Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Federal Department of Finance FDF

**Federal Office for Customs and Border Security FOCBS**
Foundations Directorate

# CAMIUNS (NETS) Interface Testspecification

## N-411 - Camiuns (NETS) Interface Testspecification

## Supplement 4 zu Anhang 2 zur Verordnung Verordnung des BAZG über die technischen und betrieblichen Vorgaben für Anbieter zur Erhebung der leistungsabhängigen Schwerverkehrsabgabe

Version 1.0, 16.02.2024

**Table of Contents**

# 1 Document History

| *Version* | *Date* | *Changes* | *Writer* | *State* |
|---|---|---|---|---|
| 1.0 | 16.02.2024 | Version for information of interested NETS Providers | BAZG | final |

## 1.1 Scope

The scope of the NETS Provider Interface Test Specification is to assess the conformity of the NETS Provider's back office interface (the system to be tested) to the BAZG's.

The focus of the tests defined in this document is the assessment of compliance of the implementation of application transactions and messages as defined in NETS Messages. The primary focus is the syntax (not the semantic) of the application data units (NETS Messages) of the NETS Provider (as the sender and the receiver of messages).

This document defines the test environment and prerequisites. It further defines the test cases for the assessment of the compliance NETS Messages, including test data.

The test cases cover only the handling of correct messages. Hence, assessment of the implementation's robustness and behaviour at (the required) maximum load are outside the scope of this document.

It should be noted that the system under test includes the NETS Provider's implementation of the transport layer, whereas no explicit test cases are defined in this document to assess the compliance of the requirements of the transport layer.

## 1.2 Purpose and Audiance

This specification completes the corresponding interface specification by a number of testcase specifications which need to be implemented and successfully executed prior to operationalize the interface. The specification identifies:

- the testcases to be executed
- defines the environements on which the execution shall take place
- potentially describes the tools to be used in order to pass the test.

The target audience of this document are the IT departments of the business partners or their architects and developers.

The test cases only cover the handling of correct messages. Hence,

- assessment of the implementation's robustness and behaviour at (the required) maximum load
- signature handling

are outside the scope of this document.

The NETS interface testspecification will be all the same for the different groups of NETS providers. Therefore whenever NETS providers are mentioned in the document it is referred to both national NETS provider (NNA) as well as authorized NETS providers (ZNA).

# 2 Test Environments

The planned interface tests between the customer system and the systems of the BAZG can be divided into:

- tests with test cases and test scenarios with a "test-container" provided by BAZG.
- technical tests (connection tests, token lifecycle, tests of the basic infrastructure and functions) on the acceptance environment.
- technical tests (connection tests, token lifecycle, tests of the basic infrastructure and functions) on the production environment.

In order for a software supplier to be allowed to go operational the interface tests must be successfully completed beforehand.

## 2.1 Differences between the environments

The main component for the message exchange (B2BHub) is the same. All messages that are valid in the B2BHub in the docker network should be valid also in the production environment. The token lifecycle is slightly diffferent beween the two environments.

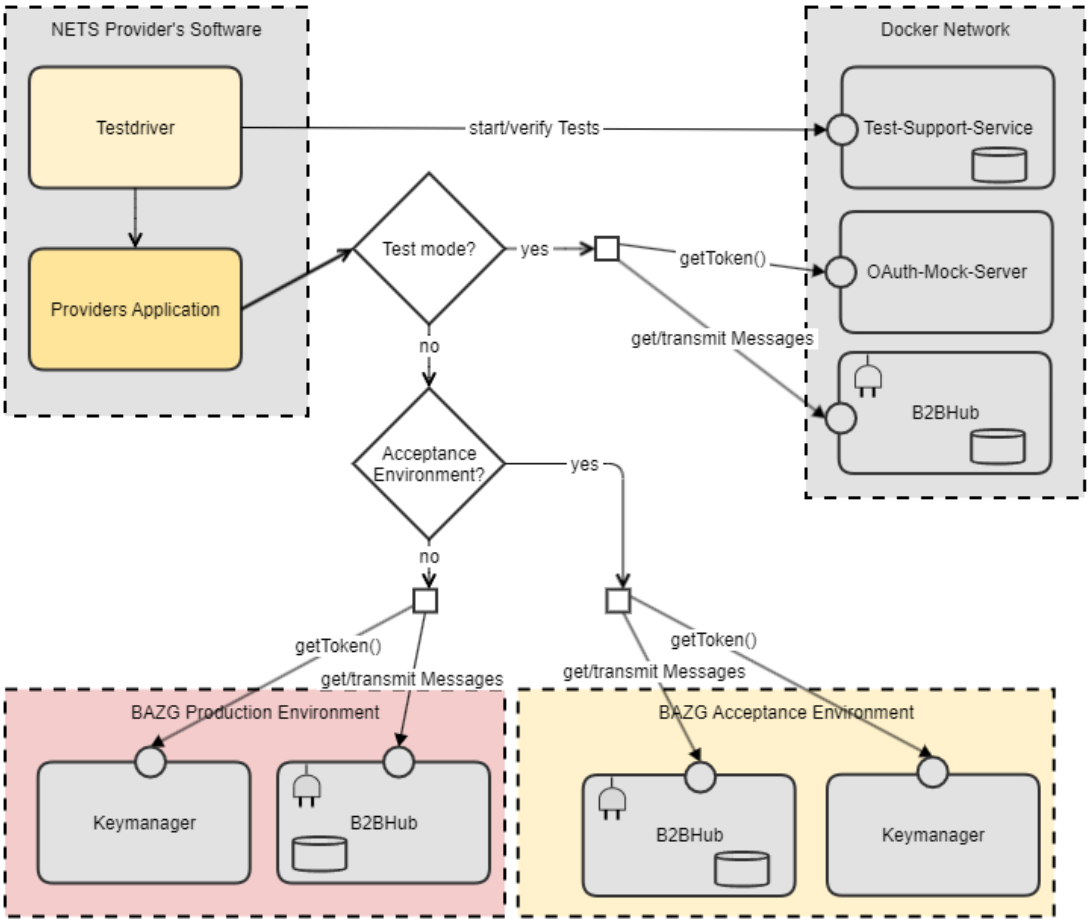



Image: Possible Architecture of the NETS Provider's Application

### 2.1.1 Test-Container Environment Preconditions

- No Preconditions needed
- Access Token can be generated in the docker container
- Test driver needed to start/verify tests and remote control providers application
- Provider's Application gets the access tokens from a component in the docker network

### 2.1.2 Acceptance Environment Preconditions

- Registred as a business partner with role *NETS Anbieter*
- A valid initial access token from self service portal
- Provider's Application must implement a token lifecycle management using an endpoint from BAZG

### 2.1.3 Production Environment Preconditions

- Registred as a business partner with role *NETS Anbieter*
- A valid initial access token from self service portal
- Provider's Application must implement a token lifecycle management using an endpoint from BAZG

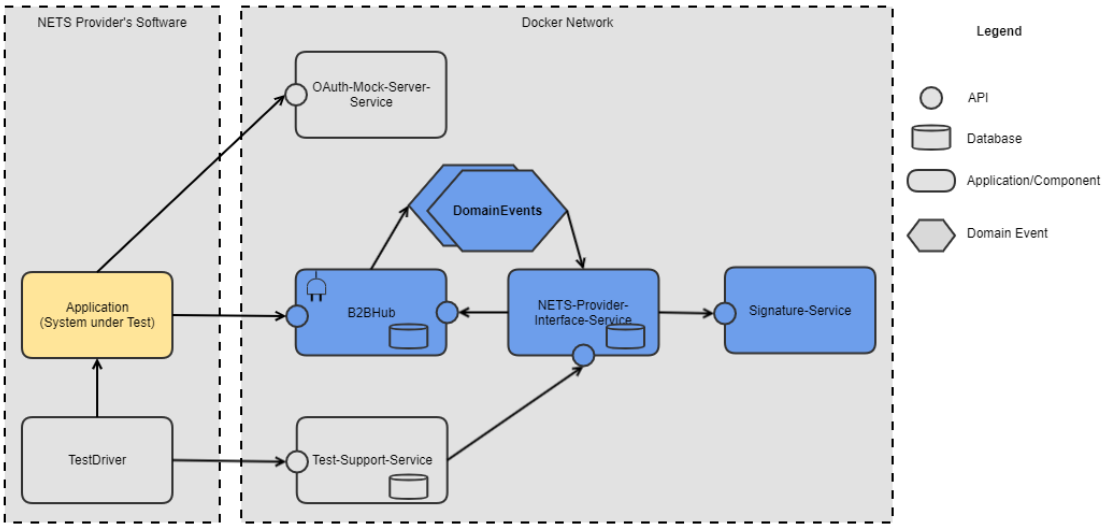
# 3 Testcase Procedures Test-Container

## 3.1 Docker Container environment overview

The test environment will be provided by the BAZG in form of a Docker Container:

- simulating BAZG's back office interface to the NETS Provider
- testing evaluation tool including basic test reporting features (i.e. test passed or failed, if failed first error indication)

The Docker Container shall be used by the NETS Provider when performing the test cases defined in this document. It can be used at the NETS Provider's premises and shall be used to assess the NETS Provider's real implementation of its back office interface to the BAZG.




The figure above illustrates the NETS Provider's real implementation of its back office to the BAZG in the test environment. The various main constituents of the testing environment are further highlighted in the following table.

| | **Description** |
|---|---|
| Providers Application<br>(System under Test) | NETS Provider's implementation of its back office interface to the BAZG. |
| Test Driver | NETS Provider's testing software to automate the tests<br>• prepare test using Test-Support-Service<br>• trigger test<br>• check result |

| | Description |
|---|---|
| Test-Support-Service | Service to prepare the test environment for a given test case and to check whether a test was successful. This service supports a list of defined test cases. For each of these test cases:<br>• it can prepare the state of the NETS Provider's Interface (contents of the database)<br>• after the executed test, it can check whether the status of the NETS Provider-Interface service is correct (i.e. as expected)<br>It may be used to record the test and generate input to the test report. |
| B2BHub | Gateway component that handles authentication and provides an API to exchange messages between a BAZG business partner (NETS Provider) and the camiuns application.<br>Communication with the test environment uses https. The NETS provider identifies himself using a token. Communication with the test system is identical to the real system, except using different tokens (and different token lifespans) for testing. |
| NETS-Provider-Interface-Services | BAZG's implementation of the EETS Provider's application interface service.<br>Responsible service to sign outgoing messages and to verify signatures on incoming messages like the real system, except using different keys and certificates for testing. |
| Domain Event | The internal application uses domain events for inter-service communication. These internals are irrelevant for a NETS Provider, but are mentioned here for describing reasons. |
| Signature Service | The signature service signs/verify the signature of the messages |

### 3.1.1 Prerequisites

A prerequisite for performing the phase 2 tests of the NETS Provider approval procedure is the successful completion of phase 1 (see section 4 in [1]).

Further, the following prerequisites apply:

- Use of (test) certificates, nothing that the Docker Container will not assess the validity of the NETS Provider's certificate in phase 2 of the assessment of the NETS Provider.
- exchange, installation and use of the test keys (incl. the import of the NETS Provider certificate in the Docker Container environment) for
  - transport layer security (see BAZG B2B-Hub-Access Point)
  - data integrity (see BAZG Message Signing)
- configuration of the test cases (test drivers) prior to their execution

### 3.1.2 Docker Container setup

#### 3.1.2.1 Prerequisites

In order to run this container the following tools have to be installed on your device:

- Docker (https://docs.docker.com/install)

- Docker-Compose (https://docs.docker.com/compose/install/)

Additionally your device should have access to https://ghcr.io

The Github container registry is used to pull the container images build by the BAZG of switzerland

#### 3.1.2.2 Get access to the Github Project

In order to pull all the images needed for the docker container setup you need to have access to our Github project.

Simply write a mail to camiuns@bazg.admin.ch with the github account (username and email address) which should be granted access.

#### 3.1.2.3 Get the newest version of container

First you need to login to the github container registry to be able to pull the images.

```
docker login ghcr.io
```

The application NETS Test-Container can be pulled on Github

**Github Url**
bazg-camiuns/NETS-Test-Container (github.com)

#### 3.1.2.4 Initial Configuration

There are several configuration options, that have to be set before the test container can be used:

- Provider Configuration
- Certificates for signing the messages

All steps are described in the projects ReadMe.md file.

##### 3.1.2.4.1 Startup

After the configuration is completed starting the test-containers is pretty simple: Just run the following command in the project root directory:

```
docker-compose up -d
```

You can also use the **clean-start-container.sh** script to start a fresh environment.

##### 3.1.2.4.2 Shutdown

```
docker-compose down
```

##### 3.1.2.4.3 Update

We will keep publishing new versions of our services to this repository as our services (and the container solution around it) evolve.

We recommend to pull from this repository on a regular basis.
If a new version is available please perform a rebuild of the containers by running:

```
docker-compose pull
```

You can also combine the build and startup command (depending on your OS and shell) by doing something like this:

```
docker-compose pull && docker-compose up -d
```

#### 3.1.2.5 Get help / Get in touch

If you have some feedback or need help please open an issue in this repository.

### 3.1.3 Docker container components

#### 3.1.3.1 b2b-hub component

The B2B hub in the Docker container also corresponds to the original component, which is also used in the acceptance or production environment.

For this reason, the BAZG B2B-Hub-Access Point also applies to this component.

Nevertheless, some things are different:

1. local urls are used
2. an other type of token is requested

| Endpoint | Method | Url |
|---|---|---|
| /messages | GET | http://localhost:9090/api/partner/v2/messages |
| /messages/{messageId} | GET | http://localhost:9090/api/partner/v2/messages/{messageId} |
| /messages/{messageId}/next | GET | http://localhost:9090/api/partner/v2/messages/{messageId}/next |
| /messages/{messageId} | PUT | http://localhost:9090/api/partner/v2/messages/{messageId} |

##### 3.1.3.1.1 Message Types

All message structure are documented under NETS Messages.

##### 3.1.3.1.2 Tokenmanagement

For generic specification see BAZG B2B-Hub-Connectivity.

The valid token for the docker container can be obtained from oauth-mock-server.

##### 3.1.3.1.3 Example

Following example shows a TollDeclaration (Regular Message) with a curl command to transmit this message to B2B Hub.

File gnss.xml see Example Testexecutions

```
curl -X PUT -T gnss.xml -v -H "bpId: 1234433211" -H "messageType: nets-regulartolldeclaration" -H "Authorization: Bearer $TOKEN" "http://localhost:9090/api/partner/v2/messages/3ea091bf-aaca-4b71-a011-a74c96e6acac"
```

The signature is missing in the example gnss.xml file. This file has to be signed first. The transmission to B2BHub should still work.

### 3.1.3.2 test-support-service component

#### 3.1.3.2.1 Test case setup

| | |
|---|---|
| **Verb** | PUT |
| **Operation Name** | Setup |
| **Endpoint** | http://localhost:8101/api/test-cases/{testCaseId} |
| **Description** | Set up a specific test case |
| **Input Parameter** | **Name** / **Required**<br>testCaseId / Yes |
| **Output Parameter** | **http-Code** / **Description**<br>204 / Test case state successfully prepared |

##### 3.1.3.2.1.1 Example

Use the following curl command to start the test case TCS01 - TOLL_DECLARATION_ACCEPTED_OK

**curl**
```
curl -X 'PUT' 'http://localhost:8101/api/test-cases/TC001' -H 'accept: */*'
```

#### 3.1.3.2.2 Test case verify

| | |
|---|---|
| **Verb** | POST |
| **Operation Name** | Verify |
| **Endpoint** | http://localhost:8101/api/test-cases/{testCaseId} |
| **Description** | Verifies a specific test case |
| **Input Parameter** | **Name** / **Required**<br>testCaseId / Yes |
| **Output Parameter** | **http-Code** / **Description**<br>200 / Test case successfully verified<br>400 / Test case has not yet been prepared |

##### 3.1.3.2.2.1 Example

**curl**
```
curl -X 'POST' 'http://localhost:8101/api/test-cases/TC001' -H 'accept: */*' -d ''
```

#### 3.1.3.2.3 Verify all test cases

| | |
|---|---|
| **Verb** | GET |
| **Operation Name** | Setup |

| | |
|---|---|
| **Endpoint** | http://localhost:8101/api/test-cases/ |
| **Description** | Verify all test cases |
| **Input Parameter** | |
| **Output Parameter** | [<br>{<br>"id": "TC001",<br>"preparedAt": "2022-11-29T15:20:05.997Z",<br>"status": "PASSED",<br>"verifiedAt": "2022-11-29T15:20:05.997Z"<br>},<br>{<br>"id": "TC002",<br>"preparedAt": "2022-11-29T15:20:05.997Z",<br>"status": "FAILED",<br>"verifiedAt": "2022-11-29T15:20:05.997Z"<br>}<br>] |

| http-Code | Description |
|---|---|
| 200 | Test case status returned |

#### 3.1.3.2.4 Testcase Delete

| | |
|---|---|
| **Verb** | DELETE |
| **Operation Name** | Setup |
| **Endpoint** | http://localhost:8101/api/test-cases |
| **Description** | Set up a specific test case |
| **Input Parameter** | |
| **Output Parameter** | |

| http-Code | Description |
|---|---|
| 204 | All test case runs deleted |

#### 3.1.3.2.5 Swagger

The Test-Support-Service provides a swagger ui http://localhost:8101/swagger-ui/index.html

#### 3.1.3.2.6 Open Api Specification

```yaml
openapi: 3.0.1
info:
  title: NETS Test Support Service
  description: Service for testing the specification conformance of the NETS API
  contact:
    name: Camiuns DevOps Teams
    url: TBD
    email: camiuns@bazg.admin.ch
  version: v1
externalDocs:
  description: TBD
  url: TBD
servers:
  - url: 'http://localhost:8101'
    description: Generated server url
tags:
  - name: Test Case Controller
    description: Controller for preparing and verifying test cases
paths:
  '/api/test-cases/{testCaseId}':
    put:
      tags:
        - Test Case Controller
      summary: Prepares the internal state for a given test case
      description: 'The internal state of previously prepared test cases will be
discarded. If this test case has already been verified, the verification result is
discarded as well.'
      operationId: prepareTestCase
      parameters:
        - name: testCaseId
          in: path
          required: true
          schema:
            type: string
      responses:
        '204':
          description: Test case state successfully prepared
    post:
      tags:
        - Test Case Controller
      summary: Verifies whether the internal state fulfils the requirements of a
given test case
      description: The test case is only verified once independently from subsequent
requests
      operationId: verifyTestCase
      parameters:
        - name: testCaseId
          in: path
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Test case successfully verified
          content:
            '*/*':
              schema:
                $ref: '#/components/schemas/TestCaseVerificationResultDto'
        '400':
          description: Test case has not yet been prepared
          content:
            '*/*':
              schema:
                $ref: '#/components/schemas/TestCaseVerificationResultDto'
```

```yaml
  /api/test-cases:
    get:
      tags:
        - Test Case Controller
      summary: Display the status of all test cases
      operationId: getTestCaseStatusList
      responses:
        '200':
          description: Test case status returned
          content:
            '*/*':
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/TestCaseStatusDto'
    delete:
      tags:
        - Test Case Controller
      summary: Reset all test case runs
      description: Basically a reset to factory settings
      operationId: resetTestCaseRuns
      responses:
        '204':
          description: All test case runs deleted
components:
  schemas:
    TestCaseVerificationResultDto:
      type: object
      properties:
        status:
          type: string
          description: The execution status of the test case
        testExceptions:
          type: array
          description: A list of exceptions from the expected test case state. Empty
if test case was successful.
          items:
            type: string
            description: A list of exceptions from the expected test case state.
Empty if test case was successful.
    RequestObjects:
      type: object
      properties:
        ids:
          uniqueItems: true
          type: array
          items:
            type: string
    Result:
      type: object
      properties:
        foundIds:
          uniqueItems: true
          type: array
          items:
            type: string
        notFoundIds:
          uniqueItems: true
          type: array
          items:
            type: string
    TestCaseStatusDto:
      type: object
      properties:
```

```
          id:
            type: string
            description: The id of the test case
            enum:
              - TC001
              - TC004
              - TC007
              - TC008
          preparedAt:
            type: string
            description: The time when the test case was prepared
            format: date-time
            nullable: true
          status:
            type: string
            description: The execution status of the test case
          verifiedAt:
            type: string
            description: The time when the test case was verified
            format: date-time
            nullable: true
    securitySchemes:
      OIDC_System:
        type: oauth2
        description: OAuth2-Authentication as System
        flows:
          clientCredentials:
            tokenUrl: 'http://localhost:9998/camiuns-oauth-mock-server/oauth/token'
            scopes: {}
      OIDC_Enduser:
        type: oauth2
        description: OAuth2-Authentication as Enduser
        flows:
          authorizationCode:
            authorizationUrl: 'http://localhost:9998/camiuns-oauth-mock-
server/oauth/authorize'
            tokenUrl: 'http://localhost:9998/camiuns-oauth-mock-server/oauth/token'
            scopes: {}
```

*Code Block 1 Open API Specification for Test-Support-Service*

#### 3.1.3.3 oauth-mock-server

##### 3.1.3.3.1 Get access token

Get a (JWT) access token to perform requests with the b2b-hub component.

| | |
|---|---|
| **Verb** | POST |
| **Operation Name** | Token |
| **Endpoint** | http://localhost:8180/camiuns-oauth-mock-server/oauth/token |
| **Description** | Returns a JWT (json web token) to perform request with the b2b-hub |
| **Header Fields** | **Name** / **Value**<br>Content-Type / application/x-www-form-urlencoded |
| **Authorization** | BASIC |

| Input Parameter | Name | Required | Value |
|---|---|---|---|
| | grant_type | Yes | client_credentials |

| Output Parameter | Name | Description |
|---|---|---|
| | access_token | The JWT Token |
| | token_type | bearer |
| | refresh_token | For the test container, there is no refresh token flow implemented. When the token expires, a new token has to be requested. |
| | expires_in | Expiration date |
| | scope | |
| | jti | |
| | id_token | |

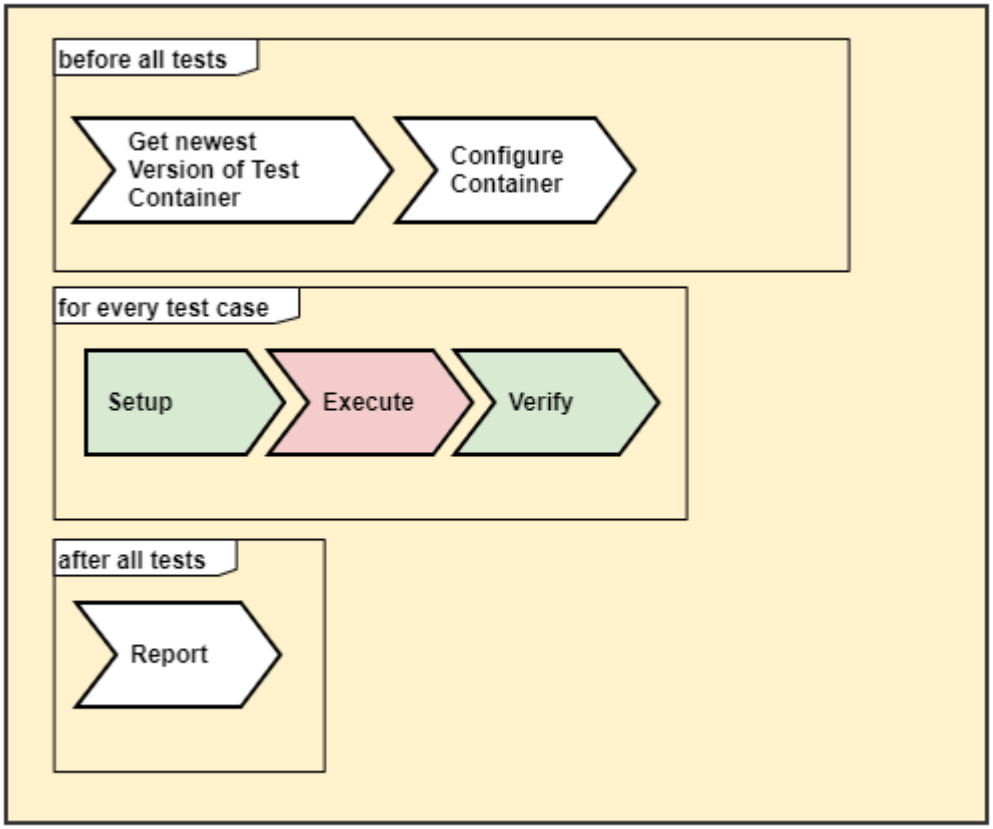
#### 3.1.3.3.2 Example

Heres an Example with curl to get an new JWT Token:

```
curl -s 'http://localhost:8180/camiuns-oauth-mock-server/oauth/token' -H 'Content-Type: application/x-www-form-urlencoded' --data-binary 'grant_type=client_credentials' -u '{username}:{password}'
```

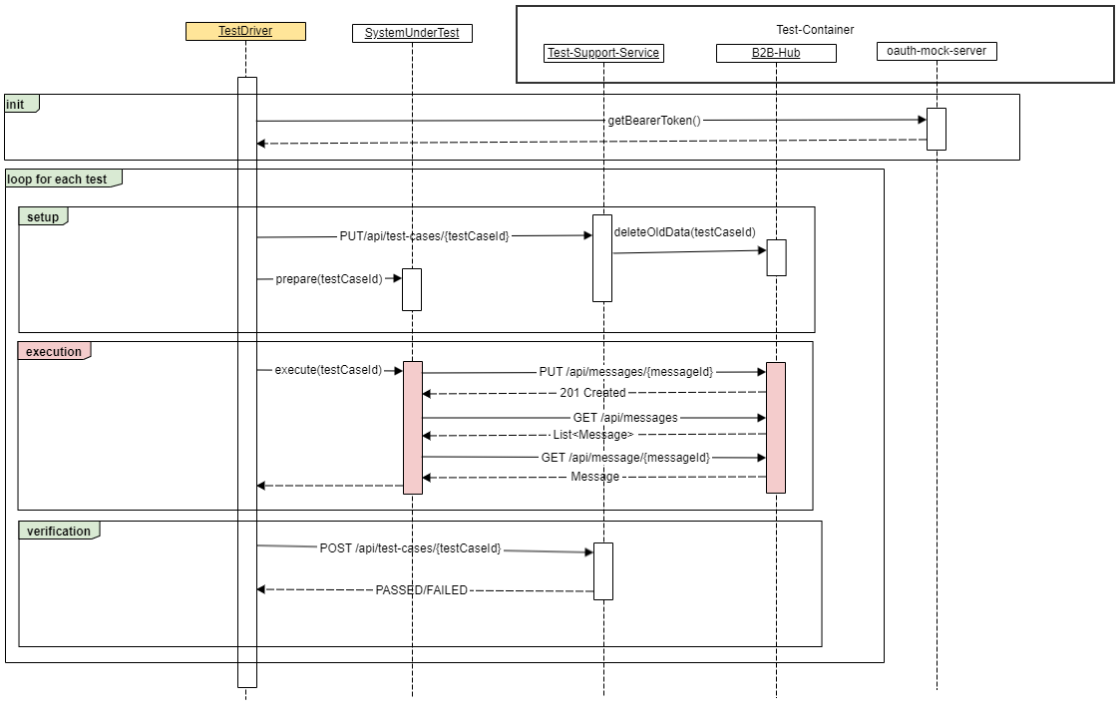
## 3.2 General test case workflow

The NETS Provider is responsible for the initialisation, preparation, execution and verification of the results of the test cases defined in this document (sections 3-5). To fulfill the admission procedure, all test case verifications have to be reported to BAZG.




The BAZG reserves the right to accompany the tests with a test witness.

The following sequence diagram illustrates the general test case workflow, the initialisation of the Test-Support-Service (once for all test cases), in the preparation, execution and verification of the test results (including basic reporting), and how the NETS Provider can test its system under test using the Docker Container.




It should be noted that some parts in the sequence diagram above are subject to design choices at the discretion of the NETS Provider, i.e. the interaction between the System under Test (SUT) and the Test Driver. The conceptual messages with the interaction between the SUT and the Test Driver have been included in order to provide an overview of the overall test case workflow.

The exchange of transport API messages during the execution step is defined in BAZG B2B-Hub-Access Point.

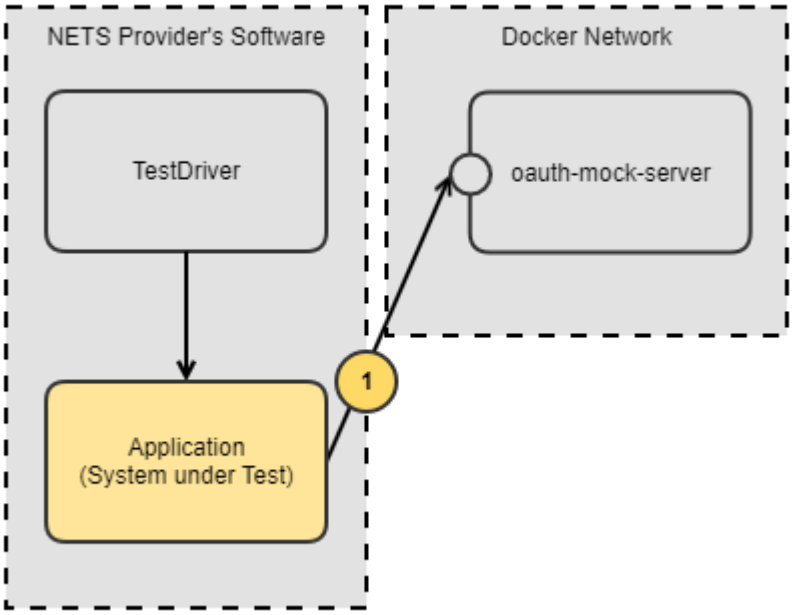
The transport API messages in the init, setup and verification steps are defined in NETS Messages.

### 3.2.1 Testcases

All availiable Testcases are found in TollDeclaration Test Scenarios.

### 3.2.2 Test initialisation

The NETS provider's application have to use a bearer token to perform the testcases.

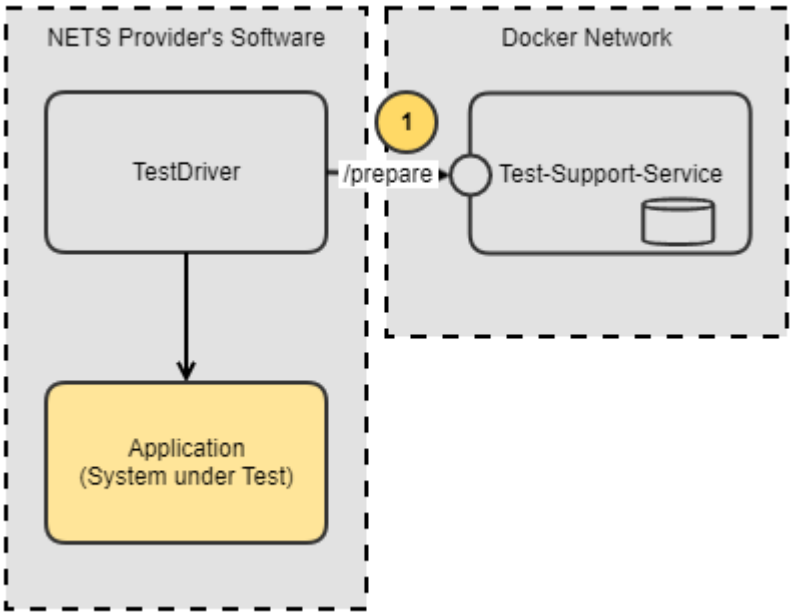



| | |
|---|---|
| **1** | To get a valid token in the test container, use oauth-mock-server API. The received token must be used to transmit and get messages from the b2b-hub component |

### 3.2.3 Test case setup

The setup method is used to start a specific test case. The following steps are performed:

- Clean up the system (Database, previous TestCase states, eg)
- Clean prior verification results of this test case
- Provide messages depending on the testcase

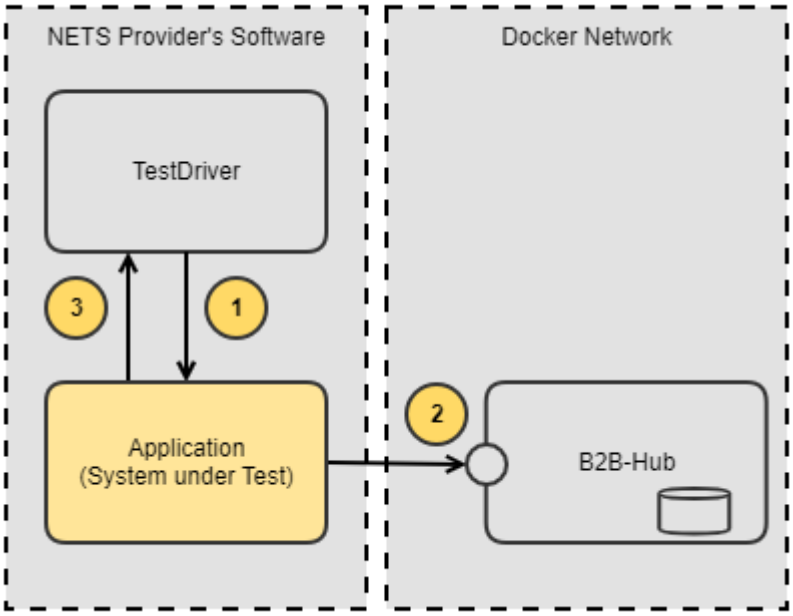



| Step | |
|---|---|
| **1** | The NETS Providers testdriver prepares the test by advising the test-support-service to prepare the system for the specified test. |

3.2.3.1 Component used

The test-support-service component API is used to setup a specific test case.

### 3.2.4 Test case execution



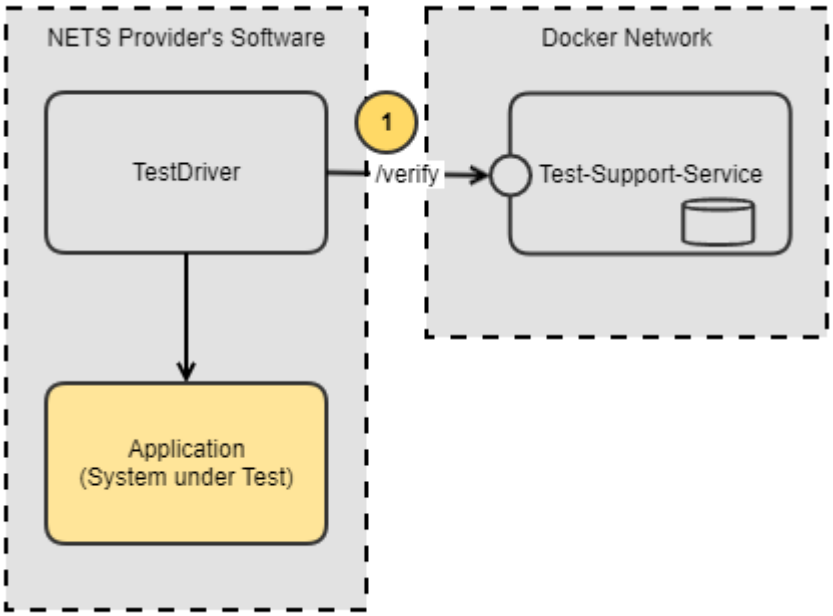
| Step | |
|---|---|
| **1** | The Testdriver guides the Application to transmit and process messages. |
| **2** | The NETS Provider Application submits and collects messages from the B2B Hub according to the specific test case |
| **3** | The TestDriver could be informed that a specific test case is performed and ready to be verified. |

3.2.4.1 Component used
The b2b-hub component API is used transmit and receive NETS Messages.

### 3.2.5 Test case verification

The setup method is used to verify a specific test case. The following steps are performed:

- Examine the incoming and outgoing messages
- Check syntactical correctness
- Check the order of messages
- Check the overall state of the system

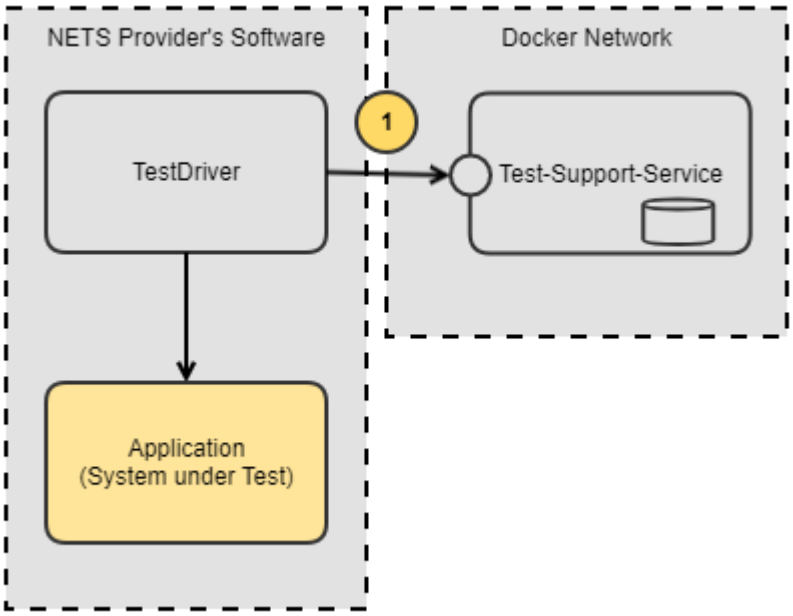



| Step | |
|---|---|
| **1** | The Testdriver verifies the executed test for correctness |

3.2.5.1 Component used
The test-support-service component API is used to verify a specific test case.

### 3.2.6 Reporting of the results

The NETS Provider is responsible for providing BAZG with a concise test report on the results of the execution of the tests defined by the test cases in this document (sections 3-5).




| Step | |
|---|---|
| 1 | All testcase results can be collected and reported to BAZG |

3.2.6.1 Concise report

The concise test report shall contain, at least, the following information:

- Identification of the NETS Service Provider (legal name and contact details)
- Unique test report identification
- Identification of the system under test (including version number)
- Identification of used test environment (Docker Container TAG)
- Overall result of the executed tests
- For each test case, the result of the executed test
    - Test case ID (as defined in this document)
    - Overall verdict of the test result (passed, inconclusive or failed)
- Date and signature of the test manager

### 3.2.7 Example Testexecutions

The Provider executes the tests cases with his own software. It is also possible to execute them with curl Requests.

Here is an example to execute the TCS01 - TOLL_DECLARATION_ACCEPTED_OK Testcase:

```
-- Start the testcase
curl -X 'PUT' 'http://localhost:8101/api/test-cases/TCS01'

-- Get the token
export TOKEN=$(curl -s 'http://localhost:8180/camiuns-oauth-mock-server/oauth2/token'
-H 'Content-Type: application/x-www-form-urlencoded' --data-binary
'grant_type=client_credentials' -u 'nets-testcontainer-b2bhub:secret' | jq
'.access_token' -r)

-- Post a gnns declaration
curl -XPUT -T gnss.xml -v -H "bpId: 1234567891" -H "Authorization: Bearer $TOKEN" -H
"Content-Type: application/xml" -H "messageType: nets-regulartolldeclaration"
"http://localhost:9090/declaration/api/messages/$(uuidgen)?topicName=declaration"

-- Poll for responses
-- The provider must store the uuid of the last message he processed, else this calls
returns all messages
curl -v -H "bpId: 1234567891" -H "Authorization: Bearer $TOKEN" -H "Content-Type:
application/xml" "http://localhost:9090/declaration/api/messages?lastMessageId{uuid-
last-processed-message}"

-- Get single response
curl -v -H "bpId: 1234567891" -H "Authorization: Bearer $TOKEN" -H "Content-Type:
application/xml" "http://localhost:9090/declaration/api/messages/{message-uuid}"

-- Acknowledget the tolldeclaration response
curl -XPUT -T acknowledge.xml -v -H "bpId: 1234567891" -H "Authorization: Bearer
$TOKEN" -H "Content-Type: application/xml" -H "messageType: nets-acknowledge"
"http://localhost:9090/declaration/api/messages/$(uuidgen)"

-- Verify the testcase
curl -X 'POST' 'http://localhost:8101/api/test-cases/TCS01'
```

#### 3.2.7.1 Example Files

```xml
<?xml version="1.0" encoding="UTF-8"?>
<message xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
    <messageContent>
        <contentHeader>
            <messageId>25a945d2-128c-4faf-8119-c9efa8182424</messageId>
            <messageDateTime>2023-06-13T12:42:11.391+02:00</messageDateTime>
            <informationSenderId>
                <issuerId>1234567891</issuerId>
            </informationSenderId>
            <informationRecipientId>
                <issuerId>1000006447</issuerId>
            </informationRecipientId>
        </contentHeader>
        <contentBody>
            <acknowledge>
                <correlationId>{uid of the message to confirm}</correlationId>
                <ackCode>OK</ackCode>
                <issues>
                    <issue>
                        <issueCode>0</issueCode>
                    </issue>
                </issues>
            </acknowledge>
        </contentBody>
    </messageContent>
   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    -- Here comes your signature -->
   </ds:Signature>
</message>
```

*Code Block 2 acknowlegde.xml*

```xml
<message>
  <messageContent>
    <contentHeader>
      <messageId>5a57dbb3-23e5-44d5-80d6-c2cb0114bbd6</messageId>
      <messageDateTime>2022-02-11T23:59:59.999Z</messageDateTime>
      <informationSenderId>
        <issuerId>1234433211</issuerId>
      </informationSenderId>
      <informationRecipientId>
        <issuerId>1000006447</issuerId>
      </informationRecipientId>
    </contentHeader>
    <contentBody>
      <tollDeclaration>
        <tollDeclarationId>1669717040423</tollDeclarationId>
        <protocolVersion>1</protocolVersion>
        <basicVersionProviderBackend>1</basicVersionProviderBackend>
        <basicVersionOnboardSystem>1</basicVersionOnboardSystem>
        <vin>XLRASH4300G232849</vin>
        <declarationPeriod>2022-02-11</declarationPeriod>
        <regularTollDeclaration>
          <regularDeclarationType>GNSS</regularDeclarationType>
          <gnssTollDeclaration>
            <rawLegs>
              <rawLeg>
                <legId>1</legId>
                <measuredPositions>
                  <measuredPosition>
                    <longitude>7431423</longitude>
                    <latitude>46948518</latitude>
                    <dateTime>2022-02-11T09:48:54.000Z</dateTime>
                  </measuredPosition>
                  <measuredPosition>
                    <longitude>7437572</longitude>
                    <latitude>46949297</latitude>
                    <dateTime>2022-02-11T13:41:01.090Z</dateTime>
                  </measuredPosition>
                  <measuredPosition>
                    <longitude>7439730</longitude>
                    <latitude>46951298</latitude>
                    <dateTime>2022-02-11T16:15:45.817Z</dateTime>
                  </measuredPosition>
                  <measuredPosition>
                    <longitude>7438956</longitude>
                    <latitude>46953438</latitude>
                    <dateTime>2022-02-11T18:50:30.544Z</dateTime>
                  </measuredPosition>
                  <measuredPosition>
                    <longitude>7439119</longitude>
                    <latitude>46956968</latitude>
                    <dateTime>2022-02-11T21:51:02.726Z</dateTime>
                  </measuredPosition>
                  <measuredPosition>
                    <longitude>7439852</longitude>
                    <latitude>46959191</latitude>
                    <dateTime>2022-02-11T23:59:59.998Z</dateTime>
                  </measuredPosition>
                </measuredPositions>
                <trailer>
                  <trailerType>T</trailerType>
                  <trailerWeight>3500</trailerWeight>
                </trailer>
              </rawLeg>
            </rawLegs>
```

```
            </gnssTollDeclaration>
          </regularTollDeclaration>
        </tollDeclaration>
      </contentBody>
    </messageContent>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      -- Here comes your signature -->
  </ds:Signature>
</message>
```

*Code Block 3 gnss.xml*

## 3.3 General Test Szenarios

### 3.3.1 Scenario Types

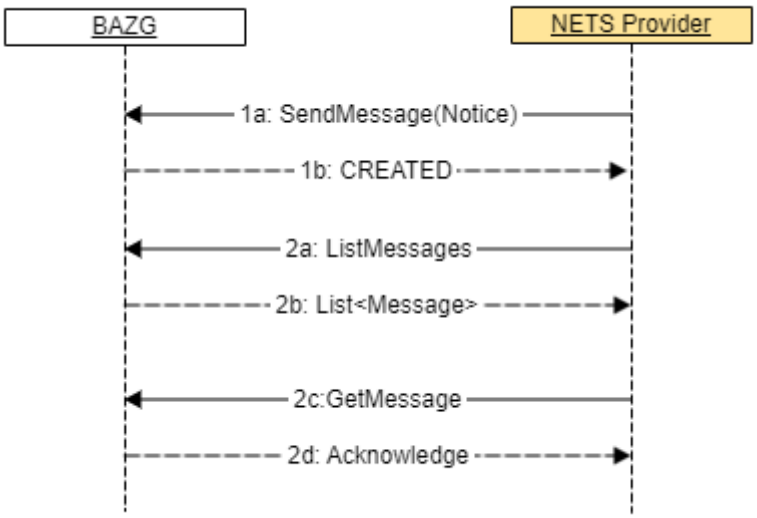
There are 2 categories of test case szenarios:

| Szenario Type | Description |
|---|---|
| SIMPLE | Mainly the delivery of messages as well as their confirmation from BAZG and the acknowledge of these messages.<br><br>These scenarios offer the possibility to submit all message types once and to consume the different answers once. Implicitly, all transmitted messages are checked by comparing the messages with the specification (xml schema). The system also checks whether the signature is present and meets expectations. These tests correspond to a simple interface test.<br><br>**Example**<br><br>→ send Request (GNSS)<br><br>← send Response (REFUSED/VALIDATION_ERROR)<br><br>→ send Acknowledge (OK) |
| COMPLEX | These complex test scenarios allow a provider to implement the different business processes in order to react adequately to them.<br><br>In the example below, the provider sends a TollDeclarationMessage to the BAZG. In this test case, we assume that the transmitted vehicle information number was wrong, respectively the vehicle could not be found in the swiss vehicle register. The BAZG therefore answers with ACCEPTED_ERROR/NO_REGISTRATION_FOR_VIN and the provider can transmit a correction of a specific vin.<br><br>**Example**<br><br>**--- Receiving inspection ----**<br><br>→ send Request (GNSS)<br><br>← send Response (OK/ACCEPTED)<br><br>→ send Acknowledge (OK)<br><br>**--- Vehicle Registration (+ 3 days) ----**<br><br>← send Response (ACCEPTED_ERROR/NO_REGISTRATION_FOR_VIN)<br><br>→ send Acknowledge (OK) |

| Szenario Type | Description |
|---|---|
| | **--- Manual correction by the provider**<br>→ send Request (MANUAL CORRECTION)<br>← send Response (REFUSED/INCORRECT_VIN)<br>→ send Acknowledge (OK)<br>Here are some examples, what use cases led to a complex test case.<br>1. Transmission and correction of a wrong vehicle identification number (VIN)<br>2. Onboarding of a new NETS service user from abroad |

### 3.3.2 API Mapping

All tests are described with sequence diagrams like this



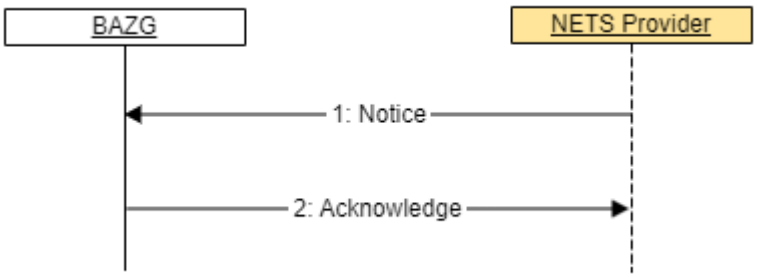

The methods match the API described in [BAZG B2B-Hub-Access Point]:

| API method name in sequence diagram | API Endpoint |
|---|---|
| SendMessage | PUT /messages/{messageId} |
| ListMessages | GET /messages |
| GetMessage | GET /messages/{messageId} OR<br>GET /messages/{messageId}/next |

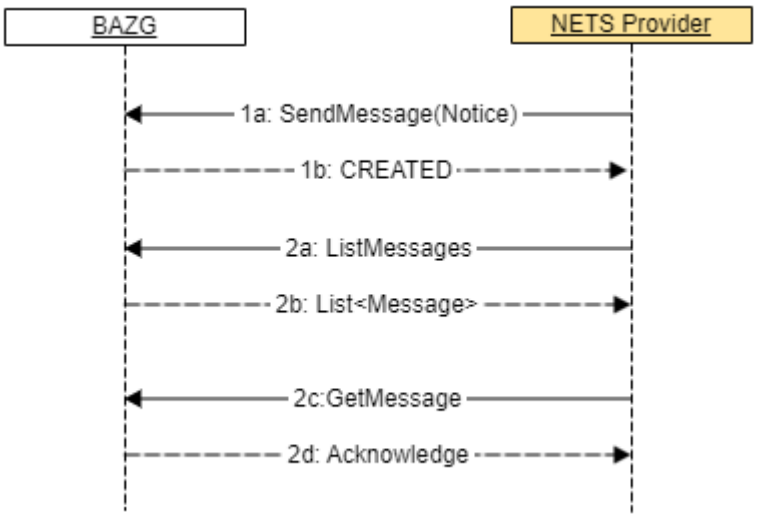
## 3.4 Notice Test Scenarios

### 3.4.1 Simple Notice message exchange scenarios

The figure below illustrates the transmission of messages from the NETS Provider to the BAZG.




The figure below illustrates conceptually the transmission of the message using the transport layer protocol, as defined in BAZG B2B-Hub-Access Point.




#### 3.4.1.1 TCS31 - Notice (Provider -> BAZG) Acknowledge OK

##### 3.4.1.1.1 Introduction

This test case can be used to simulate the start or end of a vehicle contract.

##### 3.4.1.1.2 Sequence diagram

See Simple Notice message exchange scenarios.

##### 3.4.1.1.3 Use Case

See scenario Notice (REGISTRATION)

#### 3.4.1.1.4 TestCase

| ID | TCS31 |
|---|---|
| **Title** | NOTICE_ACKNOWLEDGE_OK |
| **Description** | Standard scenario for the transmission of any kind of notice messages:<br>1. The provider sends 1 Notice message<br>2. BAZG responds with a Acknowledge message |
| **References** | Notice message, as defined in Notice<br>BAZG responds with with acknowledge message (ackCode=OK), as defined in Acknowledge. |
| **Input Data** | |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the Notice message<br>• Verification that the NETS Provider receives Notice response (with responseType = ACKNOWLEDGE and ackCode= OK) from BAZG in response to the Notice message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The correlationId matches the vin of the Notice Message. |
| **Remarks** | - |

### 3.4.2 Complex Notice message exchange scenarios

#### 3.4.2.1 TCC31_01 - Notice Registration End (BAZG -> Provider)

##### 3.4.2.1.1 Introduction

The compensation for a vehicle ends. This complex case (see General Test Szenarios) can be used to simulate the end of the obligation to dispose of the vehicle.

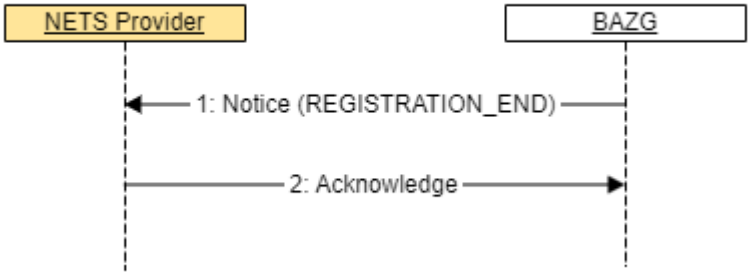
There can be several reasons for the end of the compensation obligation:

The provider sends a Registration End Notice to BAZG. This can be simulated with the test case TCS31 - Notice (Provider -> BAZG) Acknowledge OK

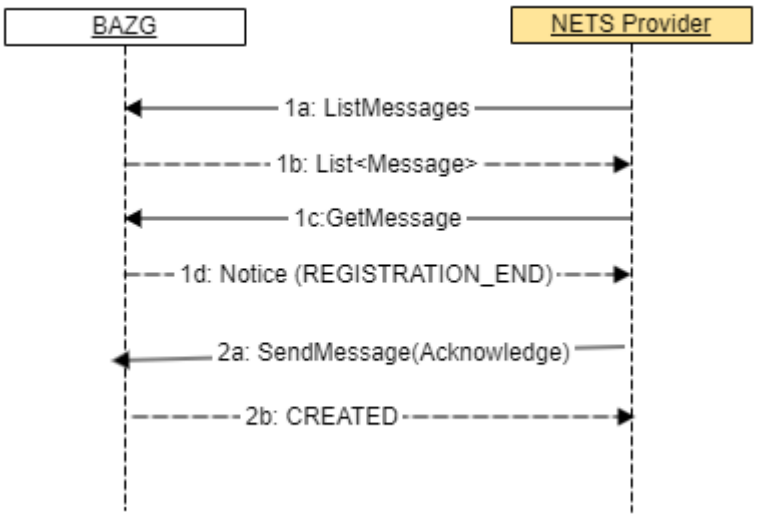
In all other cases, the end of the payment obligation is triggered by the BAZG. Possible cases are:

- Provider change (the corresponding vehicle is suddenly attached to a different provider)
- Vehicle is out of service

##### 3.4.2.1.2 Sequence diagram




The figure below illustrates conceptually the transmission of the message using the transport layer protocol, as defined in BAZG B2B-Hub-Access Point.




#### 3.4.2.1.3 Use Case

See scenario Notice (REGISTRATION)

#### 3.4.2.1.4 Test Case

| **ID** | **TCC31_01** |
|---|---|
| **Title** | NOTICE_REGISTRATION_END |
| **Description** | Szenario:<br>• BAZG sends a Notice of Type **REGISTRATION_END**<br>• Providers sends an Acknowledge (OK) |
| **References** | BAZG initiates a new Conversation with a Notice message, as defined in Notice of noticeType REGISTRATION_END<br>Provider responds with with acknowledge message (ackCode=OK), as defined in Acknowledge. |

| ID | TCC31_01 |
|---|---|
| **Input Data** | • The used vin can be overwritten (optional)<br>Test data matrix (correctVin Attribute) |
| **Expected result / success criteria** | Verification of Notice of Type **REGISTRATION_END** by the NETS Provider.<br>Correct syntax and attribute value ranges of the Acknowledge message.<br>Verification that the NETS Provider responds with an Acknowledge containing<br>• correlationId = messageId of prior sent Notice with **REGISTRATION_END**<br>• ackCode= OK<br>in response to Notice. |
| **Remarks** | |

### 3.4.2.2 TCC31_02 - Notice Status (Provider -> BAZG)

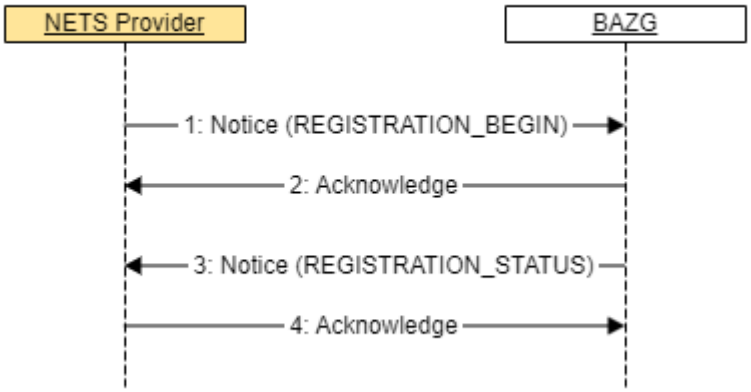
#### 3.4.2.2.1 Introduction

A provider wants to take a foreign vehicle under service. Simulate the contract takeover with this complex case (see General Test Szenarios)

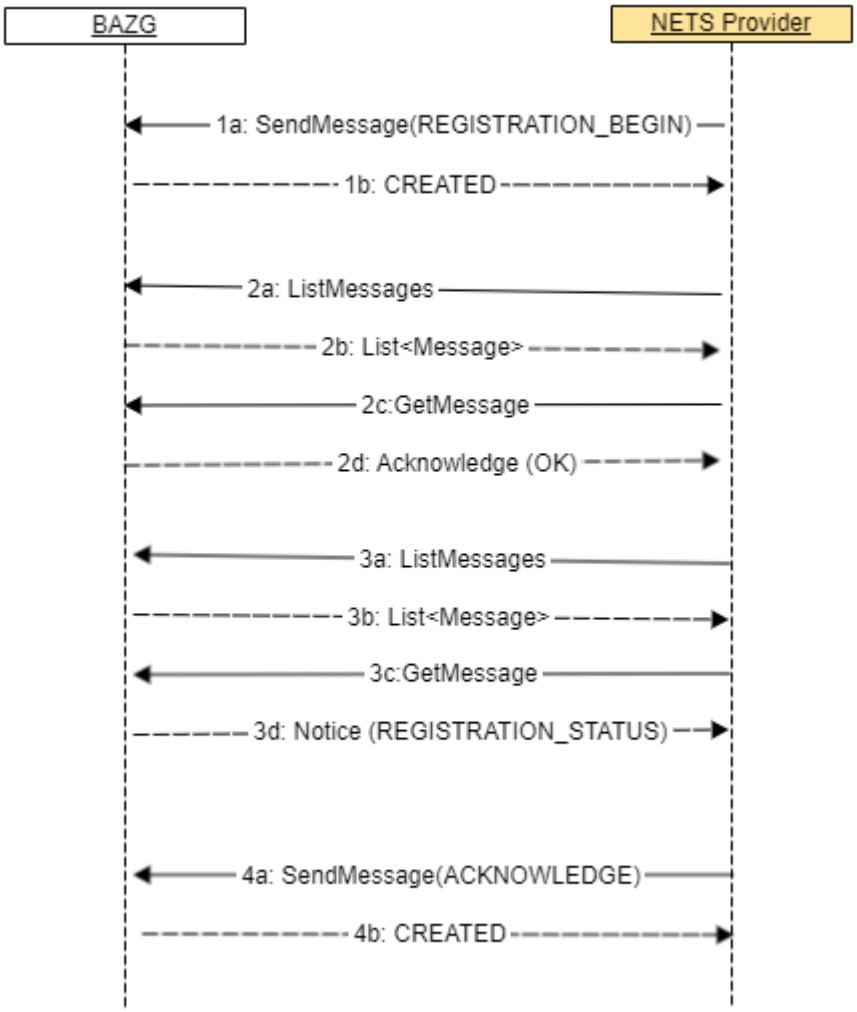
The contract acceptance of a domestic vehicle can be simulated with the test case TCS31 - Notice (Provider -> BAZG) Acknowledge OK

The difference to a domestic vehicle is that the BAZG checks whether the vehicle in question is registered. This registration is reported back with a registration status message. If the vehicle is not registered, the vehicle may not be taken under contract.

#### 3.4.2.2.2 Sequence diagram




The figure below illustrates conceptually the transmission of the message using the transport layer protocol, as defined in BAZG B2B-Hub-Access Point.




#### 3.4.2.2.3 Use Case

See scenario Notice (REGISTRATION)

#### 3.4.2.2.4 Test Case

| **ID** | **TCC31_02** |
|---|---|
| **Title** | NOTICE_REGISTRATION_STATUS |
| **Description** | Szenario :<br>• Provider sends a Notice of Type **REGISTRATION_BEGIN**<br>• BAZG sends an Acknowledge (OK)<br>• BAZG sends an Notice of Type **REGISTRATION_STATUS**<br>• Provider sends an Acknowledge (OK) |
| **References** | Provider initiates a new Conversation with a Notice message, as defined in Notice of noticeType REGISTRATION_BEGIN<br>BAZG responds with with acknowledge message (ackCode=OK), as defined in Acknowledge. |

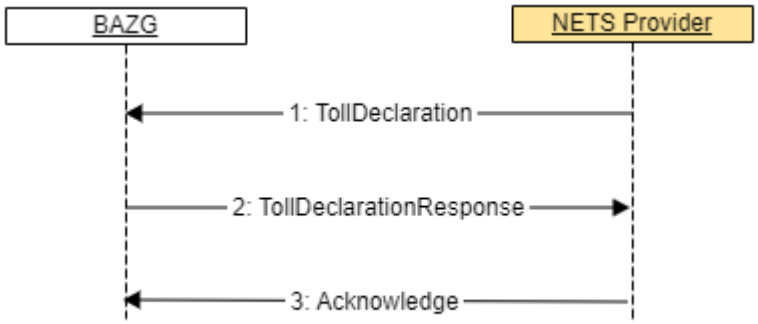
| ID | TCC31_02 |
|---|---|
| | BAZG initiates a new Conversation with a Notice message, as defined in Notice of noticeType REGISTRATION_STATUS<br>Provider responds with with acknowledge message (ackCode=OK), as defined in Acknowledge. |
| **Input Data** | • The used vin can be overwritten (optional)<br>Test data matrix (correctVin Attribute) |
| **Expected result / success criteria** | Verification that the NETS Provider transmits a Notice of type **REGISTRATION_BEGIN**<br>Verification that the BAZG responds with an Acknowledge containing<br>• correlationId = messageId of prior sent Notice with **REGISTRATION_BEGIN**<br>• ackCode= OK<br>in response to the Notice of type **REGISTRATION_BEGIN**.<br>Verification that the BAZG responds with an Notice of type **REGISTRATION_STATUS** containing<br>• isRegistered = true<br>• vin = the transmitted vin<br>Verification that the NETS Provider responds with an Acknowledge containing<br>• correlationId = messageId of prior sent Notice with **REGISTRATION_STATUS**<br>• ackCode= OK<br>in response to the Notice of type **REGISTRATION_STATUS**. |
| **Remarks** | In this example, we assume that the VIN transmitted is always a foreign vehicle. |

## 3.5 TollDeclaration Test Scenarios

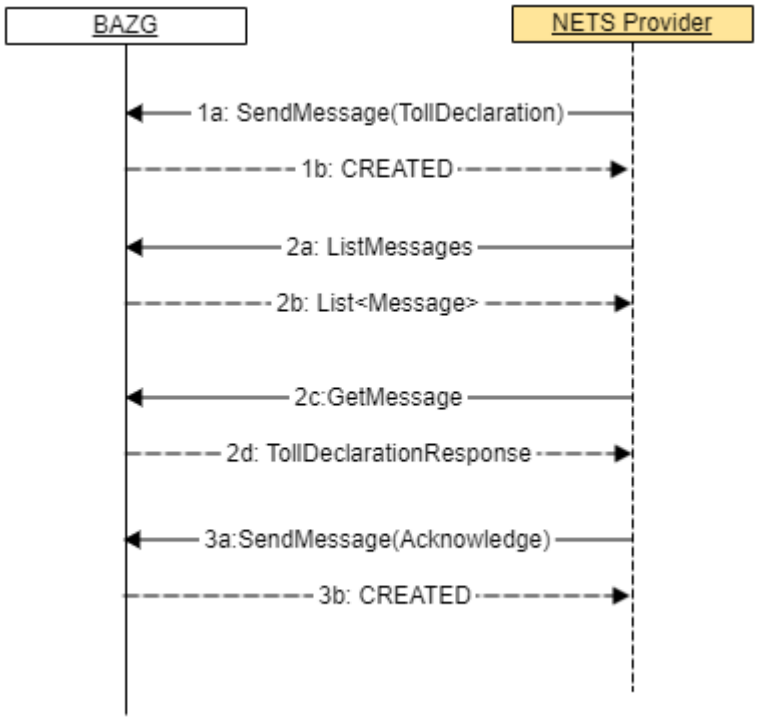
### 3.5.1 Simple TollDeclaration message exchange szenarios

#### 3.5.1.1 Overview

The figure below illustrates the transmission of messages from the NETS Provider to the BAZG.




The figure below illustrates conceptually the transmission of the message using the transport layer protocol, as defined in [2] section 3.5.




#### 3.5.1.2 TCS01 - TOLL_DECLARATION_ACCEPTED_OK

##### 3.5.1.2.1 Introduction

With this simple scenario, all TollDeclaration messages (regular and manual) can be transmitted. Both the structure and the signature are verified. A positive reply is always returned.

##### 3.5.1.2.2 Sequence Diagram

See Simple TollDeclaration message exchange szenarios.

##### 3.5.1.2.3 Use Case

See scenario TollDeclaration (OK).

#### 3.5.1.2.4 Test Case

| ID | TCS01 |
|---|---|
| **Title** | TOLL_DECLARATION_ACCEPTED_OK |
| **Description** | Standard scenario for the transmission of any kind of toll declaration message:<br>• The provider sends 1 TollDeclaration message<br>• BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages)<br>BAZG responds with with ACCEPTED (OK), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge |
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with responseType = ACCEPTED and responseReasonType = OK) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The correlationId matches the messageId of the TollDeclarationResponse. |
| **Remarks** | - |

### 3.5.1.3 TCS02 - TOLL_DECLARATION_REFUSED_DECLARATION_ID_NOT_UNIQUE

#### 3.5.1.3.1 Introduction

In this simple test case, the response from the BAZG is simulated if a submitted TollDeclaration does not have a unique ID. In this case, the declaration is refused.

#### 3.5.1.3.2 Sequence Diagramm

See Simple TollDeclaration message exchange szenarios.

#### 3.5.1.3.3 Use Case

See scenario TollDeclaration (DECLARATION_ID_NOT_UNIQUE).

#### 3.5.1.3.4 Test Case

| ID | TSC02 |
|---|---|
| **Title** | TOLL_DECLARATION_REFUSED_DECLARATION_ID_NOT_UNIQUE |
| **Description** | Szenario for the transmission of any kind of toll declaration messages where its tollDeclarationId was not unique:<br>• The provider sends 1 TollDeclaration message<br>• BAZG responds with a TollDeclarationResponse message |

| ID | TSC02 |
|---|---|
| | • The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages)<br>BAZG responds with with REFUSED (DECLARATION_ID_NOT_UNIQUE), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge |
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with responseType = REFUSED and responseReasonType = DECLARATION_ID_NOT_UNIQUE) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The correlationId matches the messageId of the TollDeclarationResponse. |
| **Remarks** | - |

#### 3.5.1.4 TCS04 - TOLL_DECLARATION_REFUSED_DEADLINE_MISSED

##### 3.5.1.4.1 Introduction

In this simple test case, the response from the BAZG is simulated if the TollDeclaration was submitted too late. In the corresponding TollDeclarationResponse, the NETS provider is informed that its declaration has been refused.

##### 3.5.1.4.2 Sequence Diagramm

See Simple TollDeclaration message exchange szenarios.

##### 3.5.1.4.3 Use Case

See TollDeclaration (DEADLINE_MISSED).

##### 3.5.1.4.4 Test Case

| ID | TCS04 |
|---|---|
| **Title** | TOLL_DECLARATION_REFUSED_DEADLINE_MISSED |
| **Description** | Standard scenario for transmission of outdated toll declarations:<br>• The provider sends 1 TollDeclaration message<br>• BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages)<br>BAZG responds with with REFUSED (DEADLINE_MISSED), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge |

| ID | TCS04 |
|---|---|
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with responseType = REFUSED and responseReasonType = DEADLINE_MISSED) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The correlationId matches the messageId of the TollDeclarationResponse. |
| **Remarks** | |

#### 3.5.1.5 TCS05 - TOLL_DECLARATION_REFUSED_VALIDATION_ERROR

##### 3.5.1.5.1 Introduction

In this simple test case, the response from the BAZG is simulated if the TollDeclaration is rejected because of validation errors. There are many different causes that lead to a validation error. The same validation error is always returned for this test case.

##### 3.5.1.5.2 Sequence Diagramm

See Simple TollDeclaration message exchange szenarios

##### 3.5.1.5.3 Use Case

See TollDeclaration, Notice (VALIDATION_ERROR).

##### 3.5.1.5.4 Test Case

| ID | TCS05 |
|---|---|
| **Title** | TOLL_DECLARATION_REFUSED_VALIDATION_ERROR |
| **Description** | Szenario for the transmission of a manual toll declaration message which has an overlap with another manual toll declaration message:<br>• The provider sends 1 manual TollDeclaration message<br>• BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Manual Correction Message).<br>BAZG responds with with REFUSED (VALIDATION_ERROR), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge |
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with responseType = REFUSED, responseReasonType = |

| ID | TCS05 |
|---|---|
| | VALIDATION_ERROR and responseReasonText = Correction period overlaps with correction period of 1 other declaration of type CORRECTION_JOURNEY) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The correlationId matches the messageId of the TollDeclarationResponse. |
| **Remarks** | - |

#### 3.5.1.5.5 Possible Validation Errors

The following table shows the different validation errors, that can occur in the declaration process (13.06.2023 - list may grow):

| ResponseType | ResponseReasonType | ResponseReasonText | Description |
|---|---|---|---|
| REFUSED | VALIDATION_ERROR | Correction period overlaps with correction period of 1 other declaration of type MANUAL_CORRECTION_TRACK | There are other manual corrections in the specific timespan that are already accepted.<br>The provider can either revoke the previous manual corrections or skip the actual message. |
| REFUSED | VALIDATION_ERROR | Correction begin is not before correction end | The manual toll declaration has a correction begin that is not before the correction end. |
| REFUSED | VALIDATION_ERROR | Correction begin date or correction end date do not match declaration date | The manual toll declaration has a declaration date that is not equal to the date of either the correction begin or the correction end. |
| REFUSED | VALIDATION_ERROR | GNSS toll declarations must contain at least one waypoint | The GNSS toll declaration has either no GNSS track, no usage statements or no usage statement with at least one waypoint. |
| REFUSED | VALIDATION_ERROR | Begin of period and end of period should cover the whole day | The manual toll declaration has |

| ResponseType | ResponseReasonType | ResponseReasonText | Description |
|---|---|---|---|
| | | | either has a correction begin that is not equal to the declaration day at 0:00:00 or a correction end that is not equal to the declaration day at 23:59:59. |
| REFUSED | VALIDATION_ERROR | Replacement VIN must be different from original VIN. | The correction VIN toll declaration has a replacement VIN that is equal to the VIN from the original toll declaration |
| REFUSED | VALIDATION_ERROR | There is already a correction VIN registered for this declaration | Another correction VIN toll declaration has previously been submitted for the same original toll declaration |
| REFUSED | VALIDATION_ERROR | Toll declaration with id *tollDeclarationId* not found or not feasible for revocation | The toll declaration meant to be revoked either not exists or is itself a revocation toll declaration |

##### 3.5.1.5.6 Technical errors

See NETS Information Model for more information about technical error. This kind of errors, cannot be handled by this test cases:

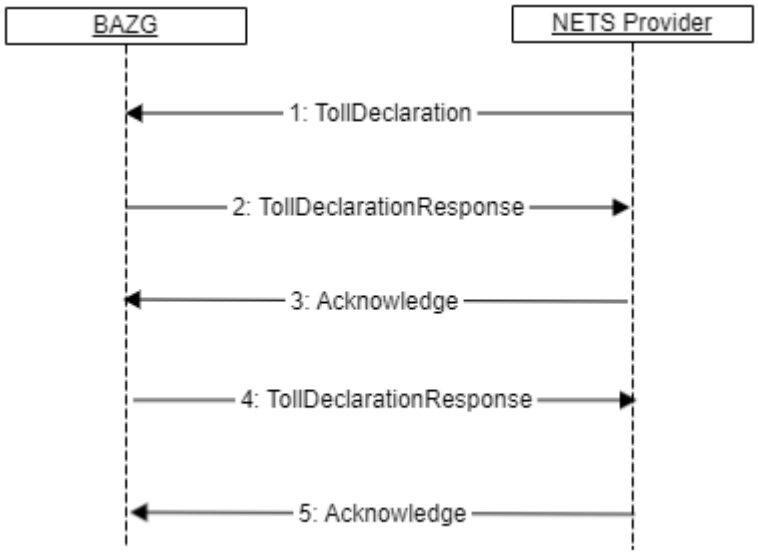
- The transmitted XML is not compliant to the XSD scheme defined by BAZG.
- The signature is not valid
- The issuer is unknown

### 3.5.2 Complex Tolldeclaration message exchange szenarios

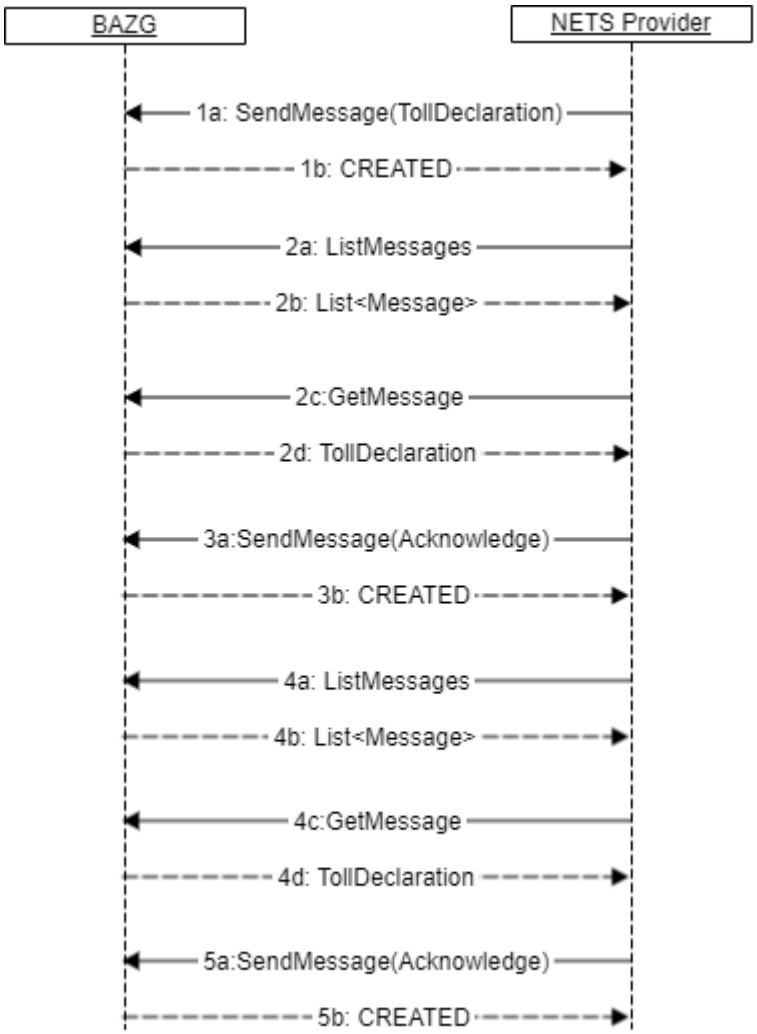
#### 3.5.2.1 Concept of business process test cases

The main concept of the complex test cases (in contrast of the Simple TollDeclaration message szenarios) is the possibility for a NETS-Provider to implement and test specific business process szenarios.

Usually, there are more interactions between a provider and BAZG than in the simple cases.




The figure below illustrates conceptually the transmission of the message using the transport layer protocol, as defined in BAZG B2B-Hub-Access Point.




### 3.5.2.2 TCC01_01 CORRECTION_VIN_SUBMITTED_INCORRECTLY

#### 3.5.2.2.1 Introduction

The HV tax is calculated based on vehicle information (weights, eg). The provider declares only the vin (vehicle identification number) and the vehicle information is loaded from the swiss vehicle register. When the message is received, only a sematic check is carried out and if this is OK, the message will be accepted.

Three days after receipt of the message, the VIN is checked against the vehicle register and confirmed with an error in the following cases:

- the VIN does not exist in the vehicle register
- no registration exists for the VIN given
- a registration exists for the given VIN, but it is inactive. (Only after a given period of time, see TollDeclaration (NO_REGISTRATION_FOR_VIN) - Registration cancelled).

This is a request to correct the incorrect VIN (if it was submitted incorrectly).

Once the VIN has successfully corrected by the provider, as requested in this scenario, the messages in error will be accepted during activation.

Note: if no successfull correction has been received, the message will be refused at day 11 following the declaration period.

#### 3.5.2.2.2 Sequence Diagram

See Complex Tolldeclaration message exchange szenarios.

#### 3.5.2.2.3 Use Case

See TollDeclaration (NO_REGISTRATION_FOR_VIN).

#### 3.5.2.2.4 Test Case

| **ID** | **TCC01_01** |
|---|---|
| **Title** | CORRECTION_VIN_SUBMITTED_INCORRECTLY |
| **Description** | Standard scenario for transmission of GNSS toll declarations:<br>• The provider sends 1 TollDeclaration message containing 1 RawLeg<br>• BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message<br>• After a short duration BAZG responds with another TollDeclarationResponse message<br>• The provider acknowledges the second TollDeclarationResponse with an Acknowledge message<br>• The provider send 1 TollDeclaration - CorrectionVin message with the correct VIN<br>• BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages), with<br>• declarationType = GNSS, as defined in TollDeclaration (Regular Message)<br>BAZG responds with with ACCEPTED (OK), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge<br>BAZG responds with ACCEPTED_ERROR (NO_REGISTRATION_FOR_VIN), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge<br>TollDeclaration - CorrectionVin message, as defined in TollDeclaration (Manual Correction Messages)<br>• manualDeclarationType = CORRECTION_VIN<br>• replacementVin = the VIN used for correction<br>BAZG responds with with ACCEPTED (OK), as defined in TollDeclarationResponse<br>Acknowledge with ackCode = OK, as defined in Acknowledge |
| **Input Data** | correctVin = during test preparation set the vin, which is used as replacementVin in the CORRECTION_VIN message defined in TollDeclaration (Manual Correction Messages). |

| ID | TCC01_01 |
|---|---|
| | Check Test data matrix for more information |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with responseType = ACCEPTED and responseReasonType = OK) from BAZG in response to the TollDeclaration message<br>• Verification that the NETS Provider receives the second TollDeclarationResponse message (with responseType = ACCEPTED_ERROR and responseReasonType = NO_REGISTRATION_FOR_VIN) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the manual TollDeclaration - CorrectionVin message<br>• Verification that the NETS Provider receives the third TollDeclarationResponse message (with responseType = ACCEPTED and responseReasonType = OK) from BAZG in response to the manual TollDeclaration - CorrectionVin message<br>• Correct syntax and attribute value ranges of the Acknowledge messages. The correlationId matches the messageId of the TollDeclarationResponse. |
| **Remarks** | - |

#### 3.5.2.3 TCC01_02 CORRECTION_VIN_SUBMITTED_CORRECTLY_FILED_INCORRECTLY

##### 3.5.2.3.1 Introduction

Not all cases in which the BAZG reports an error of the type NO_REGISTRATION_FOR_VIN are caused by an invalid vin. It is possible that a vin was incorrectly registered with the road traffic office. In such cases, the holder must correct the vin promptly at the road traffic office. Note: This error can occur where the provider reads the vin directly from the vehicle. In cases where the vin has taken over from the registration papers a wrongly registered vin at the road trafic office may not be discovered but also has no effect on proper processing.

##### 3.5.2.3.2 Sequence Diagram

See Complex Tolldeclaration message exchange szenarios.

##### 3.5.2.3.3 Use Case

See TollDeclaration (NO_REGISTRATION_FOR_VIN).

##### 3.5.2.3.4 Test Case

| ID | TCC01_02 |
|---|---|
| **Title** | CORRECTION_VIN_SUBMITTED_CORRECTLY_FILED_INCORRECTLY |
| **Description** | Standard scenario for transmission of GNSS toll declarations:<br>• The provider sends 1 TollDeclaration message containing 1 RawLeg<br>• BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |

| ID | TCC01_02 |
|---|---|
| | • After a short duration BAZG responds with another TollDeclarationResponse message<br>• The provider acknowledges the second TollDeclarationResponse with an Acknowledge message<br>• After some time, before activation, BAZG responds with a TollDeclarationResponse message<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages), with<br>• `declarationType = GNSS`, as defined in TollDeclaration (Regular Message)<br>BAZG responds with with `ACCEPTED (OK)`, as defined in TollDeclarationResponse.<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge<br>BAZG responds with `ACCEPTED_ERROR (NO_REGISTRATION_FOR_VIN)`, as defined in TollDeclarationResponse.<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge<br>After some time, before activation, BAZG responds with with `ACCEPTED (OK)`, as defined in TollDeclarationResponse.<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge |
| **Input Data** | Check Test data matrix for more information |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with `responseType = ACCEPTED` and `responseReasonType = OK`) from BAZG in response to the TollDeclaration message<br>• Verification that the NETS Provider receives the second TollDeclarationResponse message (with `responseType = ACCEPTED_ERROR` and `responseReasonType = NO_REGISTRATION_FOR_VIN`) from BAZG in response to the TollDeclaration message<br>• Verification that the NETS Provider receives the third TollDeclarationResponse message (with `responseType = ACCEPTED` and `responseReasonType = OK`) from BAZG in response to the original TollDeclaration - GNSS message<br>• Correct syntax and attribute value ranges of the Acknowledge messages. The `correlationId` matches the `messageId` of the TollDeclarationResponse. |
| **Remarks** | - |

#### 3.5.2.4 TCC02_01 - DECLARATION_FROM_DIFFERENT_PROVIDER_EXISTS

##### 3.5.2.4.1 Introduction

If a vehicle has been registered at a new provider, messages from its predecessor are refused.

##### 3.5.2.4.2 Sequence Diagram

See Simple TollDeclaration message exchange szenarios.

##### 3.5.2.4.3 Use Case

See TollDeclaration (DECLARATION_FROM_DIFFERENT_PROVIDER_EXISTS).

##### 3.5.2.4.4 Test Case

| **ID** | **TCC02_01** |
|---|---|
| **Title** | DECLARATION_FROM_DIFFERENT_PROVIDER_EXISTS_NEW_FIRST |
| **Description** | Standard scenario for transmission of toll declarations from exhausted provider:<br>• The provider sends 1 TollDeclaration message<br>• BAZG responds with a TollDeclarationResponse message: `responseType = REFUSED` and `responseReasonType = DECLARATION_FROM_DIFFERENT_PROVIDER_EXISTS`<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages)<br>BAZG responds with `REFUSED` (`DECLARATION_FROM_DIFFERENT_PROVIDER_EXISTS`), as defined in TollDeclarationResponse.<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge |
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with `responseType = REFUSED` and `responseReasonType = DECLARATION_FROM_DIFFERENT_PROVIDER_EXISTS`) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The *correlationId* matches the *messageId* of the *TollDeclarationResponse*. |
| **Remarks** | |

#### 3.5.2.5 TCC03_02 - CORRECTION_JOURNEY

##### 3.5.2.5.1 Introduction

If a vehicle has been moved on an non taxable journey ex. being towed, loaded on a train a holder can prevent taxation by sending a message of type CORRECTION_JOURNEY.

##### 3.5.2.5.2 Sequence Diagram

See Simple TollDeclaration message exchange szenarios.

#### 3.5.2.5.3 Use Case

See TollDeclaration (CORRECTION_JOURNEY).

#### 3.5.2.5.4 Test Case

| **ID** | **TCC03_02** |
|---|---|
| **Title** | CORRECTION_JOURNEY_VEHICLE_LOADED |
| **Description** | • The provider sends 1 TollDeclaration message with type CORRECTION_JOURNEY<br>• BAZG responds with a TollDeclarationResponse message: `responseType = ACCEPTED` and `responseReasonType = OK`<br>• The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Manual Correction Message)<br>BAZG responds with ACCEPTED (`OK`), as defined in TollDeclarationResponse [95].<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge [98] |
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with `responseType = ACCEPTED` and `responseReasonType = OK`) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge message. The *correlationId* matches the *messageId* of the *TollDeclarationResponse*. |
| **Remarks** | |

### 3.5.2.6 TCC05_01 - CORRECTION_TRAILER

#### 3.5.2.6.1 Introduction

Correction Trailer always aligns to a previously submitted regular TollDeclaration of type GNSS. The timeframe in which CORRECTION_TRAILER can be applied shall be limitted to underlying GNSS tracks.

In this scenario a GNSS track with 2 different legs are submitted. Later the second of the 2 legs is being corrected by the holder, splitting the second half of it to a journey without trailer.

Original declaration:

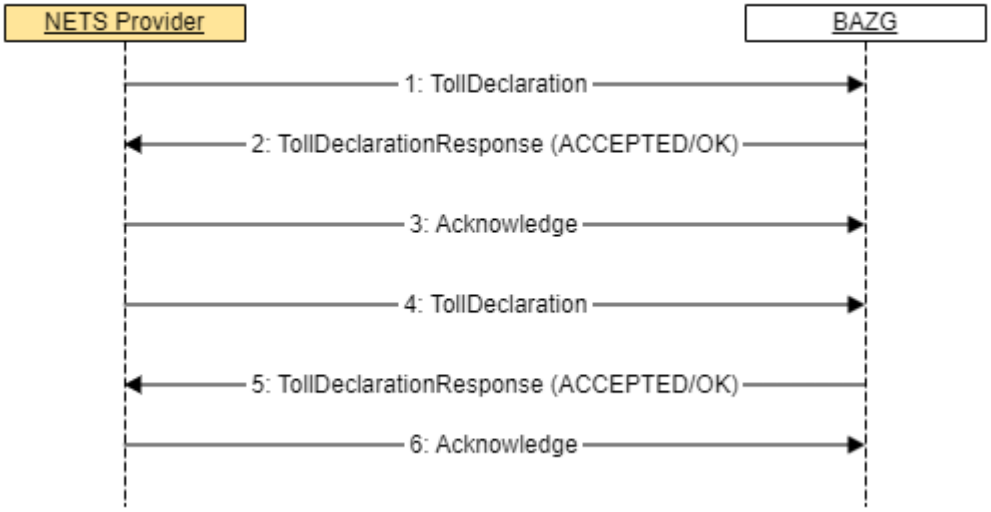
14:00 - 16:00 GNSS / Declared Trailer 18t

Manual correction:

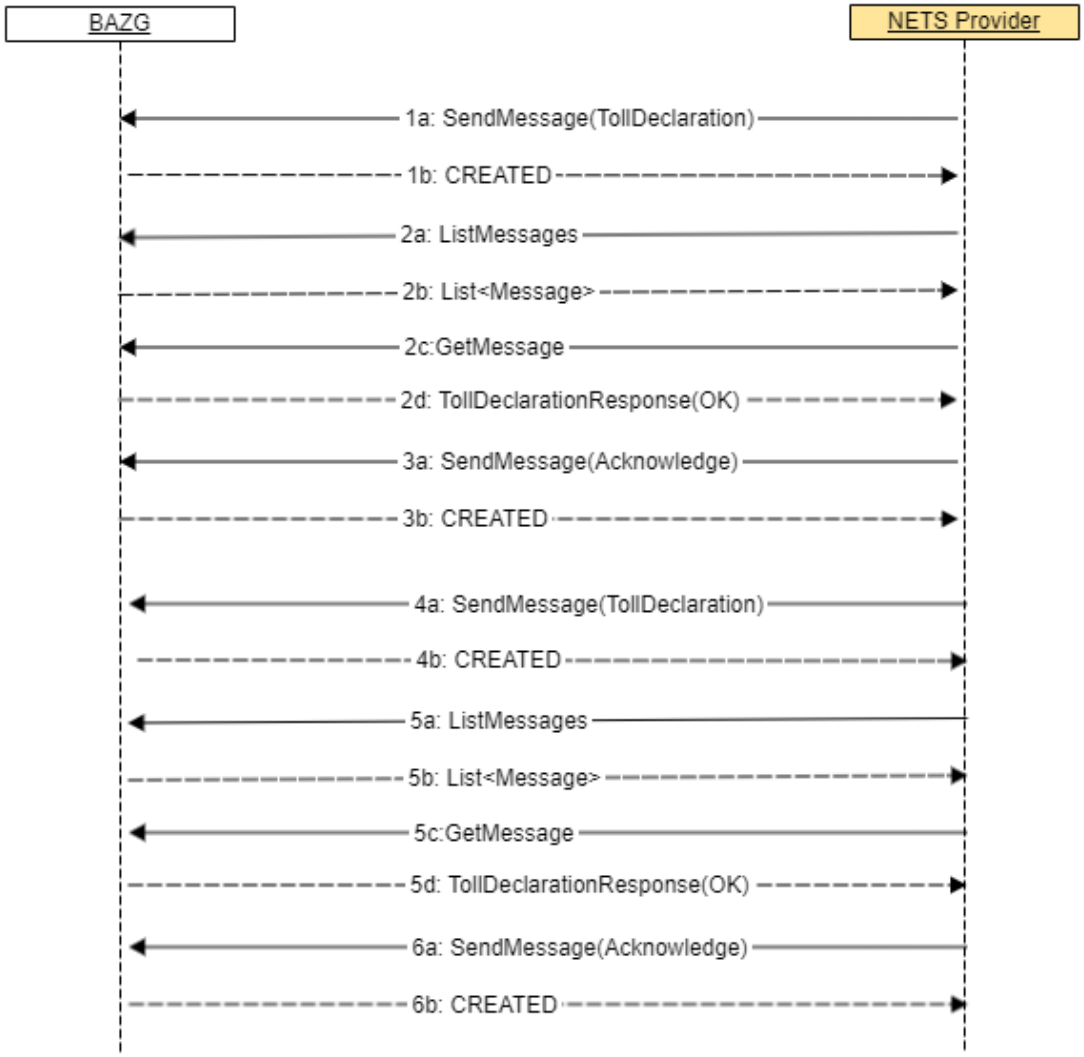
Correction Trailer / Wrong Trailer

15:00 - 16:00 CORRECTION_TRAILER / with NO_TRAILER

#### 3.5.2.6.2 Sequence Diagram




The figure below illustrates conceptually the transmission of the message using the transport layer protocol, as defined in BAZG B2B-Hub-Access Point.




#### 3.5.2.6.3 Use Case

See TollDeclaration (CORRECTION_TRAILER).

#### 3.5.2.6.4 Test Case

| **ID** | **TCC05_01** |
|---|---|
| **Title** | CORRECTION_TRAILER |
| **Description** | The following scenario for transmission of GNSS toll declarations applies to the test case:<br>•<br>○ The provider sends 1 TollDeclaration message containing 1 RawLeg<br>▪ The first WayPoint is at 14:00 and the last WayPoint is at 16:00<br>○ BAZG responds with a TollDeclarationResponse message |

| ID | TCC05_01 |
|---|---|
| | ○ The provider acknowledges the TollDeclarationResponse with an Acknowledge message<br>○ The provider sends 1 Manual TollDeclaration message of type CORRECTION_TRAILER (with no trailer information)<br>▪ The correction period is 15:00 - 16:00<br>○ BAZG responds with a TollDeclarationResponse message<br>○ The provider acknowledges the TollDeclarationResponse with an Acknowledge message |
| **References** | TollDeclaration message, as defined in TollDeclaration (Regular Messages), with<br>• `declarationType = GNSS`, as defined in TollDeclaration (Regular Messages)#TollDeclarationMessage_GNSS<br>BAZG responds with with `ACCEPTED (OK)`, as defined in TollDeclarationResponse.<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge<br>Manual TollDeclaration message (CORRECTION_TRAILER, as defined in TollDeclaration (Manual Correction Messages), with<br>• `manualDeclarationType = CORRECTION_TRAILER` , as defined in TollDeclaration (Manual Correction Messages),<br>BAZG responds with with `ACCEPTED (OK)`, as defined in TollDeclarationResponse.<br>Acknowledge with `ackCode = OK`, as defined in Acknowledge |
| **Input Data** | - |
| **Expected result / success criteria** | • Correct syntax and attribute value ranges of the TollDeclaration message<br>○ the waypoints are between 14:00 and 16:00<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with `responseType = ACCEPTED` and `responseReasonType = OK`) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Manual TollDeclaration message (CORRECTION_TRAILER)<br>○ the correction period is 15:00 - 16:00<br>○ no trailer info is delivered<br>• Verification that the NETS Provider receives TollDeclarationResponse message (with `responseType = ACCEPTED` and `responseReasonType = OK`) from BAZG in response to the TollDeclaration message<br>• Correct syntax and attribute value ranges of the Acknowledge messages. The `correlationId` matches the `messageId` of the TollDeclarationResponse. |
| **Remarks** | |

## 3.6 TollDeclaration test data

This section defines test data that are referenced in test cases.

### 3.6.1 Dynamic Testdata

It can be advantageous for a provider's specialist application to validate its own data. Otherwise, the transmitted data must correspond to the following table.

Overridden test data can be specified in the request body of the Test case setup phase.

| Parameter name | Type | Value Range | Default Value | Description |
|---|---|---|---|---|
| correctVin | String | | VF6WTTG40E9999999 | Used in szenarios, where a wrong VIN has to be corrected (for example due to a typo). This VIN is verified as the correct VIN. |
| protocolVersion | String | | 0 | Used to maintain different versions of this interface. |
| basicVersionProviderBackend | Sting | | 1.0.0.1 | The basicVersionProviderBackend contains the software version of the backend used at the providers side to generate this request. |
| basicVersionOnboardSystem | String | | 1.0.0 | The basicVersionOnboardSystem contains the software and hardware version of the onboardsystem used or to generate this request. |

#### 3.6.1.1 Specific Testdata for every test including TollDeclaration messages

The following testdata will be verified in every test containing TollDeclaration messages:

- protocolVersion
- basicVersionProviderBackend
- basicVersionOnboardSystem

#### 3.6.1.2 Specific Testdata

| Test case id | Used Testdata |
|---|---|
| TCC01_01 CORRECTION_VIN_SUBMITTED_INCORRECTLY | correctVin |

# 4 Testcase Procedures Acceptance

Depending on the NETS provider type, other tests must be run on the acceptance environment.

An overview per type can be found in the following table:

| Provider Type | Testcase Documentation |
|---|---|
| NNA | Alle Kapitel betr. Probebetrieb<br>420 - NNA Test- u...spezifikation.pdf |
| ZNA | Kapitel 2.3 Probebetrieb<br>Supplement 3 ZN... 13.10.2023.pdf |

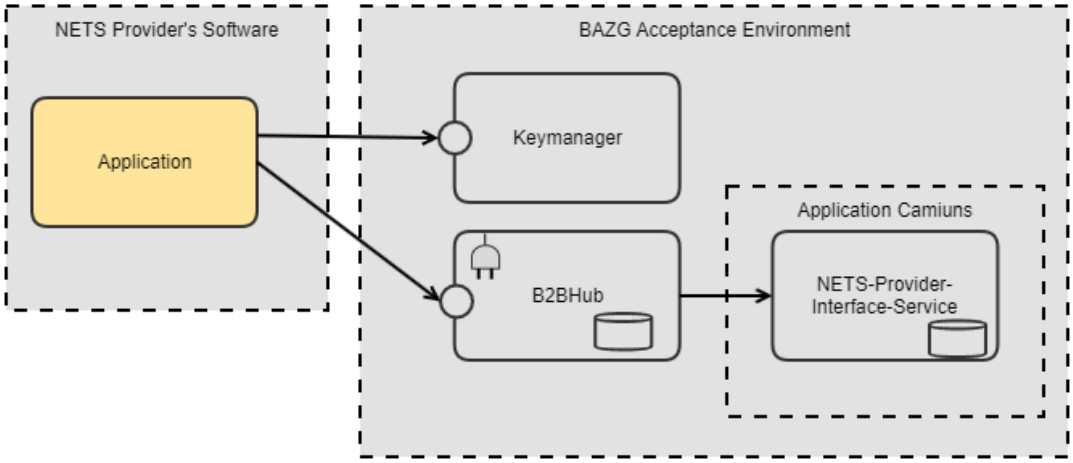
## 4.1 Acceptance environment overview

After surpassing all interface tests with the test container environment, the NETS provider's back office interface can be integrated with the acceptance environment.

The integration of the acceptance environment offers several advantages:

- The token flow can be integrated with a keymanager that is more near to the production environment than the docker container environment
- New features can be tested in a dedicated environment without the impacts on real assessment notices, billings, etc

The acceptance environment does not have the same requirements in terms of stability and availability as the production environment.




### 4.1.1 Acceptance environment components

4.1.1.1 b2b-hub component (acceptance)
See BAZG B2B-Hub-Access Point.

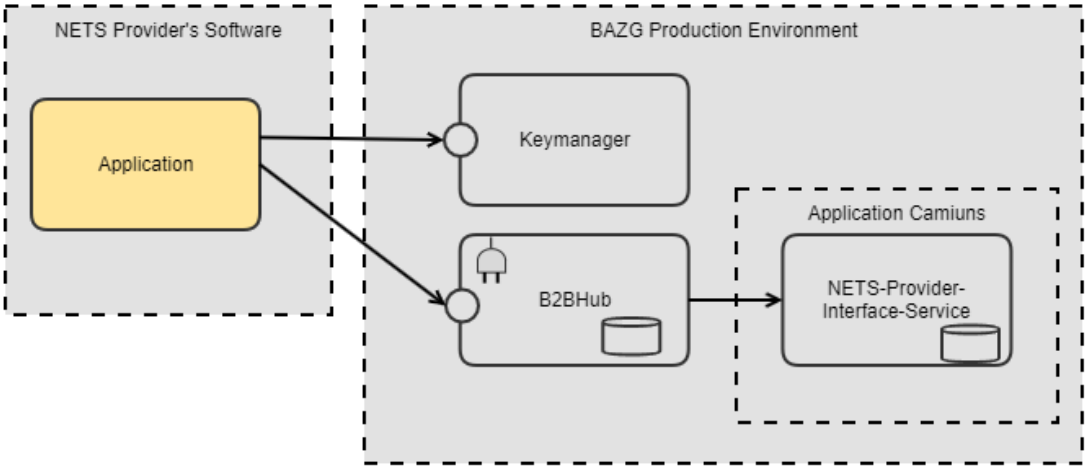
4.1.1.2 key-manager (acceptance)
See BAZG B2B-Hub-Connectivity.

# 5 Testcase Procedures Production

No test cases need to be run in the production environment.

## 5.1 Production environment overview

After surpassing all interface tests with the test container and the acceptance environment, the NETS provider's back office interface can be integrated with the production environment. During the admission procedure, declarations can be made without being billed. If the admission prodecure ends, the NETS Providers application is already fully integrated.




### 5.1.1 Production environment components

5.1.1.1 b2b-hub component (production)
See BAZG B2B-Hub-Access Point.

5.1.1.2 key-manager (production)
See BAZG B2B-Hub-Connectivity.