2F, 0x5C, 0x3A, 0x3F, 0x22, 0x3C, 0x3E which is printable / \ : * ? “ < > |.
ITSS_LocationInfo	Location_ZIP (optional): number Location_City (optional): string Location_Street (optional): string Location_Description (optional): string Location_Country (optional): string Location_UIC_Code (optional): string Location_GeoZone (optional): string
Wagon coordinate system	X-axis of railway wagon parallel to the track Y-axis of railway wagon perpendicular to the track Z-axis of railway wagon vertical to the track see “Diagram 2: wagon coordinate system”
ITSS_CustomerSystemID	The unique ID of a customer system used in any request to a telematics application. This ID is validated by the telematics application to guard against

	unauthorized access. See System Architecture Overview for details on management and generation of an ITSS_CustomerID. An ITSS_CustomerID is limited to 20 printable ASCII codes in the range of hexadecimal 0x21 to 0x7E with the exception of hexadecimal 0x2F, 0x5C, 0x3A, 0x3F, 0x22, 0x3C, 0x3E which is printable / \ : * ? “ < > |. An ITSS_CustomerID must not be empty.
ITSS_TelematicsApplicationID	The unique ID of a telematics application used in any response and event delivered to a customer system. This ID together is validated by the customer system to guard against unauthorized data reception. See System Architecture Overview for details on management and generation of an ITSS_TelematicsApplicationID An ITSS_TelematicsApplicationID is limited to 20 printable ASCII codes in the range of hexadecimal 0x21 to 0x7E with the exception of hexadecimal 0x2F, 0x5C, 0x3A, 0x3F, 0x22, 0x3C, 0x3E which is printable / \ : * ? “ < > |. An ITSS_TelematicsApplicationID must not be empty.
ITSS_PassPhrase	The ITSS_PassPhrase has been removed for the ITSS Specification starting with version 1.2. Use Basic Authentication for mandatory security.
GNSS_Position	• GNSS_UTCtimestamp: UTCtimestamp • GNSS_Latitude: number (WGS84) • GNSS_Longitude: number (WGS84) • GNSS_Speed_kmph number • GNSS_Heading_deg number • GNSS_Accuracy (optional): number (in meters) • GNSS_Altitude (optional): number (WGS84) • ITSS_LocationInfo (optional)
GNSS_PositionList	Array of GNSS_Position
UTCtimestamp	Count of seconds since 1970-01-01 00:00 UTC with fractional part, if available
ITSS_LoadingState	Represents the loading state as case sensitive text. The defined values are: • “loaded” • “unloaded” • “overloaded” • “unknown”

ITSS_DeviceList	An array of objects with
	• ITSS_TelematicsDeviceID
	• ITSS_TransportDeviceID
	
	E.g. with ITSS_TransportDeviceID:
	`[{ { "ITSS_TransportDeviceID" : "1006-SZM5_CA730", "ITSS_TelematicsDeviceID" : " MANUF000000751" } }]`
	
	E.g. without ITSS_TransportDeviceID:
	`[{ { "ITSS_TelematicsDeviceID" : " MANUF000000751" } }]`

ITSS_SensorValueList	Array of ITSS_SensorValue
ITSS_SensorValue	An object representing a single sensor measurement with the following entries: • SamplingUTCtimestamp: UTCtimestamp • ITSS_SensorId • Value: float • ITSS_SensorType (required) • ITSS_SensorPosition (required) • ITSS_SensorTrigger (optional) Further explanations: **Value** is the measured value in float. The unit of the value is determined from the ITSS_SensorType. The following is an example for a single measurement of a pressure sensor mounted at the lid of a tank wagon: `{ "SamplingUTCTimestamp": 1436722345.154, "Uid": "A0456798BF123456", "Value": "1.75", "ITSS_SensorType": "pressure", "ITSS_SensorPosition": "tank" }`
ITSS_SensorId	20 printable ASCII codes from ‘0’ to ‘9’ and from ‘A’ to ‘Z’ built from ITSS_ManufacturerID (5 digits)

	concatenated with manufacturer specific sensor Id (15 digits)
ITSS_SensorType	Is the physical measuring type of the sensor: • "temperature" in kelvin • "relativeFillLevel" in % • "relativeHumidity" in % • "pressure" in pascal • "distance" in meter • "speed" in meter per second • "voltage" in volt • "current" in ampere • "power" in watt • "mass" in kilogram • "accelerationX" in milli g (g = 9.81 m / s²)[1] • "accelerationY" in milli g (g = 9.81 m / s²)1 • "accelerationZ" in milli g (g = 9.81 m / s²)1 • "illuminance" in lux • "gyroscopeX" in radians per second • "gyroscopeY" in radians per second • "gyroscopeZ" in radians per second • "magnetometerX" in tesla • "magnetometerY" in tesla • "magnetometerZ" in tesla • "inclinationX" in radians • "inclinationY" in radians • "inclinationZ" in radians • "signal" - false: value=0.0, true: value<>0.0 • "custom" – customer specific type, unit not specified The unit must be used from SI unit system. Additional sensor types can only be defined and issued by the ITSS practice group.

[1] Milli g is used instead of the SI unit m/s^2 to be consistent with the already introduced "Notification of a detected shock"

ITSS_SensorPosition	Defines the mounting location on the waggon (direction of orientation, designation of components based on DIN 25005, details see diagram 3 below): • “ambient” • “axleBearing[1..N][L/R]”, e.g. “axleBearing1R” • “axleBrakeDisk[1..N][L/M/R]” • “wheel[1..N][L/R]” • “wheelset[1..N]” • “bogie[1..N]” • “tank[1..N]” • “mainAir” • “brakeCylinder[1..N]” • “weighingValve” • “brakeLeverEmpty” • “handbrakeReleased” • “doorOpen[1..N][L/R]”, e.g. “doorOpen1R” • “hatchOpen[1..N]” • “waggon” • “custom” – customer specific position
ITSS_SensorTrigger	Represents the reason why a sensor value has been read and/or transmitted as a case sensitive text. The trigger can be time based, e.g. every 5 minutes or event based, e.g. a valve has been closed or a threshold value has been reached. The trigger methods can change for every sensor value, e.g. the status of a valve is checked on a regular time base and a status change is reported in the moment of opening/closing of the valve. Defined values are: • “cyclic” (For time based sensor values.) • “statusChange” • “threshold” (If a sensor value exceeds a limit. If the sensor value and limit is checked time based the threshold trigger takes priority over the cyclic trigger and the value has to be reported only once) • “requested” (If the sensor reading has been manually requested, e.g. from the telematics application.)
ITSS_GeofenceEvent	Represents the reason why a geofence notification has to be created. The content text is case sensitive. Defined values are:

	• "on_enter" (if the transport device enters the geofence area) • "on_exit" (if the transport device leaves the geofence area) • "on_both" (if the transport device enters or leaves the geofence area)

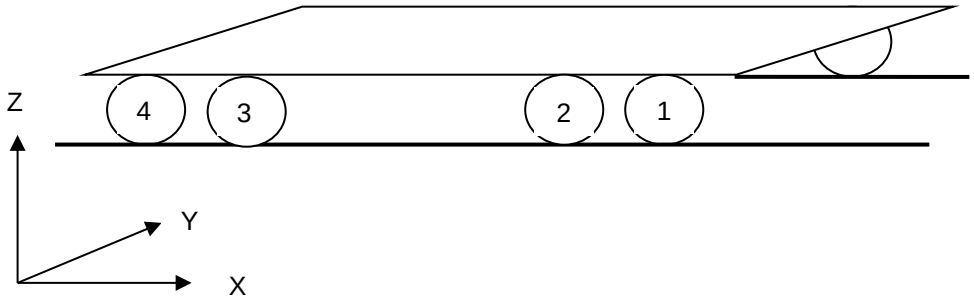


Diagram 2: wagon coordinate system

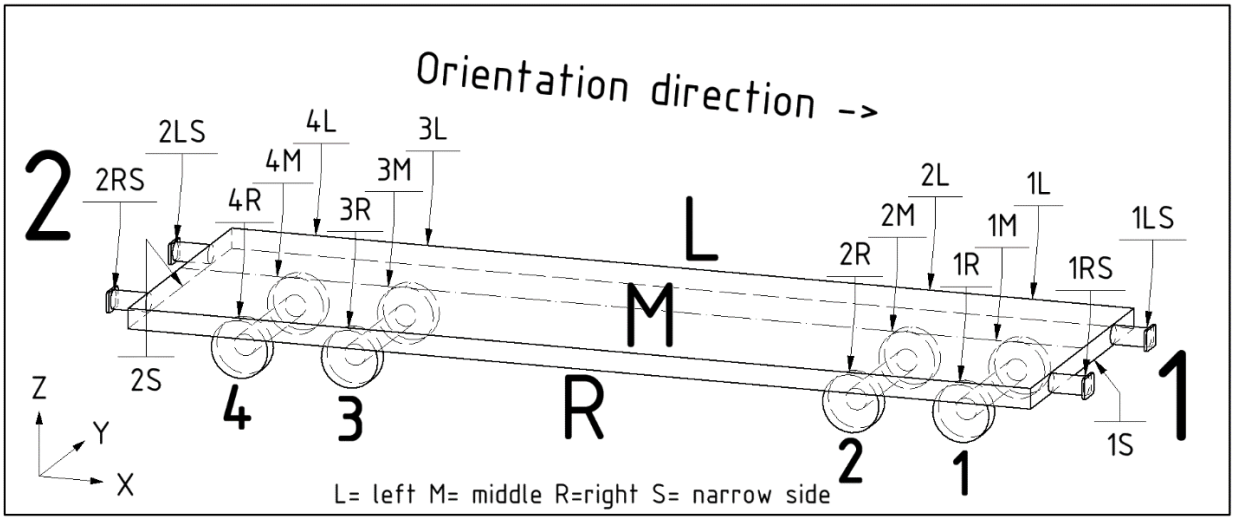


Diagram 3: Rules for representation - Direction of orientation, designation of components

Orientation examples:

1, 2	wheel set
1L, 1R, 2L, 2R	axle bearing
1L1, 1L2, 1R1, 1R2, 2L1, 2L2, 2R2, 2R2	wheel set holder
1L, 1R, 2L, 2R	door
1LS, 1RS, 2LS, 2RS	buffer

1S, 2S	coupling
1LS, 1RS, 2LS, 2RS	air cock
1L, 1M, 1R, 2L, 2M, 2R	brake disk

Waggon end definition:

- For wagons with hand brake the location of the hand brake defines wagon end 2
- For wagons with compressed-air brake the location of the piston rod of the brake cylinder defines the wagon end 2.

Change log

New in Version 1.1:

- New request allDevices added to get a list of all known devices
- New request sensorValuesTimeInterval added, to submit sensor data for a defined time interval
- New concept of push notifications introduced
- 6 push notifications added to support data push for all relevant information available via the ITSS interface 1
- Enhancement of gnss_position by gnss_altitude and gnss_accuracy
- Loading states “overloaded” and “payload” added
- Notification about a detected overload condition removed, because now included in general notification of a loading state
- Changed orientation/numbering of axles in diagram 2
- Wagon end and direction of orientation definition added, including diagram 3
- Review of security issues (no changes)

New in Version 1.2:

- The authentication via ITSS_PassPhrase has been removed from all API methods
- For authentication of each HTTP transaction, the method of basic access authentication with user name and password has been added
- Requests to the API methods can be made either by using ITSS_TransportDeviceID or ITSS_TelematicsDeviceID
- Reference to TAF TSI added
- Added new post method “Assembled notification”
- ITSS_TransportDeviceID extended to 50 printable ASCII codes (to support GIAI numbers)
- Added a new chapter about geofencing. Geofence related methods are set up for creating, reading, updating and deleting of geofences.
- Added UTCtimestamp to “Notification about the mileage” and “Request all known devices”