



## **EU-MIDT**

Card Issuing & Networking Committee

EU-MIDT/CINC/011-2006

TACHOnet Test Plan

Version 01\_40



**REF : EU-MIDT/CINC/011-2006**

**EU-MIDT SECRETARIAT DOCUMENT PREPARATION**

OPERATION	NAME	ORGANISATION	DATE
PREPARED BY	DG TREN	European Commission	02/08/2004
CHECKED BY	Thierry GRANTURCO	Granturco & Partners	26/07/2006
APPROVED BY	Marie-Christine BONNAMOUR	Cybèle – MIDT Secretariat	26/07/2006
ISSUED BY	Secretariat MIDT	MIDT	27/07/2006

---

**CHANGE CONTROL LIST**

VERSION	DATE	NAME	DESCRIPTION



**EUROPEAN COMMISSION**

DIRECTORATE-GENERAL FOR ENERGY AND TRANSPORT

# TACHOnet

## Test Plan

**02-Aug-04**

**Version 01\_40**



## Document Approval

	NAME	DATE	SIGNATURE
Prepared by:	F. Silvestre	02-Aug-04	
Checked by:	W. Van Acker		
Quality control by:	P. Delmée		
Approved by:			

## Distribution List

COMPANY	NAME	FUNCTION	FOR INFO / APPROVAL
DG TREN	Y. Hardy	Project Manager	Approval
DG TREN	L. Huberts		Info
Getronics	P. Delmee	Project Manager	Info
Getronics	F. Silvestre	Technical Coordinator	Info

## Change Control History

VERSION	DATE	AUTHOR	DESCRIPTION	PARAGRAPHS
00_01	08-Dec-03	W. Van Acker	Original Issue	
00_02	09-Jan-04	W. Van Acker	Adding first level of test scenarios	
01_10	14-Jan-04	W. Van Acker	First official issue	
01_20	02-Feb-04	W. Van Acker	Revision based on workshop feedback	
01_21	20-Feb-04	W. Van Acker	Update test ids and introduction of test data and member state simulator	
01_22	23-Feb-04	W. Van Acker	Adding	
01_30	13-May-04	W. Van Acker	Adding information about the Pilot Test Phase	
01_40	02-Aug-04	F. Silvestre	Update the url of the central TACHOnet test system	

## Document information

CREATION DATE:	15-May-04
FILENAME:	TCN-TestPlan-01_40-EN.doc
LOCATION:	
NUMBER OF PAGES:	127

## CONTENTS

Changes .....	4
Chapter 1: Introduction .....	6
<b>Chapter 2: Requirements for Test.....</b>	<b>8</b>
Overview .....	8
List of Functional Requirements .....	9
List of Non-Functional Requirements.....	12
<b>Chapter 3: Test Strategy .....</b>	<b>13</b>
Overview .....	13
Test Types – Level of Importance.....	14
Test Types – Techniques.....	19
Test Tools.....	27
<b>Chapter 4: Test Organization.....</b>	<b>28</b>
Overview .....	28
Resources and steps.....	29
Test steps within Tachonet test strategy.....	34
Test Execution.....	38
Test Reporting.....	42
Test Data and Member State Simulation.....	44
CiA Client Simulator.....	47
Pilot Test Phase.....	58
<b>Chapter 5: Test Scenarios .....</b>	<b>75</b>
Overview .....	75
Logon .....	76
Get Phonex Search Keys.....	79
Get US/Ascii Transliteration.....	82
Get US/Ascii Transliteration, continued .....	83
Check driver’s issued card .....	85
Check tachograph card status.....	93
Check tachograph card status, continued .....	96
Declaration of card status modification.....	100
Declaration of card status modification, continued.....	103
Send Card/Driving license assignment.....	108
Create Card – First Issue .....	114
Create Card – First Issue, continued .....	116
Card Renewal .....	118
Card Exchange .....	119
Card Exchange, continued.....	121
Card Replacement .....	122
Browse Statistics .....	123
Manage Users.....	125
Non-Functional Testing.....	126

# Changes

**Summary of changes**      Version 1.10 to 1.20

Page	Map / Block text	Description of the changes
p.31	Test Support	Update
p.32	Test Page	Update
p.37	Connectivity	Update
p.42-85	Test Cases, Test Scenarios	Revision of test cases, test scenarios and test steps based on feedback received during workshop of 22/01/2004. Added graphs to better place the test cases.

**Summary of changes**      Version 1.20 to 1.22

Page	Map / Block text	Description of the changes
p.37	Connectivity	Update, complete list of urls to use
p.43-44	Test Data & Member State Simulator	New section
p.45-55	CiA Client Simulator	New section
p.67	Test Scenarios	Update
p.68	Remark on Test Ids	New block
p.75	Test Scenarios	Update
p.76	Remark on Test Ids	New block
p.81	Test Scenarios	Update
p.82	Remark on Test Ids	New block
p.89	Test Scenarios	Update
p.89	Remark on Test Ids	New block
p.103	Browse Statistics	Update section
p.105	Manage Users	Update section
p.106	Non-Functional Testing	New section

**Summary of changes**      Version 1.22 to 1.30

Page	Map / Block text	Description of the changes
p.37	Test Phases	Update dates
p.57-73	Pilot Test Phase	New section

*Continued on next page*

## Changes, Continued

---

**Summary of changes**      Version 1.30 to 1.40

<b>Page</b>	<b>Map / Block text</b>	<b>Description of the changes</b>
p.39	Test execution	Replace <a href="http://tcn-test.tren.eu-admin.net/">http://tcn-test.tren.eu-admin.net/</a> by <a href="https://webgate.cec.eu-admin.net/tachonet/test/">https://webgate.cec.eu-admin.net/tachonet/test/</a>
p.48-55	CIA Client Simulator	Replace <a href="http://tcn-test.tren.eu-admin.net/">http://tcn-test.tren.eu-admin.net/</a> by <a href="https://webgate.cec.eu-admin.net/tachonet/test/">https://webgate.cec.eu-admin.net/tachonet/test/</a>

---

# Chapter 1: Introduction

---

## Purpose

This Test Plan document for the TACHOnet project supports the following objectives:

- Identify the functional and non-functional requirements as targets for testing
  - Recommend and describe the testing strategies to be employed
  - Identify the required resources
  - Recommend and describe the test organization
  - Provide an approach to test and bug reporting
  - Present a list of test scenarios to execute
- 

## Background

The TACHOnet project final objective is to create a telematics network aiming at facilitating the data exchange between national administrations in charge of the issuing of the tachographs cards, as stated in Council Regulation (EEC) n° 3821/85 amended by Council Regulation (EC) n° 2135/98.

Council Regulation (EEC) no 3821/85 provides for the installation and use of the tachograph for the enforcement of driving time and rest periods of professional drivers in the field of road transport.

The Regulation has been amended by Council Regulation (EC) No 2135/98 that introduced the new digital recording equipment and personal smart cards for drivers. The driver card allows for the identification of drivers when they start their journey and for the recording of their activities. A key element of Regulation (EC) No 2135/98 is to guarantee that that a driver holds only one card.

The individual Member States where drivers have their normal residence are competent to issue the cards. The competent national authority must be able to check that only one card is issued per driver. To avoid a driver holding cards from other Member States such a check should not only be carried out by the own Member States' authority, but also by the competent authorities of other Member States.

In order to guarantee a reliable system of checking the issuing of unique driver cards between Member States, it was felt necessary to have an appropriate telematics network.

---

## Test Justification

Using XML messages over an asynchronous line standardizes the requesting and obtaining of card status and driver/card information. The existing XML Message Specification demands the creation of *unit* and *integration testing*.

Because the solution requires an asynchronous exchange of data, we will include a set of *performance profiling* and *stress tests*.

The process of requesting the card status, the creation of a card by renewal or through loss and the changing of a card status demand the introduction of *function* and *business cycle testing*.

---

*Continued on next page*



---

## Scope

To better position the test plan some definitions are in order. The test discipline distinguishes two major test techniques each having their own merit and purpose. Each development project will require one or both of the following techniques:

White box testing is concerned with testing the software product; it cannot guarantee that the complete specification has been implemented. White box testing is testing against the implementation and will indicate if the implementation is faulty. Test types that fall in this category are unit and integration tests.

Black box testing is concerned with testing the specification; it cannot guarantee that all parts of the implementation have been tested. Black box testing is testing against the specification and will indicate that part of the specification has not been fulfilled. People often speak here of acceptance testing. Test types that fall in this category are function and user interface tests.

We require both approaches often referred to as gray box testing, starting with white box testing early on in the Construction Phase and later initiate black box testing at the end of the Construction Phase and during the Transition Phase. There will be a gradual shift from quality control (of the architecture) towards quality assurance (of the overall solution).

---

## Risks

- In order to correctly perform the testing, the development lifecycle should foresee a mechanism of timely builds (and version control) and a test environment isolated from the development and production environment.
- Deployment of produced units and modules prior to integration and testing require a serious understanding of the architecture.
- Deployment of final and intermediate solutions at the customers' site requires a serious understanding of the customers' work environment in terms of hardware, software and network topology. So configuration and installation testing is needed.

---

## Constraints

- The major constraint is time. The smaller the timeframe for testing the higher the risk of missing (implemented) functionalities that jeopardizes acceptance of the solution.
  - The sooner we start with unit and integration testing the more control we gain over the solution and its behavior.
  - The test plan provides only those cases that cover the central system and the interaction between central system and a local (member state) implementation. It does not provide scenarios for testing the local implementation.
-

## Chapter 2: Requirements for Test

### Overview

---

#### Introduction

This chapter describes the different requirements (functional and non-functional) identified as targets for testing and classified according the different types of test.

These lists represent *what* will be tested (the *how* will be explained in the next chapter).

---

#### Contents

This chapter contains the following topics:

Topic	See Page
List of Functional Requirements	8
List of Non-functional Requirements	11

---

# List of Functional Requirements

---

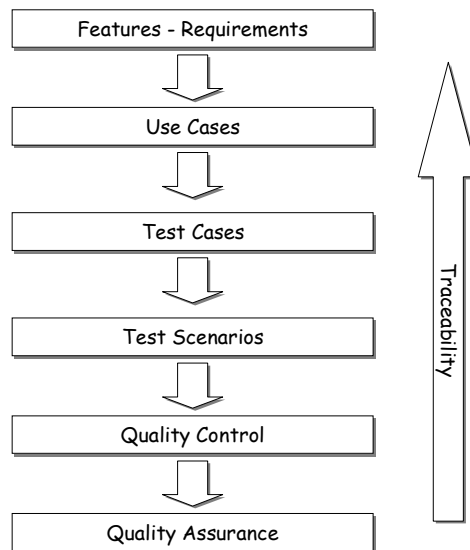
## Definition

Functional requirements specify actions that a system must be able to perform, without taking physical constraints into consideration. Functional requirements thus specify the input and output behaviour of a system. Therefore, these requirements are obvious and mandatory candidates for testing.

---

## Traceability

Each to be developed solution must respond to the stakeholders' requests. Each requirement, functional or non-functional, must be analysed, designed and tested. The following diagram clarifies the connection between requirements and test cases. This aspect is also known as traceability and will permit us to trace back a particular test scenario back to its origin, the requirement.



---

*Continued on next page*

## List of Functional Requirements, continued

**Abbreviations** Although the different test types will be explained more in detail in the next chapter we will introduce some abbreviations used in the following tables.

- FT: Function testing
- BCT: Business Cycle testing
- UIT: User Interface or Usability testing
- PP: Performance profiling
- LSV: Load, Stress and Volume testing
- SAT: Security & Access Control Testing
- FRT: Failover & Recovery testing
- CIT: Configuration & Installation testing
- UT: Unit testing
- IT: Integration testing
- DIT: Data & Data Integrity testing

### List of Requirements

The table below lists all the functional requirements being translated into use cases and business use cases, that have been identified as targets for testing along with the type(s) of test to carry out:

Functional Requirement Id	Use Case/Business Use Case	Type of test
FUN12	Logon CiA Application to TCN Central System	FT/SAT
<i>FUN09/FUN13</i>	<i>Create DriverCard</i>	<i>FT/UT/IT</i>
FUN09	Update CardStatus	FT/UT/IT
FUN11	Read CardStatus	FT/UIT
<i>FUN13</i>	<i>Read Driver</i>	<i>FT/UIT</i>
FUN11	Get Phonex Search Keys	FT/UIT/UT
FUN11	Get US/ASCII transliteration	FT/UIT/UT
<i>FUN09</i>	<i>Search Driver</i>	<i>FT/UT</i>
<i>FUN09/FUN02</i>	<i>Search DriverCard</i>	<i>FT/UIT/IT</i>
FUN09/FUN11	Read DriverCard	FT/UIT
FUN01	Update Driver	FT/UIT
FUN09/FUN01	Update DriverCard	FT/UT
FUN03	Browse MS Statistics (°)	FT/UIT
FUN07	Browse TCN Statistics (°)	FT/UIT
FUN05	Monitor message exchange (°)	FT/UIT/IT
FUN08/FUN10	Manage Users (°)	FT/UIT
FUN09	Check driver's issued card	BCT
FUN11	Check tachograph card status	BCT
FUN09	Declaration of card status modification	BCT

Functional Requirement Id	Use Case/Business Use Case	Type of test
FUN01	Send Card/Driving License assignment	BCT
FUN09	Create Card – first issue	BCT
FUN02	Bulk check of issued driver card holders	BCT

---

**Remarks**

- The list of use cases presented deviates from those presented in the Global Business Analysis document in that we have tried to distinguish basic functionalities (covered by FT) and particular transactions (covered by BCT).
  - Those use cases with (°) have to be analyzed out further before test cases can be defined and will not be covered in this version of the test plan.
  - The list of use cases presented here is actually extended with a set of abstract use cases (shown in *italic*) to aid the developers and testers in performing their quality control. The above list is rather used for (customer) acceptance testing.
-

## List of Non-Functional Requirements

---

### Definition

Non-functional requirements describes the software requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. Regarding testing activities, the lists of these non-functional requirements are being covered.

---

### List of Requirements

The following table outlines the non functional requirements and system constraints (architectural factors) that have been identified as targets for testing along with the type(s) of test to carry out:

Requirement Id	Description	Type of test
USA-01	The system must guide users through an interface based on end user concepts.	UIT
REL-03	The system must give stable and reproducible results.	PP/FRT
PER-02	There will be no restriction in time or place for the use of the software built from the specifications.	CIT
PER-04	The system will be designed so that background tasks can continue while the user performs foreground tasks.	PP/LSV
PER-05	The system will be used 24x7 by operators under pressure to produce results rapidly. The system must respond rapidly to user requests irrespective of any background tasks. Such high-availability (24x7) is also required from the Member States systems to ensure acceptable response time (less than 1 minute) to enforcement authorities requests.	FRT
SUP-02	The system must be maintainable and extensible.	FT/PP
SUP-05	The system must provide solutions/rules regarding data encoding problems such as supporting different character sets, ...	FT/UIT

---

### Remarks

- The non-functional requirements mentioned above will not be translated in a whole set of test cases or test scenarios, but they are rather presented here as features that can be tested and verified by even the customer, once again allowing customer acceptance.
  - For a complete list and description of all non-functional requirements please refer to the [Global Business Analysis](#) document.
-

## Chapter 3: Test Strategy

### Overview

---

#### Introduction

This chapter describes the test types that will be performed according to their importance in the project, *why* they are performed (or not) and *how* they are performed in order to obtain a satisfying result.

---

#### Contents

This chapter contains the following topics:

Topic	See Page
Test Types – Level of Importance	13
Test Types - Techniques	18
Test Tools	26

---

## Test Types – Level of Importance

---

### Introduction

In the previous chapter we added a first indication of test type besides each functional and non-functional requirement without explaining what exactly is being done or measured when performing the test.

A whole list of abbreviations was provided without any indication if that type of test would be performed or was required.

In this section we justify which test types are necessary and why, in order of importance.

---

### Unit Testing

The Software Architecture Document clearly describes the approach that will be followed by the developers. The technical choices and object-oriented approach, clearly promotes and supports the execution of unit tests.

The designers and architects have identified the building blocks by means of class models. Each building block is responsible for performing a number of actions based on incoming parameters. The purpose of a unit test is to guarantee that the class (or block) works as intended by the developer.

Besides the creation of test classes, special error-handling classes can be developed to clearly describe what went wrong in a building block.

It is up to the developers to create and execute the unit tests.

---

### Integration Testing

During development, and more specific at the end of an iteration cycle, all developed application blocks have to be integrated into a library, framework, namespace or application. Any integration error will result in a malfunctioning of one or more function test.

A function test, being a particular use case realization will often consist of a sequence of function calls or methods. Methods that are defined within a building block and that send their results as input to other building blocks. So the execution of a particular (user) activity is nothing more than a sequence of building block activations. As such it is imperative that unit tests are followed by integration tests and function tests.

Special error-handling classes can be extended to describe where something went wrong in a building block or library.

---

*Continued on next page*



## Test Types – Level of Importance, continued

---

### **Function Testing**

The acceptance of the application by the end-users stands or falls with the correct functioning of all required features. The features captured by the system analysts during meetings and workshops have been converted into use cases, use case diagrams and detailed analysis documents (like the Global Business Analysis) allowing the test analyst to deduce the according test cases.

The results of several black box test cycles will indicate if the specification is fulfilled.

The more thorough unit and integration tests are the less function test cycles will be needed to guarantee a good functioning of the application.

---

### **User Interface Testing**

Also known as Usability testing, the focus lies on ease of use of the different features offered (through a user interface). The basis being talks between client, analysts and HCI designers, is reflected into a User Interface Specification which itself resulted into a web site prototype.

Such a prototype is often static in nature but serves the purpose of convincing the client that the final solution will behave as he wants it to be and serves the developers in finalizing their framework on how to capture, convert and pass through user input to the underlying business objects.

During those tests we verify the user interface specs, the validation of field contents and the interaction between fields and buttons on the screen.

The level of importance is considered low because not all requirements need a user interface.

---

### **Business Cycle Testing**

The fact that the major part of the to-be developed features are covering the asynchronous behavior of the system (transactions) and that we are dealing with a system that centralizes messaging between applications over different European countries, business cycle testing is quite important.

The line between function testing and business cycle testing is often narrow and this is reflected in the list of use cases presented earlier.

Business Cycle tests stand towards function tests like integration tests to unit tests.

---

*Continued on next page*

## Test Types – Level of Importance, continued

---

### **Security & Access Control Testing**

When looking at the list of actors defined in the use case model one can see that access rights play an important role when providing features and data to the end-user. The amount of actors will influence the amount of test scenarios per test case.

Even when not using the web interface, systems exchanging information need to authenticate themselves. Because the application should adhere to the TESTA-II network procedures and security requirements described in a separate document, we will not invent new test cases.

---

### **Performance Profiling**

As soon as integration tests are executed and provide positive results, we should start looking at application performance. A separate document has provided estimation on the traffic to expect once TACHOnet becomes operational. Based on the relative amount of messages that will be exchanged, performance profiling is important.

A number of tests should provide feedback so that optimization of code, process and exchange of data can be initiated.

---

### **Data & Database Integrity Testing**

A most crucial aspect of the application is that the XML messages are correct first in form and then in content. If there is no message validation according to XML standards and defined XML schema the receiving end cannot deal with the incoming data (request). This also works the other way around, Data Requester and Data Provider need to follow the XML messaging specifications to the letter. How the messages are created and decomposed is up to the implementers.

Since a limited section of each message transmitted is being stored by the central system, this aspect is considered low in priority.

---

### **Load & Stress & Volume Testing**

These kind of tests require a clear view on the traffic that TACHOnet will generate over time in order to allow the different members states to simulate incoming data requests in terms of frequency and volume and to allow the central system to have an indication on the message load being exchanged.

The latter will be combined with the performance profiling.

---

*Continued on next page*

## Test Types – Level of Importance, continued

### Failover & Recovery Testing

All systems are susceptible to breakdown or temporarily inactivity, especially interconnected systems. A breakdown of one of the local systems should not cause a breakdown of the overall network. Replication and backup systems should be anticipated.

This aspect is reinforced due to the amount of messages that will/can be exchanged during a day.

### Installation & Configuration testing

Installation and configuration testing is dependant on how the solution is implemented by the different members states. The test scenarios should be written in light of their infrastructure.

### Summary

The following cross-reference table should give an indication of the importance of each test type, when to perform them and by whom. It is clear that each member state, implementing its own solution is free to ignore the contents of this table.

Test Type	Construction Phase – I1	Construction Phase – I2	Transition Phase	Member State	TCN Central System
UT	High	Medium	Low	X	X
IT	Medium	High	Low	X	X
FT	Low	High	Medium	X	X
UIT	Low	Low	Medium	X	
BCT	Low	Medium	High	X	X
SAT	Medium	Medium	High		X
PP	Low	Medium	High	X	X
DIT	Low	Low	Low	X	X
LSV	Low	Medium	Medium	X	X
FRT	Low	Medium	Medium		X
CIT	Low	Low	Low	X	X

### Remark

Some sources and reference works might introduce other types of tests or might group them a bit differently. We have followed the test classification as close as possible as it is described in the Unified Process implemented by Rational©.

*Continued on next page*

## Test Types – Level of Importance, continued

---

### **Regression Testing**

Regression testing is not really another type of test but rather a particular test strategy in that previously executed tests are being re-executed against a new version of the application or deliverable.

Regression testing ensures that the quality of the target has not regressed, moved backward while new capabilities have been added.

The TCN project plan clearly indicates that several iterations in the Construction Phase are anticipated which invites the introduction of regression testing.

In this project we strongly suggest to the member states to perform regression testing, even within each of the proposed test phases

---

### **Acceptance Testing**

The purpose of acceptance testing is to test the complete application by the end-users (or representatives) in order to determine the application's readiness for deployment.

Acceptance testing is best performed on a test environment mirroring the client's production environment.

The tests should be a subset of all available test scenarios and should reflect the stakeholders' interest in the project meaning all requested features and initial requirements should be met as close as possible.

In fact, acceptance testing is a kind of contract between both parties. It provides a guarantee to the customer that what is delivered complies with what was initially requested and on the other hand is a means for the solution provider to block any unrealistic (change) requests from the customer.

As a guideline we suggest the notion of acceptance testing between member states and central authority due to the centralized nature of the architecture. In other words, do not add a new member state to the TCN system if it does not comply with the specifications.

---

## Test Types – Techniques

---

### Introduction

Having assigned an order of importance on the different test types, we can start with creating the test scenarios. For those inexperienced with the test discipline we provide some extra information on how to perform the different test types and on how to interpret the test results.

If the reader has already experience with testing he can skip this part.

---

### Function Testing

Function testing focus on any requirement for test that can be traced directly to use cases or business functions. This type of testing is based upon black box techniques; that is verifying the application and its internal processes by interacting with the application via the graphical user interface and analyzing the output or results.

Test Objective:	Ensure functionality.
Technique:	Perform a sequence of actions on the interface using valid and invalid data to verify expected results when data is valid, and the appropriate error message when invalid data is entered. Also verify if each business rule is properly applied.
Completion Criteria:	All planned tests have been executed. All identified defects have been addressed.
Special Considerations:	Each new requirement or changed feature results in a new test case and underlying test scenarios.

---

### Business Cycle Testing

Business Cycle testing should emulate those activities (or sequence of activities) that are performed over a certain period of time. The asynchronous aspect of the SSN solution is one of the targets of those tests. Time is a keyword transaction another.

Test Objective:	Ensure correct and timely flow of activities.
Technique:	The appropriate function test cases are modified (test scenario) to reflect the asynchronous or transactional aspect. Timing-out and loss of connection situations have to be taking into account.  All functions that occur on a periodic schedule will be executed at the appropriate time. All date-sensitive functions will be executed using valid and invalid dates.  Testing will include using valid and invalid data to verify expected results when using valid data and the error or warning messages when using invalid data. Also verify if each business rule is properly applied.
Completion Criteria:	All planned tests have been executed. All identified defects have been addressed.
Special Considerations:	System dates and events may require special support activities. Business model/flow charts are required to identify appropriate test requirements and procedures.

---

*Continued on next page*

## Test Types – Techniques, continued

### User Interface Testing

User Interface (UI) testing verifies a user’s interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Test Objective:	Ensure that navigation through the application reflects business needs and requirements, this includes window-to-window, field-to-field and access methods.
Technique:	The UI testing will be executed during function testing. This implies that the test scenarios derived from the test case will verify the correct interface navigation and change of object states (mandatory fields, dropdown list, locking, etc.).
Completion Criteria:	Each window successfully verified.
Special Considerations:	None.

### Unit Testing

Unit testing consists of a set of coded use cases that document what a class does, given a controlled set of inputs. It provides a design for a class that the test will exercise, allowing the developer to focus on small chunks of code. Writing tests first lets the developer know when the class is complete.

Test Objective:	Ensure that the classes being developed are functioning as intended by the developers. That a set of given inputs generates the expected result.
Technique:	For each class being developed a corresponding test class is developed with a pre-defined constructor and a pre-defined result. The test class/case is executed and the result is compared with the expected result.
Completion Criteria:	If the test case result equals the expected result the class is good for implementation and integration.
Special Considerations:	None.

*Continued on next page*

## Test Types – Techniques, continued

### Integration Testing

Integration testing is a continuation of the unit tests. Typically classes are grouped into packages or components bundling a set of functionalities used in unison or having a common purpose. These can be tested by means of a test suite running a set of test cases in sequence.

Test Objective:	Ensure that a component or package grouping classes and as such functionalities classes perform as intended by the developers. That a set of given inputs generates the expected result.
Technique:	For each package or component a set of test classes or test suite is developed. The test suite is executed and the result is compared with the expected result. The sequence of test cases executed should mimic a possible function test or section of a function test.
Completion Criteria:	The test suite result equals the expected result.
Special Considerations:	Using sequence diagrams could aid in the implementation of integration tests.

### Security & Access Control Testing

Security and Access Control Testing focus on two key areas of security:

- Application-level security, including access to the Data or Business Functions
- System-level Security, including logging into or remote access to the system.

Application-level security ensures that, based upon the desired security, actors are restricted to specific functions or use cases, or are limited in the data that is available to them. System-level security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.

Test Objective:	Application-level Security: Ensure that an actor can access only those functions or data for which they have permission.  System-level Security: Ensure that an actor with a specific set of permissions actually have access to the application.
Technique:	Application-level Security: Identify and list each user type and the functions or data each user type has permissions for. Create test scenarios per test case for each user type and verify that functions and data are available or denied.  System-level Access: See Special Considerations below
Completion Criteria:	For each known actor type the appropriate function or data are available, and all transactions function as expected.
Special Considerations:	Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.

*Continued on next page*

## Test Types – Techniques, continued

### Failover & Recovery Testing

Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without loss of data or transactions.

Recovery testing is a test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures or invalid database pointers. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

Test Objective:	<p>Verify that recovery processes (manual or automated) properly restore the database, applications, and system to a desired, known, state.</p> <p>The following types of conditions (power interruption client and server, communication interruption, invalid database pointer or key, corrupted data element) are to be included in the testing.</p>
Technique:	<p>Reuse of function or business cycle tests. Initiate one of the above mentioned conditions once the starting point of the scenario is reached.</p> <p>Once the above conditions or simulated conditions are achieved, additional transactions should be executed and upon reaching this second test point state, recovery procedures should be invoked.</p> <p>Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools).</p>
Completion Criteria:	<p>In all cases above, the application, database, and system should, upon completion of recovery procedures, return to a known, desirable state. This state includes data corruption limited to the known corrupted fields, pointers or keys, and reports indicating the processes or transactions that were not completed due to interruptions.</p>
Special Considerations:	<p>Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Alternative methods, such as diagnostic software tools may be required.</p> <p>These tests should be run after hours or on an isolated machine.</p>

*Continued on next page*



## Test Types – Techniques, continued

### Configuration Testing

Configuration testing verifies the operation of the application on different software and hardware configurations.

Test Objective:	Verify that the application functions properly on the required hardware and software configurations.
Technique:	Reuse of Function Test scripts. Open and close various non-application related software, such as the Microsoft applications as part of the test or prior to the start of the test.
Completion Criteria:	For each combination of the application and non-application software, all transactions are successfully completed without failure.
Special Considerations:	What non-application software is needed, is available, and is accessible on the desktop?

### Installation Testing

Installation testing has two purposes. The first is to insure that the software can be installed under different conditions—such as a new installation, an upgrade, and a complete or custom installation—under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, etc.

The second purpose is to verify that, once installed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing.

Test Objective:	Verify that the application properly installs onto each required hardware configuration as a new installation, an as an upgrade, testing the replacement of older versions of files.
Technique:	Launch or perform installation.
Completion Criteria:	The application executes successfully without failure.
Special Considerations:	Selection of those function tests that guarantee a correct installation.

*Continued on next page*

## Test Types – Techniques, continued

### Performance Profiling

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated.

The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune a target-of-test's performance behaviors as a function of conditions such as workload or hardware configurations.

Test Objective:	Verify performance behaviors for designated transactions or business functions under the following conditions: <ul style="list-style-type: none"> <li>• normal anticipated workload</li> <li>• anticipated worst case workload</li> </ul>
Technique:	Use Test Procedures developed for Function or Business Cycle Testing.  Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs.  Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see Special Considerations below).
Completion Criteria:	Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction.
Special Considerations:	Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  The databases used for Performance Testing should be either actual size or scaled equally.

*Continued on next page*

## Test Types – Techniques, continued

### Data & Database Integrity Testing

The databases and the database processes should be tested as a subsystem within the system. These subsystems should be tested without the target-of-test’s User Interface as the interface to the data.

Additional research into the DataBase Management System (DBMS) needs to be performed to identify the tools and techniques that may exist to support the testing identified below.

Test Objective:	Ensure database access methods and processes function properly and without data corruption.
Technique:	Invoke each database access method and process, seeding each with valid and invalid data or requests for data.  Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons
Completion Criteria:	All database access methods and processes function as designed and without any data corruption.
Special Considerations:	Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases.

### Load Testing

Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and ability of the target-of-test to continue to function properly under these different workloads

Test Objective:	Verify performance behavior time for designated transactions or business cases under varying workload conditions.
Technique:	Use tests developed for Function or Business Cycle Testing.  Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.
Completion Criteria:	Multiple transactions or multiple users: Successful completion of the tests without any failures and within acceptable time allocation.
Special Considerations:	Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.

*Continued on next page*

## Test Types – Techniques, continued

### Stress Testing

Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions.

Stress testing can also be used to identify the peak workload the target-of-test can handle.

Test Objective:	Verify that the target-of-test functions properly and without error under the following stress conditions: <ul style="list-style-type: none"> <li>• little or no memory available on the server</li> <li>• maximum actual or physically capable number of clients connected or simulated</li> <li>• multiple users performing the same transactions against the same data or accounts</li> </ul>
Technique:	Use tests developed for Performance Profiling or Load Testing.
Completion Criteria:	All planned tests are executed and specified system limits are reached or exceeded without the software failing or conditions under which system failure occurs is outside of the specified conditions.
Special Considerations:	Stressing the network may require network tools to load the network with messages or packets.

### Volume Testing

Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period.

Test Objective:	Verify that the target-of-test successfully functions under the following high volume scenarios: <ul style="list-style-type: none"> <li>• Maximum (actual or physically- capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period.</li> <li>• Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.</li> </ul>
Technique:	Use tests developed for Performance Profiling or Load Testing.
Completion Criteria:	All planned tests have been executed and specified system limits are reached or exceeded without the software or software failing.
Special Considerations:	None

## Test Tools

---

**Introduction** The test strategy can be described as a sequence of activities to be performed by specific people each using one or more tools to create and maintain their work.

---

**Tools Used** The following tools will be employed for this project:

Activity	Tool	Vendor
Test Management	Office 2000	Microsoft
Defect Tracking	Mantis + mail server	Open Source
Project Management	Project 2002	Microsoft
Test Analysis & Design	Rational Rose + Excel 2000	Rational
Test Implementation	JUnit, Cactus, JUnitPerf	Open Source
Test Evaluation	Excel 2002	Microsoft

---

**Proposed Tools** The following tools could aid and automate the test process:

Activity	Tool	Vendor
Test Management	TestDirector	Mercury Interactive
Test Implementation	QuickTest Pro	Mercury Interactive

---

## Chapter 4: Test Organization

### Overview

---

**Introduction** Explaining how the test types are to be executed is done in the previous section. This chapter will focus on the *how* to organize the test execution, independent of test type.

---

**Contents** This chapter contains the following topics:

Topic	See Page
Resources and steps	28
Test steps within Tachonet test strategy	33
Test Execution	37
Test Reporting	40
Test Data and Member State Simulation	43
CiA Client Simulator	45
Pilot Test Phase	56

---

## Resources and steps

### Staffing

The following table shows the staffing assumptions for the project, related to the test discipline. The table is a proposition for both the member states and the central Tachonet system because development and tests are being executed at several sites.

Role in Test Discipline	Min. Resources	Covered by (MS, central system, both)	Specific Responsibilities or Comments
Test Manager, Test Project Manager	1	Both	Provides management oversight. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>provide technical direction</li> <li>acquire appropriate resources</li> <li>provide management reporting</li> </ul>
Test Designer	1	Both	Identifies, prioritizes, and implements test cases. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>generate test plan</li> <li>generate test model</li> <li>evaluate effectiveness of test effort</li> </ul>
Tester	2	Both	Executes the tests. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>execute tests</li> <li>log results</li> <li>recover from errors</li> <li>document change requests</li> </ul>
Test System Administrator	1	Both	Ensures test environment and assets are managed and maintained. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>administer test management system</li> <li>install and manage worker access to test systems</li> </ul>
Dbase Administrator, Dbase Manager	1	Both	Ensures test data (database) environment and assets are managed and maintained. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>administer test data (database)</li> </ul>
Designer (°)	1	Both	Identifies and defines the operations, attributes, and associations of the test classes. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>identifies and defines the test class(es)</li> <li>identifies and defines the test packages</li> </ul>
Implementer (°)	2	Both	Implements and unit tests the test classes and test packages. <u>Responsibilities:</u> <ul style="list-style-type: none"> <li>creates the test classes and packages implemented in the test model</li> </ul>
(°) Designer and Implementer are considered to cover Unit and Integration test cases for the Member State CiA implementations.			

Continued on next page

## Resources and steps, continued

---

### Remarks

- The roles described in the previous table should be seen as activities someone can execute during the test discipline and the overall project. This implies that one physical person can take one or more responsibilities.
  - Designers and developers should aid the system engineers in defining installation and configuration tests, not executing them. The latter are also responsible for the creation and execution of performance profiling, load tests and failover & recovery test.
- 

### Test Preparation

Executing tests is not simple sitting behind a screen and start clicking at random on a presented interface. In order to obtain valuable test results some preparation in terms of people and material is required. The next list provides a kind of checklist of actions to perform before testing.

- Assign roles and responsibilities (°)
- Capture team members coordinates and assign contact persons (internal and external)
- Set up test environment (°)
- Make up list of test cases and test scenarios available
- Make up list of test cases and test scenarios that will be executed, and group them into a test cycle (°)
- Provide matching test data when testing the XML interface
- Prepare test database, and database connection, introduce test data
- Contact TCN Central System to initiate testing.

(°) will be covered in more detail in the following paragraphs

---

*Continued on next page*



## Resources and steps, continued

---

### Test Roles - Staffing

During each test phase (and test cycle) the member state should inform the TCN Central System:

- Who the test manager is, and if he acts as SPOC (provide name/email address/ phone)
- Who the testers are (provide name/email address/phone) (relevant only for test result reporting and bug reports when extra information is required)
- Who the test system admin is in case test environment problems occur (provide name/email address/phone)
- When the above people are available (working hours)

During each test phase (and test cycle) the TCN Central System should inform the member state participating:

- Who the contact persons are for DGTren / Getronics (provide email address/ phone) in terms of organizational follow-up
  - Who the contact persons are for hardware issues
  - Who the contact persons are for function testing support (scenario execution, checkpoints, result interpretation) (provide name/email address/phone)
  - When the above people are available (working hours)
- 

### Test Cycle

As will be indicated in the next section of the document, step 2 of the test strategy consists of 3 clearly marked test phases. Timeframes of each 1 calendar month, the test phases can be split up into test cycles.

Each test cycle contains 4 consecutive working days filled in as follows for a **Member State**:

Day1: start of the test cycle – perform test scenarios

Day2: digest test results and bugs detected – send out bug report to TCN CS

Day3: request new test cycle from TCN CS

Day4: receive acknowledgement for next test cycle – end of current test cycle

The test cycle for the **TCN Central system** is organized as follows:

Day1: start of test cycle with MS – perform test scenarios

Day2: digest own test results, bugs detected, enter into central bug tracking system

Day3: digest incoming bug reports, treat new test cycle requests

Day4: make updated central bug tracking system available to MS – send out acknowledgement for next test cycle – end of current test cycle with MS

---

*Continued on next page*

## Resources and steps, continued

---

### Remarks

- Even if a MS is not able to execute its complete list of tests, the testing ends after the first day of the testing activity. The remaining test scenarios have to be executed in the next test cycle.
  - With an average of 20 working days during a test phase, each participant should have the opportunity to test at least twice with a maximum of up to five test cycles.
  - We anticipate that the number of test cycles will diminish in test phase 3 where the number of test scenarios to execute will be the largest. In that case we might shift towards a test cycle of 5 or 6 days. We leave it up to the member states to inform the TCN Central System at least a week in advance of test phase 3 if they want to change the test regime.
- 

### Test Environment

Each member state should have besides its XML interface implementation, a test environment. This requires the existence of a test interface allowing the tester to select the contents of the XML message to generate by the XML interface implementation. In other words, providing the member states with the knowledge of which XML message should be sent out to the central system is insufficient to test the local implementation. This only allows us to test the correct functioning of the central system.

---

### Test Support

During a Test Phase problems could occur of diverse nature like connection issues, functional misunderstandings, organizational difficulties and others. In order to minimize and resolve those problems we will provide a helpdesk phone and mailbox:

TCN Helpdesk phone: +32 (0) 2 229 9040 from 9am (GMT+1) to 12am (GMT+1) and 1pm (GMT+1) to 5pm (GMT+1), Monday till Friday.

TCN Helpdesk newsgroup: An extra newsgroup will be made available in the Circa directory under the “TREN: The digital tachograph” section to allow member states to submit their questions and to allow us to provide answers and announce new releases or functionalities made available for testing purposes. The newsgroup will be called `europa.tren.digtacho.testphases`.

TCN mailbox: [be.test.tachonet@Getronics.com](mailto:be.test.tachonet@Getronics.com) will be used for requesting test participation, support during tests and test reporting from and to a member state.

---

### Test Initiation

As soon as the TCN Central System has confirmed the test cycles and its start date and time, the following steps have to be performed:

- Execute test cycle
- Capture test results, store results, generate report
- Describe bugs encountered

*See section Test Execution for details on test cycle execution and result capturing.*

---

*Continued on next page*

## Resources and steps, Continued

---

### Test Page

Before any actual XML exchange can be done, each member state must test its access with the central system. The TCN Central System will provide some urls consisting of a simple html page. If the XML interface implementation is not capable of obtaining this page then connection between both systems is missing and the test cycle cannot take place.

This page will be provided to the member states before the actual test phase starts allowing them to prepare their test environment (and connection).

Before executing your test scenarios test your TESTA connection through:

<https://webgate.cec.eu-admin.net/tachonet/test/start.html>

---

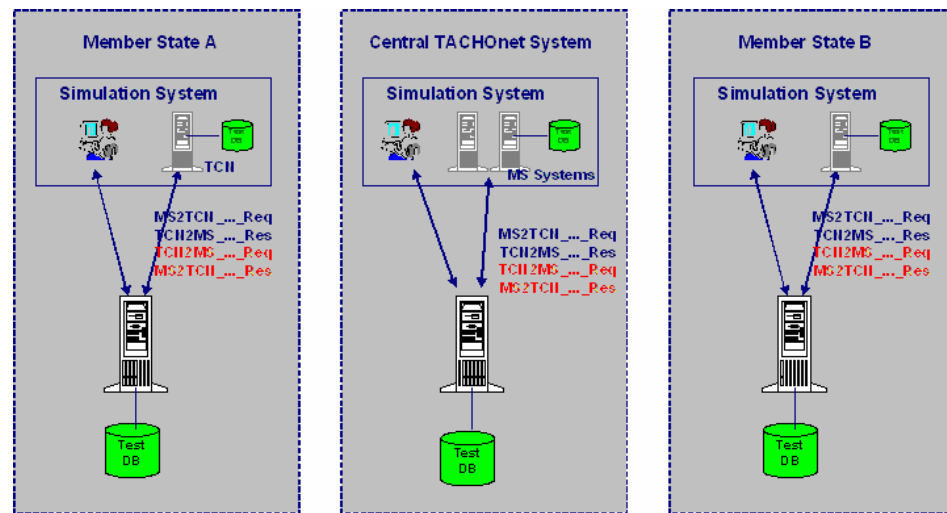
### Test Resolution

As soon as the tests are performed and the test results are known, the following concluding steps are to be done:

- Provide feedback of tests to DG Tren
  - Provide feedback of bugs to local developers
  - Provide feedback of bugs encountered to DG Tren
  - Request new test cycle (if required)
-

## Test steps within Tachonet test strategy

### First step



### Local testing – 1

The first step in the testing procedure consists of executing a number of tests (and test types) locally in each member state. Focus lies on the implementation of the CiA Application and XML interfaces between the MS and the TCN XMS (Tachonet XML messaging system).

Because the connection between MS and TCN XMS does not exist yet at that stage of the project, the member states (MS) will simulate the responses they get from the central system.

### Test Focus – 1

During the local testing, the MS will execute at least the following test types:

- Unit Tests and Integration Test: to ensure their implementation of the architecture
- Data & Data Integrity Tests: to ensure generation and storage of XML messages
- Function Tests: to ensure an initial set of functional requirements
- Security and Access Control Tests: to ensure that all defined actors at the member state side are recognized as such
- Performance Profiling: to ensure that the local architecture can cope with the load and volume of the message exchange.

*Continued on next page*

## Test steps within SafeSeaNet test strategy, continued

---

### Local Testing - 2

Besides the testing done by the member states, the central system has to be tested also. Focus lies on the implementation of the TCN XML interfaces between a MS and the TCN XMS (Safeseanet XML messaging system).

Secondly, the testing of the Phonex Search keys and transliteration mechanisms.

Because the connection between MS and TCN XMS does not exist yet at that stage of the project, the central system will simulate the data requests/responses being send by the MS. The forwarding of data requests must be covered too.

---

### Test Focus – 2

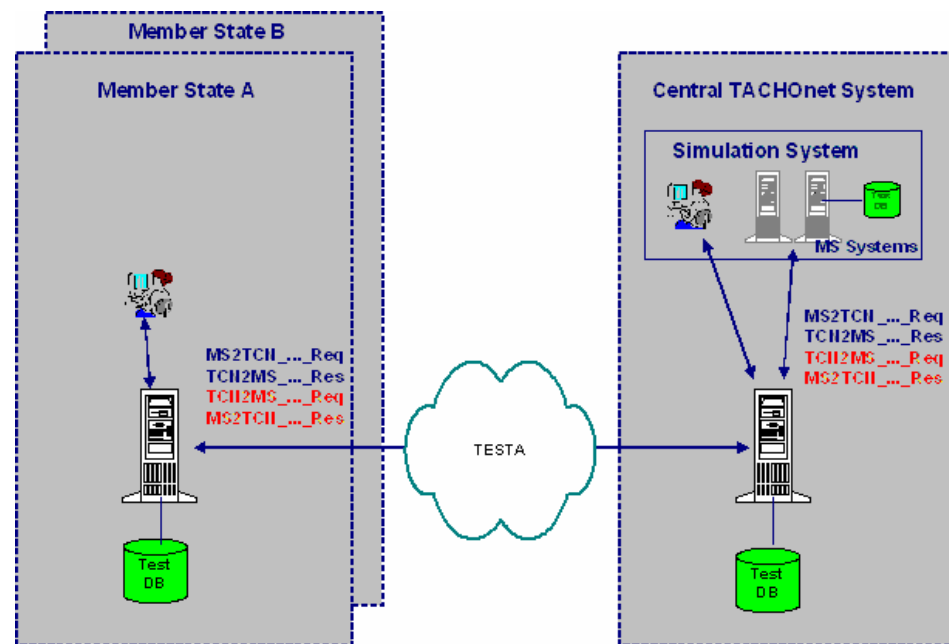
During the local testing, the Central TCN system will execute at least the following test types:

- Function Tests: to ensure an initial set of functional requirements
  - Security and Access Control Tests: to ensure the all defined actors at the central systems side are recognized
  - Performance Profiling: to ensure that the local architecture can cope with the load and volume of the message exchange.
  - Configuration and Integration Tests: to ensure that the developed solution can be installed and deployed at the customer's site.
- 

*Continued on next page*

## Test steps within SafeSeaNet test strategy, continued

### Second step



### TCN Simulation

The second step in the testing procedure consists of executing a number of tests (and test types) between a member state and the actual TCN XMS. Focus lies on the exchange of XML messages between two parties.

Because only one MS is included, the TCN XMS will simulate the responding to data requests from the MS and the TCN XMS will also simulate requests for details addressed/forwarded towards the (one) MS.

### Test Focus

During the second step, the following test types should be executed:

- Data & Data Integrity Tests: to ensure valid and well-formed XML messages
- Function Tests: to ensure an extended set of functional requirements
- Business Cycle Tests: to ensure the exchange of XML messages between a MS and the Central system, mainly the handling of single data requests/responses
- Performance Profiling: to ensure that the overall architecture can cope with the load and volume of the message exchange.
- Stress Testing and Failover & Recovery Tests: to ensure the network and hardware choices are holding up.

*Continued on next page*

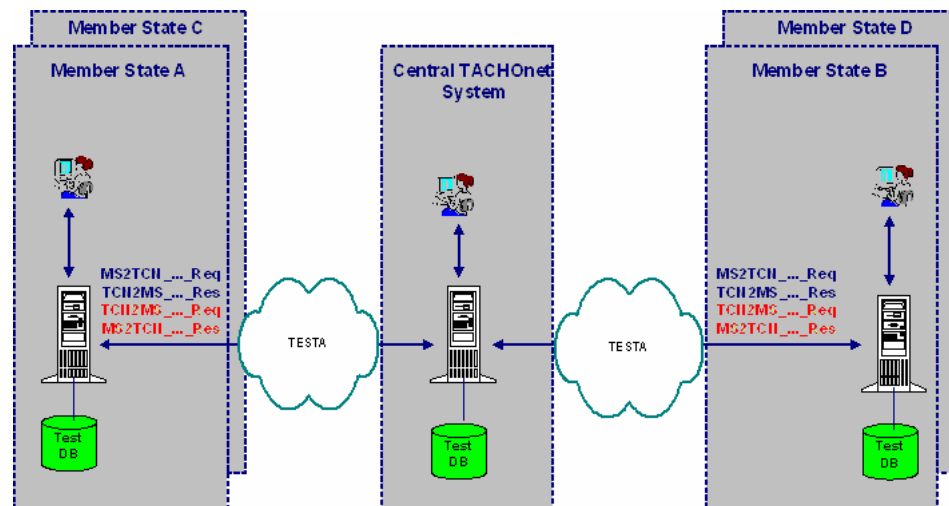
## Test steps within Tachonet test strategy, Continued

### Remark

This second step in the testing process should be conclusive for the correct local implementation for and the integration of a member state in the overall Tachonet network.

The central TCN system should plan during this test phase one or more test cycles where more than one member state is connected in order to do some stress testing.

### Third step



### Pilot Testing

The third step in the testing procedure consists of executing tests (and test types) between member states through/with the actual TCN XMS. Focus lies on the complete exchange of XML messages between (pre-defined) Member states and the Central System.

In this stage of the project it might be useful to have some member states involved which are known to have a large amount of requests in order to test the stability of the overall network.

### Test Focus

During the third step, the following test types should be executed:

- Data & Data Integrity Tests: to ensure valid and well-formed XML messages
- Function Tests: to ensure a complete set of functional requirements
- Business Cycle Tests: to ensure a complete set of XML data exchanges
- Performance Profiling: to ensure that the overall architecture can cope with the load and volume of the message exchange.
- Load, Stress and Volume Tests: to ensure the network and hardware choices are holding up.

## Test Execution

---

### Test Phases

Independent of the previously defined test steps, we have provided a test time line allowing all participants to align their development and test efforts. All participants in the Tachonet project are responsible for their local interface. This includes local testing and deployment. Within the same timeframe development, testing and deployment of the central system is performed. This activity is scheduled from mid Nov 03 till mid May 05.

As soon as both member states and central system are stable and reliable, tests can be initiated to verify if the XML exchange is working as intended. The message exchange is limited between one member state and the central system at a time. This does not exclude a situation where two or more member states would perform tests side by side. This activity called the normal test phase is currently scheduled to take place between mid Feb 2004 and mid May 2005.

A separate activity called pilot testing is currently scheduled between end May 2004 and begin June 2004. During this pilot testing the overall network will be tested. A number of member states will be invited to partake and will act as both data requester and data provider towards another member state being connected. Subsequently, other pilot testing phases should be scheduled between September 2004 till May 2005.

As soon as this part of the architecture is stable and tested we will extend the number of connected participants or simulate them in order to test the actual performance of the network in terms of load and broadcasting ability.

During the test period a helpdesk and mailbox will be available to all member states to aid in the integration and testing of their interface implementation into the overall network. (see Test Support)

---

### Test Readiness

Each member state has to provide feedback on its local implementation of the XML interface in terms of functionalities/use cases developed and ready for test.

Each member state being aware of the planned test phases indicates if they participate in a test phase and what they like to test in terms of interface (XML and/or Web) and use cases.

The central system too has to broadcast the use cases that they have implemented and are available for testing.

This allows the Test Manager to have an overview of the test readiness of each member state and central system.

---

*Continued on next page*



## Test Execution, Continued

### Connectivity

If the connectivity test (see Test Page) have been successful and the member state has received confirmation from the Central System that it can start with its proper testing, it will use one or more of the following urls

The following table lists all TACHOnet urls (given for the test environment - simply replace **tachonet/test** by **tachonet/production** in the production environment) accessible from the Member States:

Url	Description
<a href="https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll">https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll</a>	Official url for sending XML messages to TACHOnet.
<a href="https://webgate.cec.eu-admin.net/tachonet/test/TCNWS/tcnws.aspx?wsdl">https://webgate.cec.eu-admin.net/tachonet/test/TCNWS/tcnws.aspx?wsdl</a>	Official url to get the <b>WSDL</b> file of the <b>TCN Web Services</b> .
<a href="https://webgate.cec.eu-admin.net/tachonet/test/TCNWSConsumer/tcnws.aspx">https://webgate.cec.eu-admin.net/tachonet/test/TCNWSConsumer/tcnws.aspx</a>	Official url to get access to the <b>TCN Web Services Consumer</b> web application enabling Member States CIA users to use the Tachonet web services.
<a href="https://webgate.cec.eu-admin.net/tachonet/test/CIAClientSimulator/main.aspx">https://webgate.cec.eu-admin.net/tachonet/test/CIAClientSimulator/main.aspx</a>	Official url to get access to the <b>CIA Client Simulator</b> web application enabling Member States testers (acting as Liechtenstein) to send their own <i>MS2TCN_&lt;TxName&gt;_Req</i> Xml messages (prepared as files) to TACHOnet and to view the corresponding <i>TCN2MS_&lt;TxName&gt;_Res</i> messages sent back in response by TACHOnet (after having processed and simulated Member States based on some test data files - see below).
<a href="https://webgate.cec.eu-admin.net/tachonet/test/TCNTestData">https://webgate.cec.eu-admin.net/tachonet/test/TCNTestData</a>	Official url to get access to the list of test data files (xml files) used for simulating Member States.

The member states themselves will use: <http://tcn.[countrycode].eu-admin.net/>

In a latter stage all message exchange will be over https

### Test Variables

When each member state is transmitting to the central system what it likes to test during a test cycle, it should use the following nomenclature:

- Member State: XX, 2 letters representing the member state
- Test Phase: N or P (N= normal test phase, P = pilot test phase)
- Test Cycle: 1 to n
- Test scenario id: each test scenario being provided in the last section of this document contains a test scenario id, which corresponds with the TestID proposed in the latest version of the XML Messaging Reference Guide.

*Continued on next page*

## Test Execution, Continued

### Examples of Test Announcement by Member States

**NO-N1-S0111-01** stands for

The Norwegian member state is executing test scenario S0111-01 during the normal test phase, first test cycle.

**DE-P2-S0521-05** stand for

The German member state is executing test scenario S0521-05 during the pilot test phase, second test cycle.

### TestID uniqueness

The TestID itself is unique in that it takes into account the use case, the interface used and the actor performing the activity.

### XML Test Message

When a Member State is testing its XML interface implementation, it will use its test environment (and test interface) to generate XML messages that will be sent to the TCN Central System. The test interface should provide a means to enter the TestID related with the test scenario being executed. This TestID should then be incorporated into the XML message at the exact location being defined in the XML Messaging Reference Guide. The TestID consists of a sequence of 8 characters (-) included.

When testing the Web interface, the interface should contain an extra input field for entering the TestID. This TestID will be captured together with other user input to generate an XML message.

Return XML messages from the TCN Central System will repeat the TestID.

### Test Scenario Nomenclature

Because a separate section of this document is dedicated to the test scenarios themselves, we will limit ourselves by indicating that each Use Case provided in the Software Requirement Specification and repeated in the first section: List of Functional Requirements is covered by one or more Test Cases and each Test Case corresponds with one or more Test Scenarios (according to for instance the tester/user role).

*This means that **S1010-01** stands for a test scenario where the tester is logged on as an Enforcer and follows the default flow of checking a driver's issued card by searching in the local application's database. S1010-01 falls under TC-1010, which is the default test case, and BUC-0010, which stands for the use case: Check driver's issued card. This gives the following traceability:*

BUC-0010 ► TC-1010 ► S1010-01

The complete set for UC-0110 looks like this:

Use Case ID	Test Case ID	TestID
BUC-0010	TC-1010	S1010-01 to S1010-04
	TC-1011	S1011-01 to S1011-04
	TC-1012	S1012-02, S1012-04

*Continued on next page*

## Test Execution, Continued

---

**TestID creation** The TestID consists of 8 characters based on the following pattern:  
Sxxxx-yy where S stands for scenarios, xxxx for the test case number and yy for the test variant.  
If a member state due to its particular local implementation or organizational structure requires or detects extra variants, his numbering should start at yy=20.  
If extra variants are created by a member state and they will be used during a Test Phase, then the member state **MUST** inform the TCN Central System and provide a description of the new variants and what they are supposed to test so the TCN Central System can anticipate and provide adequate test responses.

---

**Test Scenario Contents** Each test scenario will contain:

- A description of the test scenario, pre- and post conditions
- A sequence of steps describing what the tester or application should do
- For each step, the expected result
- A list of checkpoints where the tester should make verifications (on interface or in the database)
- The XML message(s) to send out
- The XML message(s) to receive

---

## Test Reporting

---

### Test Result Gathering

The gathering of test results can be done by using one of the tools mentioned in the Test Tools chapter. Make sure that you work by test cycles which should contain the following data:

- Name Tester
- Date(s) of testing
- Test Cycle Indicator
- Build version being tested (mainly for local tests)
- Complete test list (all test scenarios provided + those added by the implementer)

Depended on the tools being used, the execution and result gathering will be done manually or automated.

---

### Test Reporting

Each test cycle report should contain in general the test variables defined in a previous section such as Member State, Test Phase, Test Cycle and the test scenarios (TestIDs) actually executed. Besides this, for each test scenario:

- If it was executed during the test cycle
- If the test passed or failed (optionally you could add “partially passed” when during test execution something went wrong)
- For those failed tests add bug identifier

The test cycle report is send by the end of the third day of the test cycle so that possible problems or bugs could be fairly quickly resolved allowing the parties to perform an additional test cycle.

The test manager or SPOC does the transmission of test reports.

---

### Bug Reporting

When performing local tests, each member state decides for him self how they will keep track of errors occurred during testing. When participating in a test phase (and test cycle) however, the bug reporting is standardized.

The central system will use a bug-tracking tool to enter and follow-up the handling of bugs reported by the member states (and central system itself) during test cycles.

Each test report will be accompanied by a bug report (if any). The bug report consists of a list of bug identifiers used in the test report and a description of the bug encountered. If possible a screenshot of the error message could be provided too. The more detailed the bug report, the easier to reproduce the error on a developer's station.

Each bug reported will obtain a unique bug number in the centralized tracking tool and will be made available to all member states participating in the test phases. This is especially useful when identifying if a bug is caused by the central system application, the web site or the local XML interface application.

---

*Continued on next page*

## Test Reporting, continued

---

### **Bug Description**

The central bug tracking tool will characterize each bug accordingly:

- Category: an indication what part of the architecture is faulty (interface, business logic, middle-tier, back-end, database, flow etc)
- Reproducibility: how often does the same bug occur (always, often, sometimes etc)
- Severity: how sever should this bug be considered (minor, major, tweak etc)
- Summary: short, one line description of the error detected
- Description: more extensive description of the error detected
- Additional information: information related to the bug, that might aid the developers in debugging
- A unique bug sequence number

The tool supports an active follow-up mechanism by adverting bug reporters, and bug solvers about the status of the bug through email and through a web site.

---

## Test Data and Member State Simulation

---

### Introduction

During the normal test phase, each member state will test its local implementation and the exchange of information with the Tachonet central system.

Due to the nature of the overall architecture, a member state will send out data to inform other member states but will also want to retrieve information coming from other member states.

The need for a(nother) member state is real hence the creation of the MS Simulator.

Because a member state has to deal with data that he does not own, test data is required too to cover this aspect

---

### Driver Data

Each Driver can be linked towards a Driving License(DL) and a Driver Card(DC). Both DL and DC can be obtained in the country of residence or not. A simple permutation offers us 4 combinations:

- (1) DL local, DC local
- (2) DL local, DC not local
- (3) DL not local, DC local
- (4) DL not local, DC not local

A member state testing its own implementation can perform tests with Drivers having a local Driver Card and a local Driving License because this combination (1) does not require a TCN interaction.

Enforcers however, might end up with checking cards not issued by the native country or with Drivers that do not have a local DL. At that moment a TCN interface is required to request information stored at one or an unknown member state.

---

### Test Data

We have created a set of xml files that contain drivers with Phonex search keys, driver card details and driving license numbers. The drivers are grouped per country of residence and created in such a way that they cover combinations (2) and (3).

In case the issuing state of a driver card is unknown, TCN will broadcast the request to all member states connected so combination (4) would be covered too.

---

*Continued on next page*

## Test Data and Member State Simulation, Continued

---

### Test Data Example

The following snippet comes from the Norwegian test data file and represents a Driver with a local DC but a non-local DL.

```
...
<Driver>
  <DriverDetails>
    <DriverId>d035</DriverId>
    <Surname>OPSAHL</Surname>
    <Firstname>JAN-MAGNE</Firstname>
    <Skey>A1240</Skey>
    <Fkey>G5250</Fkey>
    <Birthdate>1960-06-21</Birthdate>
    <PlaceOfBirth/>
  </DriverDetails>
  <DriverCardDetails>
    <DCNumber>N91021961Y</DCNumber>
    <DCCiA>N</DCCiA>
    <DCStatus>Replaced</DCStatus>
    <DCStartOfValidityDate>2003-07-08</DCStartOfValidityDate>
    <DCExpiryDate>2005-07-08</DCExpiryDate>
    <DCStatusModifiedAt>2003-07-08T10:07:10</DCStatusModifiedAt>
  </DriverCardDetails>
  <DriverLicenseDetails>
    <DLNumber>DK493200</DLNumber>
    <DLIsNation>D</DLIsNation>
    <DLStatus>Valid</DLStatus>
    <DLIssueDate>1980-08-11</DLIssueDate>
  </DriverLicenseDetails>
</Driver>
...
```

Now an Enforcer in Denmark could be checking the Norwegian Driver's DriverCard status by accessing this file, which in fact would be located somewhere in Norway.

Also, an Enforcer in Germany could be verifying the existence of a Driver Card for the Norwegian Driver with German Driving license first locally then by broadcasting it through TCN.

---

### Member State Simulator

We have created 8 such files for 8 different member states. If a member state wants to send a request to a 9<sup>th</sup> member state, then the simulator will notice that no data is available and will send back a xml response with NotFound for that country.

The following countries are currently available: D, E, FIN, GB, GR, N, NL, and P.

---

### Use of Test Id

It is imperative that during the normal test period when using the member state simulator and test data files you introduce the correct TestId in your xml messages allowing the simulator to generate the correct and expected response.

Each TestId presented in this document can be used and is supported in that it will give a response be it OK, NotFound or TimedOut.

---

*Continued on next page*

## Test Data and Member State Simulation, Continued

---

**Member State  
as data  
provider**

If a Member State wants to be tested by TCN as provider it can reuse the test data file for drivers with a DriverCard in their country or they can provide us a list of drivers they used when they were testing their local implementation. (see combination (1) )

---

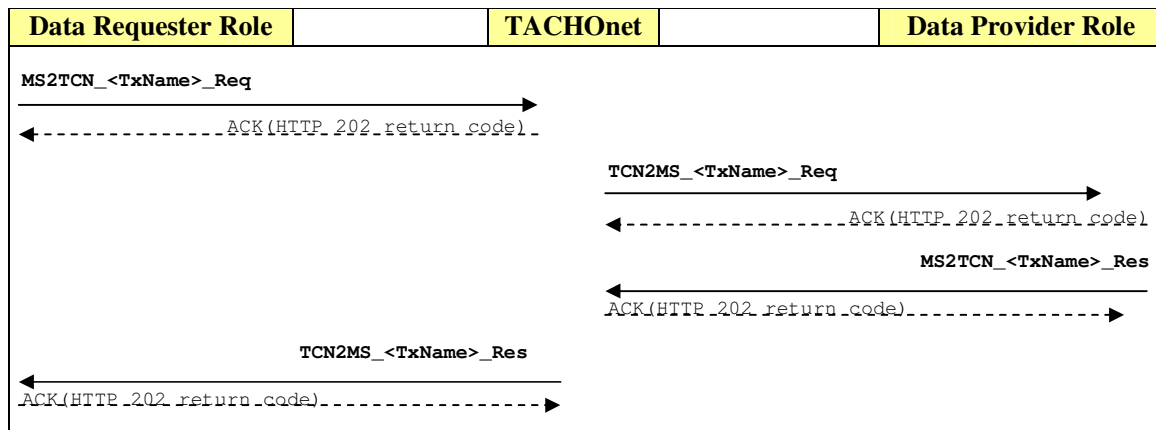


# CiA Client Simulator

## Data Requester and Data Provider Roles

As shown in **Figure 1**, testing the TCN transactions is twofold:

- A Member State needs to test the **data requester role** of her system, i.e. her ability to send *MS2TCN\_<TxName>\_Req* messages to TACHOnet and to receive corresponding *TCN2MS\_<TxName>\_Res* messages from TACHOnet.
- A Member State needs to test the **data provider role** of her system, i.e. her ability to receive *TCN2MS\_<TxName>\_Req* messages from TACHOnet and to send back corresponding *MS2TCN\_<TxName>\_Res* messages to TACHOnet.



**Figure 1 – Member State Roles in a TCN transaction**

*Continued on next page*

## CiA Client Simulator, Continued

**Testing Roles** In the test environment (and not in the production environment), a Member State could decide to test only her **data requester role** or only her **data provider role** or **both**. Each Member State should provide the TCN test team (via email to [be.test.tachonet@getronics.com](mailto:be.test.tachonet@getronics.com) ) with the roles it'd to test along with the corresponding required configuration information as outlined in the following table:

If the Member State wants to test...	Then...
Only the data requester role	<ul style="list-style-type: none"> <li>The Member State should provide the TCN test team (via email to <a href="mailto:be.test.tachonet@getronics.com">be.test.tachonet@getronics.com</a> ) with the official url address of her CIA application where TACHOnet should send back <i>TCN2MS_&lt;TxName&gt;_Res</i> response messages.</li> <li>The CIA application must send valid <i>MS2TCN_&lt;TxName&gt;_Req</i> xml message, including a valid <i>TestId</i> attribute value, to the official TACHOnet test core system: <a href="https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll">https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll</a> .</li> <li>TACHOnet will process the incoming message (eventually broadcast it to simulated Member States or to actual Member State(s) configured as testing their data provider role), the incoming responses and send back the final <i>TCN2MS_&lt;TxName&gt;_Res</i> response message to the supplied address.</li> <li><b>Important:</b> when a Member State is configured to test only her data requester role, her data provider role will be simulated by TACHOnet using some test data available as XML files at <a href="https://webgate.cec.eu-admin.net/tachonet/test/TCNTestData">https://webgate.cec.eu-admin.net/tachonet/test/TCNTestData</a> ).</li> </ul>
Only the data provider role	<ul style="list-style-type: none"> <li>The Member State should provide the TCN test team (via email to <a href="mailto:be.test.tachonet@getronics.com">be.test.tachonet@getronics.com</a> ) with the official url address of her CIA application where TACHOnet should send <i>TCN2MS_&lt;TxName&gt;_Req</i> request messages.</li> <li>The Member State will use the <b>CiA Client Simulator</b> (acting as Liechtenstein's CIA data requester) to send her own prepared <i>MS2TCN_&lt;TxName&gt;_Req</i> request messages (targeting her own Member State by specifying her Member State code as <i>IssuingMemberStateCode</i> or <i>DrivingLicenseIssuingNation</i> attribute value), including a valid <i>TestId</i> attribute value, to the official TACHOnet test core system: <a href="https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll">https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll</a>.</li> </ul>

Continued on next page

## CiA Client Simulator, Continued

### Testing Roles (continued)

If the Member State wants to test...	Then...
Only the data provider role (cont'd)	<ul style="list-style-type: none"> <li>TACHOnet will process the incoming message and send the corresponding <i>TCN2MS_&lt;TxName&gt;_Req</i> xml message, including the mapped <i>TestId</i> attribute value, to the Member State CIA application (if specified in <i>IssuingMemberStateCode</i> or <i>DrivingLicenseIssuingNation</i> attribute value) using the Member State's supplied address for testing her data provider role).</li> <li>The actual Member State CIA application will process the incoming <i>TCN2MS_&lt;TxName&gt;_Req</i> request message and send back a corresponding <i>MS2TCN_&lt;TxName&gt;_Res</i> response message to the official TACHOnet test core system: <a href="https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll">https://webgate.cec.eu-admin.net/tachonet/test/TCN/BizTalkHTTPReceive.dll</a>.</li> <li>TACHOnet will then process the response message(s) and send back the final <i>TCN2MS_&lt;TxName&gt;_Res</i> response message to the <b>CiA Client Simulator</b>.</li> <li><b>Important:</b> when a Member State is configured to test her data provider role, this Member State will no longer be simulated by TACHOnet. Therefore, every time TACHOnet needs to send a <i>TCN2MS_&lt;TxName&gt;_Req</i> message to that Member State (e.g. on behalf of another Member State testing her data requester role), it will send it to the supplied url (and no longer to the simulated one).</li> </ul>
Both roles	<ul style="list-style-type: none"> <li>The Member State should provide the TCN test team (via email to <a href="mailto:be.test.tachonet@getronics.com">be.test.tachonet@getronics.com</a>) with the official url address of her CIA application (something like <b>Error! Hyperlink reference not valid.</b>) where TACHOnet should send <i>TCN2MS_&lt;TxName&gt;_Req</i> request messages (testing the data provider role of the Member State) <b>and</b> <i>TCN2MS_&lt;TxName&gt;_Res</i> response messages (testing the data provider role of the Member State).</li> <li>Please refer to the corresponding procedure described above for testing the data requester or data provider role.</li> </ul>

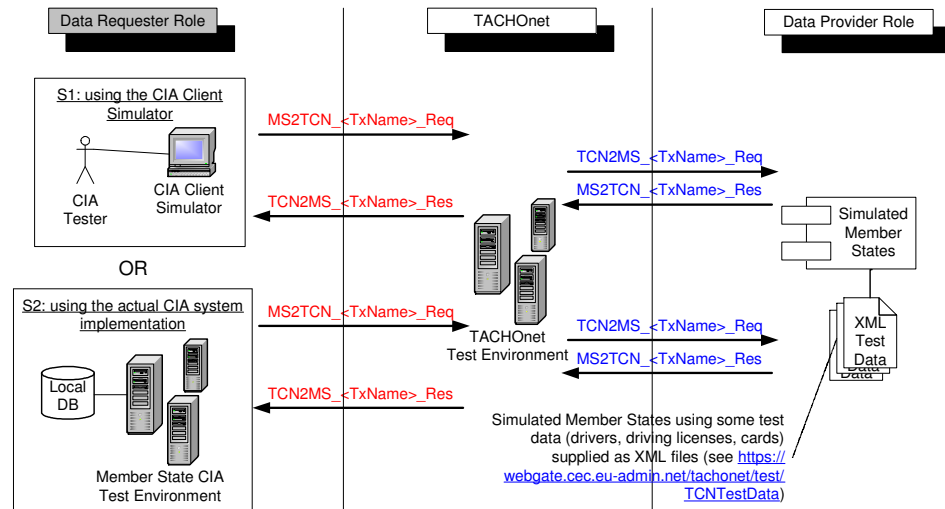
Continued on next page

## CiA Client Simulator, Continued

### Testing the Data Requester Role

As illustrated in **Figure 2**, two mechanisms are available for testing the **Data Requester role** of a Member State CIA:

- Using the **CIA Client Simulator**
- Using the actual implementation of the Member State CIA



**Figure 2 – Testing the Data Requester Role**

#### S1: Using the CIA Client Simulator

This 1<sup>st</sup> mechanism doesn't actually test any Member State implementation but is useful to check the expected **TCN2MS\_<TxName>\_Res** response message to a given **MS2TCN\_<TxName>\_Req** request message. Please refer to **"Error! Reference source not found."** at page **Error! Bookmark not defined.** for more details how to use the CIA Client Simulator.

#### S2: Using the actual CIA system implementation

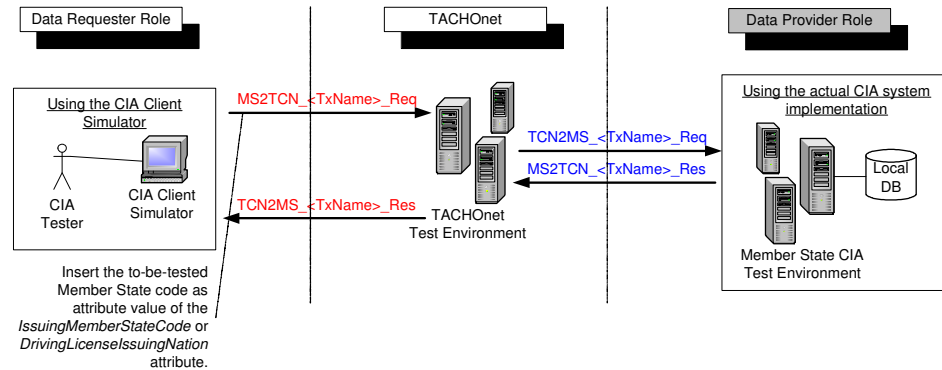
This 2<sup>nd</sup> mechanism is the only one that will actually check the ability of a current Member State system implementation to send **MS2TCN\_<TxName>\_Req** request message to TACHOnet and to receive corresponding **TCN2MS\_<TxName>\_Res** response message from TACHOnet (based on test data provided simulated Member States). Please refer to **"Testing Roles"** at page 48 for more details.

*Continued on next page*

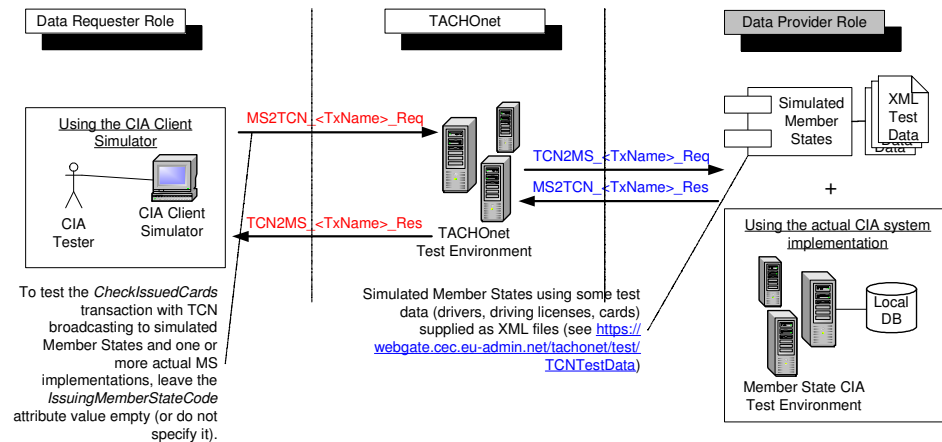
## CiA Client Simulator, Continued

### Testing the Data Provider Role

As illustrated in **Figure 3** and **Figure 4**, testing the data provider role of a Member State CIA application consists of sending *TCN2MS\_<TxName>\_Req* messages to this CIA application. Triggering such sending (i.e. telling TACHOnet to send such messages) can be done by using the **CIA Client Simulator**. Please refer to “**Error! Reference source not found.**” at page **Error! Bookmark not defined.** for more details how to use the CIA Client Simulator.



**Figure 3 – Testing the Data Provider Role (single MS)**



**Figure 4 – Testing the Data Provider Role (broadcast to MS)**

*Continued on next page*

## CiA Client Simulator, Continued

---

### What's the CIA Client Simulator?

The **CIA Client Simulator** is a simple web application, accessible to anyone connected to TESTA, that acts as the Liechtenstein's CIA and enables any user to test the TCN transactions by sending her own prepared *MS2TCN\_<TxName>\_Req* messages to TACHOnet and view the corresponding *TCN2MS\_<TxName>\_Res* messages sent back by TACHOnet.

Prior to sending to TACHOnet the selected *MS2TCN\_<TxName>\_Req* message (xml file prepared by a Member State), the **CIA Client Simulator** will change the following attributes of the *Header* element:

- *From*: it will be set to TCN\_FL (acting as Liechtenstein)
- *SentAt*: it will be set to the current date & time (UTC).
- *MSRefId*: it will be set to a new guid, used also in the filename under which the sent message will be stored (see below).

Once sent, it will save a copy of the sent message in the specified folder name (see below) with the following naming convention:

{*MSRefId*}\_<TxName>\_Req.xml

The corresponding *TCN2MS\_<TxName>\_Res* message sent back by TACHOnet will be stored under the same folder name (see below) with the following naming convention:

{*MSRefId*}\_<TxName>\_Res.xml

Please note that the xml files stored under **/CiaClientMessages** folder will be deleted when older than 2 days.

The goal of this web application is twofold:

- It can be used check the expected *TCN2MS\_<TxName>\_Res* response message to a given *MS2TCN\_<TxName>\_Req* request message based the current configuration of the test environment.
- It can be used by a Member State to test her data provider role. Indeed, such Member State can use this web application, acting as the Liechtenstein's CIA, to send prepared *MS2TCN\_<TxName>\_Req* messages (targetting her Member State and including some test data corresponding to her local database) to TACHOnet which, in turn, will send corresponding *TCN2MS\_<TxName>\_Req* messages to this Member State.

---

### Url address

Open your browser and go to:

<https://webgate.cec.eu-admin.net/tachonet/test/CIAClientSimulator/main.aspx>.

---

*Continued on next page*

## CiA Client Simulator, Continued

### User interface

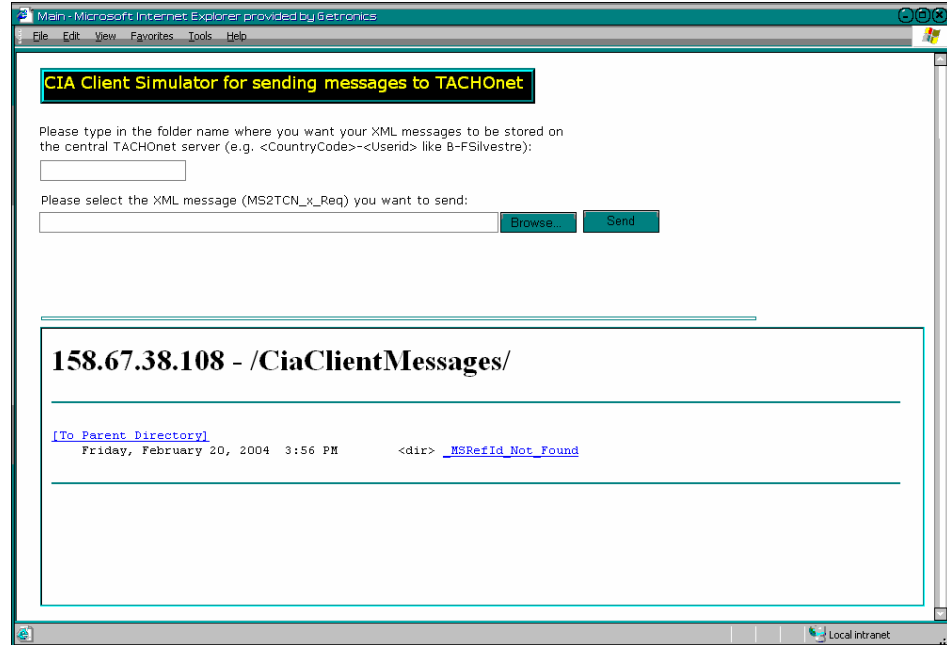


Figure 5 – CIA Client Simulator user interface

### Description of the fields

The following table gives a short description of the user interface:

Part	Type	Function
Folder name	Text box	Please provide a name that will be used, on the central TCN system, as folder name where the XML messages you send via this web application will be stored. The specified folder name will be created (if not yet existing) on the TCN system and accessible for directory browsing in the iframe window (under root CiAClientMessages/) in the second half of the screen.
XML Message + Browse	Text box + button	Please click <b>Browse</b> to select a file you've already prepared ( <i>MS2TCN_&lt;TxName&gt;_Req</i> message) and you'd like to send to TACHOnet.

*Continued on next page*

## CiA Client Simulator, Continued

### Description of the fields (continued)

Part	Type	Function
Send	Button	<p>Click this button to send the selected xml file to TACHOnet on behalf of the Liechtenstein's CIA. Indeed, prior to sending the xml file, the <b>CIA Client Simulator</b> will change the following attributes of the <i>Header</i> element:</p> <ul style="list-style-type: none"> <li>• <i>From</i>: it will be set to TCN_FL (acting as Liechtenstein)</li> <li>• <i>SentAt</i>: it will be set to the current date &amp; time (UTC).</li> <li>• <i>MSRefId</i>: it will be set to a new guid, used also in the filename under which the sent message will be stored (see below).</li> </ul> <p>Once sent, it will save a copy of the sent message in the specified folder name (see below) with the following naming convention:</p> <p style="text-align: center;">{ <i>MSRefId</i> } _&lt;TxName&gt;_Req.xml</p>
Reset	Button	Used to clear the input fields
<i>Second half part of the screen (iframe for directory browsing)</i>		
CiaClientMessages	iframe	<p>This iframe displays the folders structure where all sent and received XML messages are stored. Please click the folder name you've specified when sending the xml message. The iframe displays the list of xml files already sent (with _Req.xml extension) and received (with _Res.xml extension). The list of files is sorted in alphabetic order so that the pair of request and response files for a given transaction (MSRefId) is always displayed one close to the other.</p> <p>Please refresh this iframe (right-click in the iframe and select <b>Refresh</b>) to check for new incoming response messages.</p> <p>Please not that the special <b>_MSRefId_Not_Found</b> folder will contain all <i>TCN2MS_&lt;TxName&gt;_Res</i> messages received from TACHOnet and for which no corresponding <i>MS2TCN_&lt;TxName&gt;_Req</i> messages (based on <i>MSRefId</i> attribute value) was found in the other folders (probably because the saved request file has been deleted in the meantime).</p>

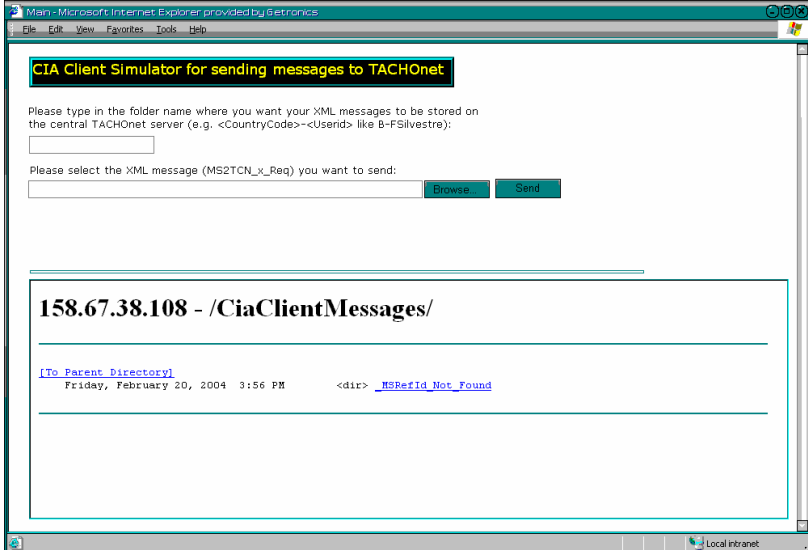
*Continued on next page*



## CiA Client Simulator, Continued

### Example

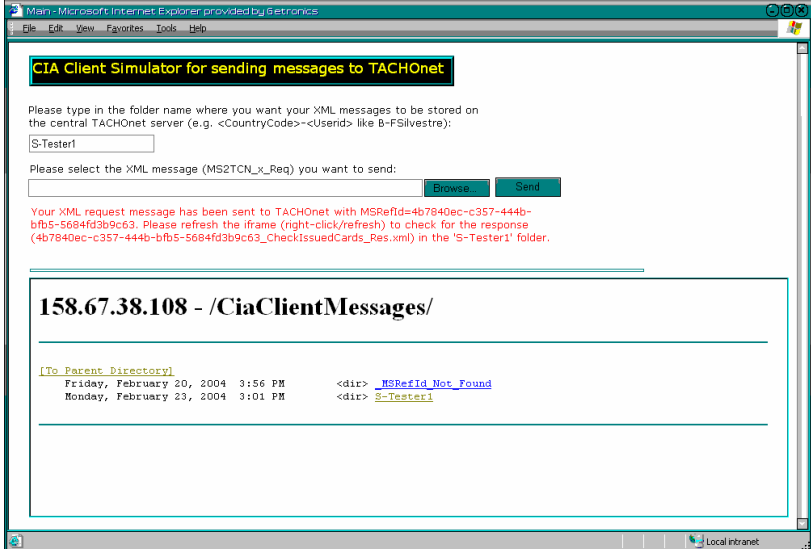
The following example illustrates the steps for sending a *MS2TCN\_CheckIssuedCards\_Req* message to TACHOnet and viewing its corresponding *TCN2MS\_CheckIssuedCards\_Res* response sent by TACHOnet:

Step	Action
1	<p>The user has prepared an XML file corresponding to the XML message she'd like to send to TACHOnet. Building such XML message can easily be done using a tool like XML Spy by asking it to generate XML message based on the TACHOnet XML schema (tcn.xsd). An example of such generated XML file could be the following:</p> <p>Do not forget to insert the <i>TestId</i> attribute value and the search criteria values corresponding to the test scenario you'd like to test. Otherwise, the simulated Member State(s) will reply 'ServerError' (if no valid <i>TestId</i> is provided) as <i>Header/StatusCode</i> attribute value.</p>
2	<p>Open the browser and go to <a href="https://webgate.cec.eu-admin.net/tachonet/test/CIAClientSimulator/main.aspx">https://webgate.cec.eu-admin.net/tachonet/test/CIAClientSimulator/main.aspx</a>. The following screen is displayed:</p> 
3	<p>Type in a folder name (e.g. <i>S-tester1</i> if you're a tester from Sweden), click <b>Browse</b>, select the file you've created (see step 1) and click <b>Send</b>.</p>

*Continued on next page*

## CiA Client Simulator, Continued

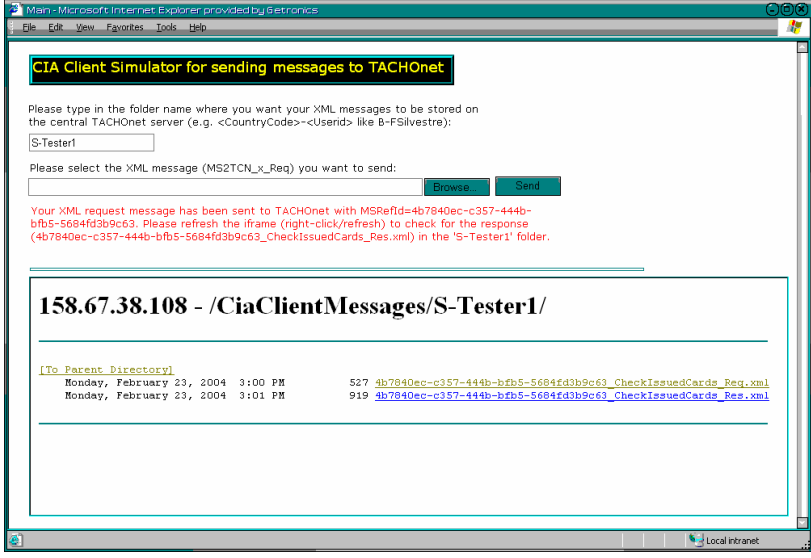
### Example (continued)

Step	Action
4	<p>The <b>CIA Client Simulator</b> will change some attributes of the <i>Header</i> element of your message (i.e. <i>From</i>, <i>SentAt</i> and <i>MSRefId</i>) and send it TACHOnet. Once sent, it will save a copy of the sent message in the specified folder name (e.g. <i>S-tester1</i>) with the following naming convention: <code>{MSRefId}_&lt;TxName&gt;_Req.xml</code></p> <p>The following message confirms the sending of the message:  <span style="color: red;">Your XML request message has been sent to TACHOnet with MSRefId=4b7840ec-c357-444b-bfb5-5684fd3b9c63. Please refresh the iframe (right-click/refresh) to check for the response (4b7840ec-c357-444b-bfb5-5684fd3b9c63_CheckIssuedCards_Res.xml) in the 'S-Tester1' folder.</span></p> 

*Continued on next page*

## CiA Client Simulator, Continued

### Example (continued)

Step	Action
5	<p>Click the corresponding folder name (e.g. <i>S-tester1</i>) to check for the response message (<i>{MSRefId}_&lt;TxName&gt;_Res.xml</i>). Right-click the iframe and click Refresh to refresh the folder contents.</p> 
6	<p>Within a minute, the corresponding <i>{MSRefId}_&lt;TxName&gt;_Res.xml</i> file should appear in the list. Click on it to view its contents.</p>

## Pilot Test Phase

---

### Introduction

As mentioned in the Test Execution section, a separate test activity called the Pilot Test phase will be initiated between those member states that have already implemented their local CIA client and the central TACHOnet system.

This activity is limited in time and scope but nevertheless very important towards a successful deployment of the TACHOnet solution into production.

The exchange of the 4 standard messages will be tested out between multiple connected member states and a first simulation of the predicted load and volume on the TESTA network will be executed.

---

### Proposition

During the workgroup meeting of 6 May 2004 those member states that were ready with their local implementation (Italy, France, Sweden, Netherlands and Spain) were invited to partake and to exchange some ideas on how they would have like to test the overall solution.

The basis of that meeting was a proposition made by the Italian participant. This proposition and the remarks collected during the workgroup meeting will form the basis of the remainder of this section.

We have taken the liberty to extend and clarify the proposition where needed.

---

### Test Procedure

The Test Phase test procedure is based on the Test Organization presented in a previous chapter in that we require a definition for

- a test cycle: when to test, when to check and when to report
  - a test id: how to communicate what we are testing
  - a test case: which case will be executed
  - the test data structure
  - the test activities: what to test first
- 

### Test Cycle

It has been convened that the Pilot Test Phase will start at May 24th and will run until June 4<sup>th</sup>, then spread over two weeks.

The first week will be dedicated to functional testing; each participating member state will have four consecutive days to send and receive xml messages. The fifth day will be used to collect, interpret and share the test results.

The second week should be dedicated to stress & load testing (unless problems during the 1<sup>st</sup> week require an extension of the functional testing).

The actual contents of the test cycle, the activities will be discussed shortly.

---

*Continued on next page*

## Pilot Test Phase, Continued

---

**Test Activities** The first week requires all member states to participate actively both as data requester and data provider. In terms of test types, the member states will be executing function and business cycle test that first week. Which tests case to execute will be discussed shortly.

A second type of tests that will be performed with the same group of Member States and the central system are stress and load tests. The goal of those tests is to verify and capture performance issues in the local implementation, the TESTA bandwidth and the central system taking into account the foreseen and expected performance thresholds. Those kinds of tests could be automated and as such be launched overnight. The contents of the xml messages and the expected results are less important, sufficient to say that they should be selected amongst the defined TestIds.

During those stress tests a distinction should be made between the sending of messages online (simulating a normal working-hours CIA client behavior) and the sending of batch messages (simulating the after-hours CIA client behavior).

---

**Remarks** During the Pilot Test Phase other member states besides those that are partaking (Italy, Spain, Sweden and the Netherlands) will not be able to test their local implementation as the test environment will fully be dedicated to the Pilot test phase. Therefore, the other Member States should be warned that they should not send any XML messages over the two weeks of the Pilot test phase (we can't prevent them from doing it).

---

**Test Id** Because the amount of test cases that will be used is restricted and we like to capture the data requester in the test id we have opted to change the contents of the test id during the Pilot Test. The upper boundary of 8 character remains but how they are filled in is changed.

YYY = stands for the sending country be it I, E, NL, or S

NNN = stands for the test case number

So for instance **S125** is test case 125 being send out by Sweden

```
<MS2TCN_CheckIssuedCards_Req xmlns="urn:eu.cec.tren.tcn">  
<Header Version="1.4" MSRefId="1234567890" SentAt="2004-04-  
27T15:05:16" TimeoutValue="60" From="TCN_S" To="TACHOnet "  
TestId="S125" />  
<Body>  
...  
</Body>  
</MS2TCN_CheckIssuedCards_Req>
```

---

*Continued on next page*

## Pilot Test Phase, Continued

---

### Test Data

During the Normal Test Phase we explained that in order to cover the maximum of test cases we needed a driver test database that contained drivers with not only local driving licenses and driver cards but also drivers with driving licenses issued in another member state and driver card issued by another member state.

It is especially those latter cases we would like to test during the Pilot Test Phase because those situations require TACHOnet and TESTA.

---

### Driver Data

We have provided in a previous section of this document the structure of a Driver with subsections DriverCardDetails and DriverLicenseDetails. This structure is used in our Test Date files of our CIA Client Simulator.

The proposed structure by our Italian colleagues is slightly different in that it contains less attributes. Each participating member state is free to use one of those structure as long as the structure contains the following information:

- Firstname
- Surname
- Birthdate
- Birthplace (optional)
- DrivingLicenseNumber
- DrivingLicenseIssuingNation
- DrivingLicenseStatus (valid/invalid)
- CardNumber
- CardStatus
- CardIssuingNation

We have added fields concerning the driving license because some member states have the intention to verify not only the driver card but the driving license too through TACHOnet.

---

*Continued on next page*

## Pilot Test Phase, Continued

---

### Driver Data file format

The driver test data that should be exchanged between the participants prior to testing should be done in XML independent of the structure followed. This will allow each member state to convert and incorporate the data into its local database.

Whenever a new pilot phase has been scheduled with the Member States, the file has to be sent to the Tachonet Test mailbox [be.test.tachonet@getronics.com](mailto:be.test.tachonet@getronics.com) where it will be grouped and distributed to all participants. Make certain that your xml file does not end on .xml but add the .txt extension to it like **ItalyTestdata.xml.txt**. This assures you that your data file will not be blocked by a firewall considering an xml file as a potential security threat.

The file has to be send to the mailbox 5 days before the launch so that each participant has sufficient time to introduce the data in their database.

The amount of drivers contained in the file should be at least 5 and must be relevant to cover the test cases. (see later).

---

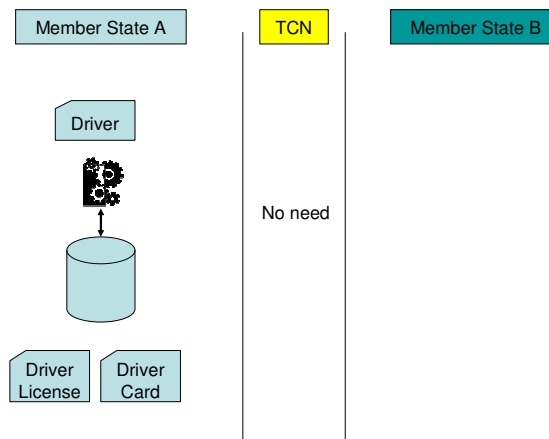
### Driver Data contents - I

As you already guessed, we are not looking for default drivers or those drivers that a member state uses to test its own application. In fact we are testing the special cases and indirectly one member state is testing the implementation of another member state by sending out requests to it where the answer is know up-front.

To better comprehend the situation we present four scenarios that will be reflected into a set of test cases executed by the member states and which should aid in filling in the test data for the test drivers.

---

### Scenario 1



All relevant data is stored and available locally in the Member State.  
Connectivity with TACHOnet and other Member States is not required.

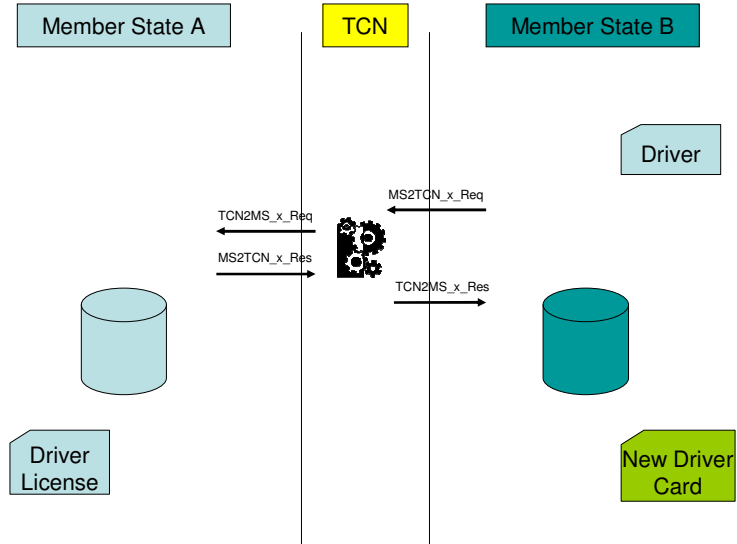
This scenario will not be reflected in the test cases because TACHOnet is not required to execute it (everything happens locally).

---

*Continued on next page*

## Pilot Test Phase, Continued

### Scenario 2



A driver with nationality A, living in B is requesting a Driver Card in B. Connectivity with TACHOnet is required to check for existing Driver Card in A and to obtain Driver license info.

This scenario can be covered by *CheckIssuedCards* by MS B to find out if the Driver requesting the creation of a Driver Card does not have already a Driver Card in another member state.

It could also be initiated by an enforcer in MS B to obtain more information on the Driver License status issued in MS A.

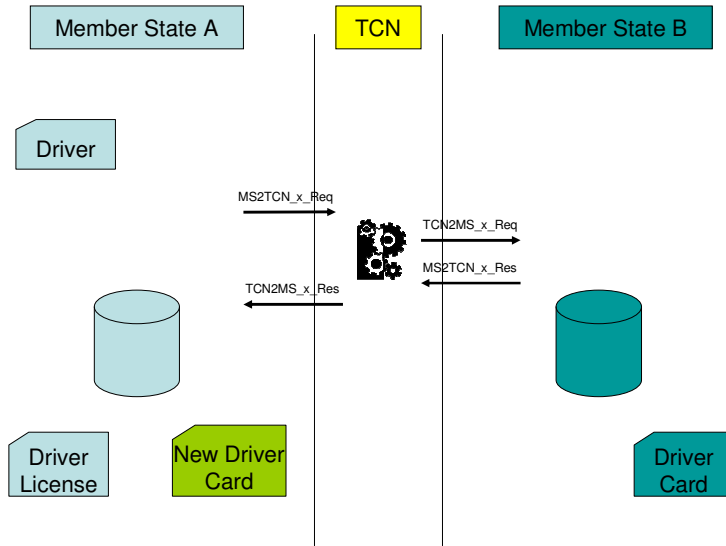
This scenario can be covered by *IssuedCardDL* when a new Driver Card is being issued by MS B and MS A is being informed.

*Continued on next page*



## Pilot Test Phase, Continued

### Scenario 3



A driver with nationality A, living in A is requesting a Driver Card in A.  
Connectivity with TACHOnet is required to check for existing Driver Card in B.

This scenario can be covered by *CheckIssuedCards* initiated by MS A to verify if the Driver (living recently back in A) has not obtained a Driver Card in MS B.

This scenario can be covered by *CheckCardStatus* when an enforcer of MS A is inspecting the Driver Card issued by MS B.

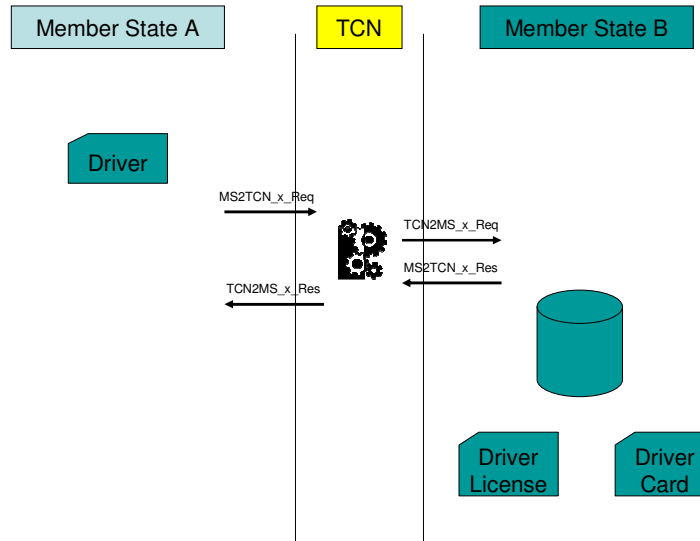
This scenario can be covered by *ModCardStatus* when an inspected Driver in MS A with Driver Card issued by MS B reports a loss of card.

*Continued on next page*

## Pilot Test Phase, Continued

---

### Scenario 4



A driver with nationality B, living in B is being checked in A.  
Connectivity with TACHOnet is required to check for existing Driver Card and status in B and to obtain Driver License info.

This scenario can be covered by *CheckCardStatus* when a Driver of MS B is being inspected by enforcer of MS A.

This scenario can be covered by *ModCardStatus* when a Driver of MS B is claiming his card is lost, stolen or malfunctioning.

---

### Driver Data contents - II

As has been made clear through those 4 scenarios, each Member State A must provide the other Member States B a set of drivers born in A but with only a Driver License issued in A or a Driver Card issued by A.

The Member States B should provide Member State A a set of drivers of nationality B, drivercard issued by B and driving license issued in B.

Remember that B stands for all non-A Member States and not just one Member State. This is required when you want TACHOnet to broadcast instead of sending it straight to one Member State.

---

*Continued on next page*

## Pilot Test Phase, Continued

---

### Driver Data contents - III

Especially for those test cases where you want more than one result you should provide Drivers with a Phonex search key that cover more than one Driver and if possible Drivers with a different nationality so that different databases have to be accessed. This could be easily simulated by introducing a fake Driver name in each existing database having a different (local) Driver License number and (local) Driver Card.

This test case also allows us to verify if each Member State has correctly implemented the Phonex algorithm and is obtaining the same Driver(s).

---

### Test Cases

The following list of test cases and their corresponding test case number (part of the TestID) are being proposed to be executed during the Pilot Test Phase. Use the following legend:

<b>TestId</b>	Test case number
<b>Request Type</b>	Message type
<b>Type</b>	OL = OnLine B = Batch
<b>To MS</b>	MS_1 = The Sender Member State MS_2 = A Receiver Member State MS_3 = A Receiver Member State MS_4 = A Receiver Member State
<b>Response expected</b>	Result of the request
<b>Note</b>	Some notes

---

*Continued on next page*

## Pilot Test Phase, Continued

**Test Cases**  
 (continued)

Request Message				Response Message	
TestId	Request Type	Type	To MS	Response expected	Note
110	CheckIssuedCard	OL	MS_2	Found	1 occurrence of DriverDetails in MS2TCN
111	CheckIssuedCard	OL	MS_2	Found	N occurrences of DriverDetails in MS2TCN
112	CheckIssuedCard	OL	MS_2	NotFound	0 occurrence of DriverDetails in MS2TCN
115	CheckIssuedCard	OL	MS_3	Found	1 occurrence of DriverDetails in MS2TCN
116	CheckIssuedCard	OL	MS_3	Found	N occurrences of DriverDetails in MS2TCN
117	CheckIssuedCard	OL	MS_3	NotFound	0 occurrence of DriverDetails in MS2TCN
120	CheckIssuedCard	OL	MS_4	Found	1 occurrence of DriverDetails in MS2TCN
121	CheckIssuedCard	OL	MS_4	Found	N occurrences of DriverDetails in MS2TCN
122	CheckIssuedCard	OL	MS_4	NotFound	0 occurrence of DriverDetails in MS2TCN
125	CheckIssuedCard	OL	All	Found	1 occurrence of DriverDetails in MS2TCN

*Continued on next page*

## Pilot Test Phase, Continued

### Test Cases - continued

Request Message				Response Message	
TestId	Request Type	Type	To MS	Response expected	Note
126	CheckIssuedCard	OL	All	Found	N occurrences of DriverDetails in MS2TCN
127	CheckIssuedCard	OL	All	NotFound	0 occurrences of DriverDetails in MS2TCN
130	CheckIssuedCard	B	MS_2	Mix Found & NotFound	N occurrences of DriverDetails in MS2TCN
135	CheckIssuedCard	B	MS_3	Mix Found & NotFound	N occurrences of DriverDetails in MS2TCN
140	CheckIssuedCard	B	MS_4	Mix Found & NotFound	N occurrences of DriverDetails in MS2TCN
145	CheckIssuedCard	B	All	Mix Found & NotFound	N occurrences of DriverDetails in MS2TCN
210	CheckCardStatus	OL	MS_2	Found	
211	CheckCardStatus	OL	MS_2	NotFound	
215	CheckCardStatus	OL	MS_3	Found	
216	CheckCardStatus	OL	MS_3	NotFound	
220	CheckCardStatus	OL	MS_4	Found	
221	CheckCardStatus	OL	MS_4	NotFound	
225	CheckCardStatus	B	MS_2	Mix Found & NotFound	
230	CheckCardStatus	B	MS_3	Mix Found & NotFound	
235	CheckCardStatus	B	MS_4	Mix Found & NotFound	
310	ModCardStatus	OL	MS_2	OK	
311	ModCardStatus	OL	MS_2	Cardnumber NotFound	
315	ModCardStatus	OL	MS_3	OK	
316	ModCardStatus	OL	MS_3	Cardnumber NotFound	

*Continued on next page*

## Pilot Test Phase, Continued

### Test Cases - continued

Request Message				Response Message	
TestId	Request Type	Type	To MS	Response expected	Note
320	ModCardStatus	OL	MS_4	OK	
321	ModCardStatus	OL	MS_4	Cardnumber NotFound	
325	ModCardStatus	B	MS_2	Mix OK and CardNumber NotFound	
330	ModCardStatus	B	MS_3	Mix OK and CardNumber NotFound	
335	ModCardStatus	B	MS_4	Mix OK and CardNumber NotFound	
410	IssuedCardDL	OL	MS_2	OK	
411	IssuedCardDL	OL	MS_2	DrivingLicenseNum ber NotFound	
415	IssuedCardDL	OL	MS_3	OK	
416	IssuedCardDL	OL	MS_3	DrivingLicenseNum ber NotFound	
420	IssuedCardDL	OL	MS_4	OK	
421	IssuedCardDL	OL	MS_4	DrivingLicenseNum ber NotFound	
425	IssuedCardDL	B	MS_2	Mix OK and DrivingLicenseNum ber NotFound	
430	IssuedCardDL	B	MS_3	Mix OK and DrivingLicenseNum ber NotFound	
435	IssuedCardDL	B	MS_4	Mix OK and DrivingLicenseNum ber NotFound	

*Continued on next page*

## Pilot Test Phase, Continued

### Load Testing

Load testing is used to validate and assess acceptability of the operational limits of the TACHOnet system under varying workloads while the system-under-test remains constant. In some variants, the workload remains constant and the configuration of the system-under-test is varied. Measurements are usually taken based on the workload throughput and in-line transaction response time. The variations in workload will usually include emulation of average and peak workloads that will occur within normal operational tolerances.

The following table gives the estimated average and peak workloads for the central TACHOnet system (see “TACHOnet Messages Load Estimates” – TCN-MessagesLoad-01\_12-EN.xls – for more details) of **incoming MS2TCN\_<TxName>\_Req messages per minute taking into account 22 connected Member States:**

MS2TCN_<TxName>_Req messages	TestIds	Online		Batch	
		Average	Peak (*3)	Average	Peak (*3)
CheckIssuedCards (first issue) → requires broadcasting to other Member States	Mix of 125, 126, 127	8	24	0,13	0,4
IssuedCardDL (first issue)	410	1	3	0,01	0,04
CheckIssuedCards (road check)	Mix of 110, 112	8	24	n.a.	n.a.
CheckCardStatus (road check)	Mix of 210, 211	15	45	n.a.	n.a.
ModCardStatus	310	1	3	n.a.	n.a.
Total	Total	33/min.	99/min.	0,14/min.	0,44/min.
			1,65/sec.		

**Table 1 – Load Testing with 22 connected Member States**

These estimated numbers are only valid for the central TACHOnet system based on the estimated workload of **MS2TCN\_<TxName>\_Req messages** sent by 22 connected Member States. Member States systems are obviously less loaded.

Pay attention to the fact that such incoming **MS2TCN\_<TxName>\_Req** messages will generate lots of other messages (**TCN2MS\_<TxName>\_Req, MS2TCN\_<TXName>\_Res,...**) that should also be processed by the central TACHOnet system, ending up with the following estimated global workload of incoming/outgoing messages to be processed per second (for 22 Member States):

- Average: 7,6 messages per second (in and out)
- Peak: 23 messages per second

*Continued on next page*

## Pilot Test Phase, Continued

### Load Testing (continued)

It's important to note that more than 75% of the estimated global workload concerns the *CheckIssuedCards* transaction at first issue, because this transaction involves broadcasting the messages to all other Member States. Therefore, it's of major importance to take that information into account when building scenarios for load testing, especially when the expected 22 connected Member States will not be available for such load testing.

Indeed, the following formula is used to estimate the number of incoming/outgoing messages for the *CheckIssuedCards* transaction at first issue (requiring broadcasting):

<b>2 * T</b>	+	<b>2 * T *(N - 1)</b>
means 1 MS2TCN_CheckIssuedCards_Req + 1 TCN2MS_CheckIssuedCards_Res messages per transaction T.		means 1 TCN2MS_CheckIssuedCards_Req + 1 MS2TCN_CheckIssuedCards_Res messages per transaction T times N -1 (where N is the number of connected Member States)

This formula can be refactored to **2 \* T \* N**.

According the estimated workload for the *CheckIssuedCards* at first issue for 22 Member States (see table at preceding page), the estimated numbers of incoming/outgoing messages supported by the central TACHOnet system are:

- Average: 352 messages per minute (2\*T\*N ≅ 2\*8\*22) or 5,86 msgs/sec.
- Peak (\*3): 1056 messages per minute or 17,6 msgs/sec

As mentioned above, these numbers are only valid provided 22 Member States are actually connected to the TACHOnet system. Obviously, load testing with 22 connected Member States won't be possible.

Simulating such load with less connected Member States is difficult and will not give accurate results since the number of transactions must be increased (*MS2TCN\_CheckIssuedCards\_Req* messages) to compensate the decrease in the number of broadcasted messages (*TCN2MS\_CheckIssuedCards\_Req* messages). Indeed, to simulate such load with less connected Member States, the number of transactions (T) must be increased by a factor equivalent to the decrease in the number of connected Member States (N). The following table gives the new number of transactions T taking into account the number of Member States (T) actually connected and participating to the load testing:

If N equals...	Then T should be...	To support...
22	8 (per minute)	352 messages per minute
6 (e.g. FL, I, F, E, S, NL)	29 (per minute)	352 messages per minute
5 (e.g. FL, I, F, E, S)	35 (per minute)	352 messages per minute
4 (e.g. FL, I, F, E)	44 (per minute)	352 messages per minute

*Continued on next page*



## Pilot Test Phase, Continued

### Load Testing (continued)

If N equals...	Then T should be...	To support...
3 (e.g. FL, I, F)	59 (per minute)	352 messages per minute
2 (e.g. FL and I)	88 (per minute)	352 messages per minute

Moreover, one should not forget that the incoming *MS2TCN\_x\_Req* messages should have *TCN\_FL* (Liechtenstein) as *From* attribute value.

In the first load & stress testing phase, Italy did propose to generate and send the initial *MS2TCN\_x\_Req* messages to reach the expected amount of transactions (see table above) taking into account the number of actually connected Member States. As more likely only 5 Member States will be connected (including simulated FL), Italy should then generate and send:

<i>MS2TCN_&lt;TxName&gt;_Req</i> messages	<i>TestIds</i>	Online		Batch	
		Average	Peak (*3)	Average	Peak (*3)
CheckIssuedCards (first issue) → requires broadcasting to other Member States	Mix of 125, 126, 127	35	105	0,5	1,5
IssuedCardDL (first issue)	410	1	3	0,01	0,04
CheckIssuedCards (road check)	Mix of 110, 112	8	24	n.a.	n.a.
CheckCardStatus (road check)	Mix of 210, 211	15	45	n.a.	n.a.
ModCardStatus	310	1	3	n.a.	n.a.
Total	Total	60/min. 1/sec.	180/min. 3/sec.	0,14/min.	0,44/min.

**Table 2 – Load Testing with 5 Member States (simulating load of 22 connected MS)**

**Important:**

- Do not forget to use *TCN\_FL* (Liechtenstein) as *From* attribute value in the first issue's messages (to include as many connected Member States in the broadcast process).
- Do not forget to use as much as possible connected Member States as *IssuingMemberStateCode* attribute value in the other messages to send requests to these connected Member States (acting as connected data providers).
- As the current test environment is not equipped to support the peak load (another server would be required – as planned in production), please only test the **average** load. Some extrapolation could be done afterwards.

*Continued on next page*

## Pilot Test Phase, Continued

---

### Load Testing (continued)

Such load testing should be performed during a couple of hours. Performance counters must be set on the TACHOnet central system (web server, BizTalk server, SQL Server) but also at network level (to be performed by TESTA people).

---

### Stress Testing

Stress testing focuses on evaluating how the system responds under abnormal conditions. Stresses on the system may include extreme workloads, insufficient memory, unavailable services and hardware, or limited shared resources. These tests are often performed to gain a better understanding of how and in what areas the system will break, so that contingency plans and upgrade maintenance can be planned and budgeted for well in advance.

Due to the current configuration of the test environment available at the data center (single application server, single database server, no clustering, no load balancing), it's rather difficult to stress test the central TACHOnet system under extreme conditions (e.g. one server down in a two server node configuration,...). This could be planned once the production servers are available, prior to going live.

Anyway, some tests might be performed to assess and extrapolate the system behaviour in the following different situations:

- Testing the system under normal conditions but over a long period of time (e.g. 2 or 3 days) to check how system resources (memory, disk space, network bandwidth,...) are consumed over time. Such kind of tests can identify potential memory leak problems or unexpected disk space usage (logging,...) and should be run automatically during the week-end (24 x 7) by sending the same kinds of *MS2TCN\_<TxName>\_Req* messages as for load testing under **average** conditions (see **Table 2** for more details) but for a longer period. Performance counters must be set on the TACHOnet central system (web server, BizTalk server, SQL Server) but also at network level (to be performed by TESTA people).
  - Testing the system under extreme conditions to identify the bottlenecks of the system so that contingency plans and upgrade can be planned. Such test should be run during 1 or 2 hours by sending the same kinds of *MS2TCN\_<TxName>\_Req* messages as for load testing (see **Table 2**) starting from **average** conditions and by increasing these peak conditions by a 10% factor till the system breaks. Again, performance counters must be set on the TACHOnet central system (web server, BizTalk server, SQL Server) but also at network level (to be performed by TESTA people).
- 

*Continued on next page*

## Pilot Test Phase, Continued

---

### Performance counters

The following table lists all the performance counters that will be captured during load and stress testing:

- DISK
  - Physical Disk\% Disk Time
  - Physical Disk\Current Disk Queue Length
  - Physical Disk\Disk Reads/Sec
  - Physical Disk\Disk Writes/Sec
  - The sum of the last two is equivalent to Transfers/Sec
- Memory
  - Memory\Available Bytes
  - Memory\Page Reads/sec
  - Memory\Page Input/sec
- Processor
  - Processor\%Processor Time
  - System\Processor Queue Length
- Network (Only within the LAN – TESTA load should be monitored by TESTA people)
  - Network Interface\Bytes Total/Sec
  - Network Interface\Output Queue Length
- IIS
  - Web Service\Post Requests/Sec - This will measure documents going in
  - Process\Private Bytes(InetInfo.exe) - For memory Leaks
  - Process\Private Bytes(Total) - For Memory Leaks
  - Process\Private Bytes(DllHost) - For Memory Leaks
  - Process\Private Bytes(DllHost#...n) - For Memory Leaks
- Microsoft Message Queue
  - Nothing to Monitor, since we don't use it
- XLANG
  - Nothing to Monitor, since we don't use it
- BizTalk
  - BizTalk Server\Synchronous Submissions/sec
  - BizTalk Server\Asynchronous Submissions/sec
  - BizTalk Server\Documents Received/sec
  - BizTalk Server\Documents Processed/sec
  - System\Context Switches/sec
- SQL
  - This will come from NETIQ diagnostics manager which will provide a comprehensive list statistics, such as DB Growth, Recompilations, Log File Growth, Sessions, Procedure Cache, Buffer Manager, so there will be no issues in regards to SQL statistics.

---

*Continued on next page*

## Pilot Test Phase, Continued

---

**Performance  
counters**  
(continued)

Log files will be generated separately for each type of counter i.e. Disk File, Memory File, Biz File

Max of 2MB per File then a new file (Reduce losing data from a corrupt CSV)

Interval will be 1 Min between collections.

---

## Chapter 5: Test Scenarios

### Overview

---

#### Introduction

This section provides all test scenarios deduced from the use cases as defined in the Software Requirements Specification document. When talking about use cases we will include business use cases also because the majority of tests that use or require an interface will be business cycle tests and not function tests.

---

#### Contents

This chapter contains the following topics:

Topic	See Page
Logon to TCN Central System	43
Get Phonex Search Keys	47
Get US/Ascii transliteration	51
Check driver's issued card	54
Check tachograph card status	58
Declaration of card status modification	62
Send Card/Driving License assignment	70
Create Card – first issue	74
Card Renewal	78
Card Exchange	82
Card Replacement	84
Bulk check of issued driver card holders	88
Browse MS Statistics	89
Browse TCN Statistics	90
Monitor message exchange	92
Manage Users	94
Volume and Stress Testing	95

---

# Logon

---

**Use Case  
Description**

Tachonet provides statistics on the message exchange between member states. Generation of lists and charts is available only to CiA Admins and TCN Admins and as such require a logon procedure.

---

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-0051	CiA Admin login
TC-0052	TCN Admin login
TC-0053	CiA App user login – out of scope (depending on local implementation)

---

**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S0051-01	CiA Admin login – normal flow
S0051-02	CiA Admin login – connection lost
S0051-03	CiA Admin login – processing timed-out
S0051-04	CiA Admin login – access denied

---

Test Id	Description
S0052-01	TCN Admin login – normal flow
S0052-02	TCN Admin login – connection lost
S0052-03	TCN Admin login – processing timed-out
S0052-04	TCN Admin login – access denied

---

**S0051-01** CiA Admin login to TCN – normal flow  
**S0052-01** TCN Admin login to TCN – normal flow

Step Number	Action to perform	Expected result
Step1a	CiA Admin logs on (id and password)	
Step1b	TCN Admin logs on (id and password)	
Step2	TCN WUI converts entered data into XML message	
Step3	TCN XMS validates the XML message	XML message is validated against XML schema / well-formed + valid
Step4	TCN XMS processes the contents of the XML message	request processed/login passed
Step5	TCN XMS sends back XML message with status code OK	status message indicates login successful
Step6	TCN WUI receives XML message with status code OK	XML message is validated against XML schema / well-formed + valid
Step7	TCN WUI interprets XML message	UI is rendered according to user's role and access rights

**S0051-02** CiA Admin login – connection lost (3a)  
**S0051-03** CiA Admin login – processing timed-out (3c)  
**S0051-04** CiA Admin login – access denied (3b)

Step Number	Action to perform	Expected result
Step1	CiA Admin logs on (id and password)	
Step2	TCN WUI converts entered data into XML message	
Step3a	TCN XMS does not receive XML message	connection is broken
Step3b/c	TCN XMS validates the XML message	XML message is validated against XML schema / well-formed + valid
Step4a	TCN WUI indicates no connection with server possible	
Step4b	TCN XMS processes the contents of the XML message	request processed/login failed
Step4c	TCN XMS processes the contents of the XML message	request not processed / timed out
Step5b	TCN XMS sends back XML message with status code Access Denied	
Step5c	TCN XMS sends back XML message with status code Timeout	
Step6b	TCN WUI receives XML message with status code AccessDenied	
Step6c	TCN WUI receives XML message with status code TimedOut	

- 
- S0052-02** TCN Admin login to TCN - connection lost (as S0051-02)
  - S0052-03** TCN Admin login to TCN - processing timed-out (as S0051-03)
  - S0052-04** TCN Admin login to TCN - access denied (as S0051-04)
-



## Get Phonex Search Keys

---

**Use Case  
Description**

Based on a given first and last name, obtain from TCN (or a local equivalent) the computed search keys (based on the Phonex algorithm).  
Because this particular use case is incorporated into several other business use cases it is imperative that each member state who does not use the TCN-offered Phonex web service should implement the Phonex algorithm in its own CiA application.

---

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-0451	Enforcer requesting Phonex search keys – TCN web service
TC-0452	CiA App user requesting Phonex search keys – TCN web service
TC-0453	CiA application requesting Phonex search keys – TCN web service

---

**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S0451-01	Enforcer requesting Phonex search keys – TCN web service – normal flow
S0451-02	Enforcer requesting Phonex search keys – TCN web service – invalid input

Test Id	Description
S0452-01	CiA App requesting Phonex search keys – TCN web service – normal flow
S0452-02	CiA App requesting Phonex search keys – TCN web service – invalid input

Test Id	Description
S0453-01	CiA application requesting Phonex search keys – TCN web service – normal flow
S0453-02	CiA application requesting Phonex search keys – TCN web service – invalid input

---

*Continued on next page*

## Get Phonex Search Keys, continued

**S0451-01** Enforcer requesting Phonex search keys – TCN web service – normal flow

Step Number	Action to perform	Expected result
step1	S0051-05	Web user logged on as Enforcer
step2	Web user select Phonex functionality	Web service is initiated, UI is presented
step3	Web user enters first name(s) and last name	Input entered by following the name spelling rules
step4	Web service generates and presents Phonex search keys	

**S0451-02** Enforcer requesting Phonex search keys – TCN web service – invalid info

Step Number	Action to perform	Expected result
step1	S0051-05	Web user logged on as Enforcer
step2	Web user select Phonex functionality	Web service is initiated, UI is presented
step3	Web user enters first name(s) and last name	Input entered is not respecting the name spelling rules
step4	Web service validates input	Invalid input, error message generated
step5	Web service presents error message	

**S0452-01** CiA App user requesting Phonex search keys – TCN web service – normal flow (same as S0451-01)

**S0452-02** CiA App user requesting Phonex search keys – TCN web service – invalid info (same as S0451-02)

**S0453-01** CiA application requesting Phonex search keys – TCN web service – normal flow  
Same as S0451-01, replace step1 by S0052-01.

## Get Phonex Search Keys, continued

---

### S0453-02

CiA application requesting Phonex search keys – TCN web service – invalid input  
Same as S0451-02, replace step1 by S0052-01.

---

## Get US/Ascii Transliteration

**Use Case Description** Based on the entered first and last name, obtain from TCN the US/Ascii (ISO 646 IRV) transliteration. Currently supported input can be a Latin-based language or Greek.

**Test Cases** Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-0461	User requesting US/Ascii transliteration – Latin-based source
TC-0462	User requesting US/Ascii transliteration – Greek source

**Test Scenarios** We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S0461-01	Enforcer requesting US/Ascii transliteration – Latin-based source – normal flow
S0461-02	Enforcer requesting US/Ascii transliteration – Latin-based source – invalid input
S0461-03	CiA App user requesting US/Ascii transliteration – Latin-based source – normal flow
S0461-04	CiA application requesting US/Ascii transliteration – Latin-based source – normal flow

Test Id	Description
S0462-01	Enforcer requesting US/Ascii transliteration – Greek source – normal flow
S0462-02	Enforcer requesting US/Ascii transliteration – Greek source – invalid input
S0462-03	CiA App user requesting US/Ascii transliteration – Greek source – normal flow
S0462-04	CiA application requesting US/Ascii transliteration – Green source – normal flow

## Get US/Ascii Transliteration, continued

**S0461-01** Enforcer requesting US/Ascii transliteration – Latin-based source – normal flow

Step Number	Action to perform	Expected result
step1	S0051-05	Web user logged on as Enforcer
step2	Web user select transliteration functionality	Web service is initiated, UI is presented
step3	Web user enters first name(s) and last name found on drivers license	Latin-based info entered
step4	Web service generates and presents US/Ascii equivalent	

**S0461-02** Enforcer requesting US/Ascii transliteration – Latin-based source – invalid input

Step Number	Action to perform	Expected result
step1	S0051-05	Web user logged on as Enforcer
step2	Web user select transliteration functionality	Web service is initiated, UI is presented
step3	Web user enters first name(s) and last name found on drivers license	Input entered is not respecting the name spelling rules
step4	Web service validates input	Invalid input, error message generated
step5	Web service presents error message	

**S0461-03** CiA App user requesting US/Ascii transliteration – Latin-based source – normal flow  
Same as S0461-01 with step1

**S0461-04** CiA application requesting US/Ascii transliteration – Latin-based source – normal flow  
Same as S0461-01 with step1

**S0462-01** Enforcer requesting US/Ascii transliteration – Greek source – normal flow  
Same as S0461-01 with step3 entering Greek info

---

**S0462-02**

Enforcer requesting US/Ascii transliteration – Greek source – invalid input  
Same as S0461-02 with step3 entering Greek info

---

**S0462-03**

CiA App user requesting US/Ascii transliteration – Greek source – normal flow  
Same as S0461-03 with step3 entering Greek info

---

**S0462-04**

CiA application requesting US/Ascii transliteration – Greek source – normal flow  
Same as S0461-04 with step3 entering Greek info

---

# Check driver's issued card

**Business Use Case**

This use case reflects the sequence of activities performed by an enforcer when checking out a driver who is for instance claiming he lost his card.

**Description**

The enforcer will fill in a web form (provided by a web application developed under the Member State's responsibility) with the driver's details (at least the driver's surname, first name(s) and date of birth) and the code of the Member State from which the driver claims having got his card.

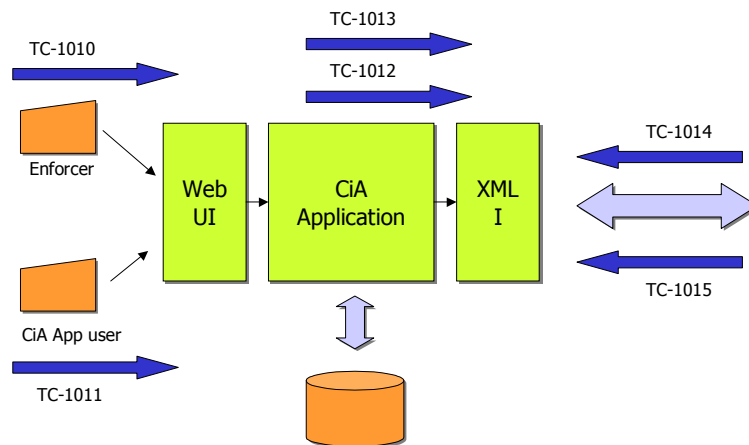
The web form will be submitted to the CIA application for checking whether the driver does actually hold a valid card locally or in another member state.

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1010	Enforcer checking driver's issued card
TC-1011	CiA App user checking driver's issued card
TC-1012	CiA application checking driver's issued card – single instance
TC-1013	CiA application checking driver's issued card – batch treatment
TC-1014	CiA application checking driver's issued card – incoming request
TC-1015	CiA application checking driver's issued card – incoming batch

**Test Cases Visualization**



**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
<i>S1010-xx</i>	<i>Enforcer checking existence &amp; status of a DriverCard</i>
<i>S1010-01</i>	<i>local DL/local DC – out of scope</i>
<i>S1010-02</i>	<i>non-local DL/local DC – out of scope</i>
S1010-03	local DL/non-local DC – send out to 1 MS
S1010-04	non-local DL/non-local DC – send out to 1 MS
S1010-05	local DL/non-local DC – broadcast towards all MS
S1010-06	non-local DL/non-local DC – broadcast towards all MS
<del>S1010-07</del>	<del>S1010-01 – no local DC – 1 MS</del>
S1010-08	Initiate S1010-01 but no local DC found – continue with S1010-05
S1010-10	S1010-04 with transliteration
S1010-11	Driver not found in foreign database
S1010-12	Phonex search keys cover multiple Drivers
S1010-13	DriverCard can not be found
S1010-14	Driver License can not be found or is not supported
S1010-15	Request timed out at one or more MS
S1010-16	ServerError due to loss of connection at one or more MS

Test Id	Description
<i>S1011-xx</i>	<i>CiA App user checking existence &amp; status of a DriverCard</i>
S1011-03	local DL/non-local DC – send out to 1 MS
S1011-04	non-local DL/non-local DC – send out to 1 MS
S1011-05	local DL/non-local DC – broadcast towards all MS
S1011-06	non-local DL/non-local DC – broadcast towards all MS
<del>S1011-07</del>	<del>S1011-01 – no local DC – 1 MS</del>
S1011-08	S1011-01 - no local DC - All MS
S1011-10	S1011-04 with transliteration

Test Id	Description
<i>S1012-xx</i>	<i>CiA application checking existence &amp; status of a DriverCard</i>
S1012-03	local DL/non-local DC – send out to 1 MS
S1012-04	non-local DL/non-local DC – send out to 1 MS
S1012-05	local DL/non-local DC – broadcast towards all MS
S1012-06	non-local DL/non-local DC – broadcast towards all MS
S1012-10	S1012-04 with transliteration

Test Id	Description
<i>S1013-xx</i>	<i>CiA App checking batch of DriverCard</i>
S1013-03	local DL/non-local DC – grouped by and send out to 1 MS
S1013-04	non-local DL/non-local DC grouped by and send out to 1 MS
S1013-05	local DL/non-local DC – broadcast towards all MS
S1013-06	non-local DL/non-local DC – broadcast towards all MS
S1013-07	Response will combine multiple S1014-03/04/05/06 cases
S1013-08	Response will combine S1014-01/02 with S1014-07
S1013-09	Response will combine S1014-01/02 with S1014-08

Test Id	Description
<i>S1014-xx</i>	<i>CiA App checking existence &amp; status of a DriverCard for other MS</i>
S1014-01	local DL/local DC – Driver found
S1014-02	non-local DL/local DC – Driver found
S1014-03	Driver not found in local database
S1014-04	Phonex search keys cover multiple Drivers



S1014-05	DriverCard can not be found
S1014-06	Driver License can not be found or is not supported
S1014-07	Request timed out
S1014-08	Request did not arrive or local server problem

Test Id	Description
<i>S1015-xx</i>	<i>CiA App checking batch of DriverCards for another MS</i>
S1015-01	local DL/local DC
S1015-02	non-local DL/local DC
S1015-03	Combining multiple S1014-03/04/05/06 cases
S1015-04	Combining S1014-01/02 with S1014-07
S1015-05	Combining S1014-01/02 with S1014-08

**User Input variants**

As indicated in the above-mentioned test cases, this use case includes the Get Phonex Search keys and US/Ascii transliteration use cases. The first is considered to be initiated always while the latter could be initiated, creating as such different variants for Latin-based and Greek input.

**Remark on Test Ids**

Test scenarios -11 through -16 are not repeated for CiA App and CiA App user because they do not test a new aspect or functionality that an actor can introduce through an interface.

Test scenarios –11 through –16 are dealing with the kind of response expected from the central system (or other MS) or cover non-standard behavior of the solution.

**Sending and receiving end**

As indicated by the diagram on page 52, there is a link between test cases TC1010/11/12 and TC1014 as there is a link between TC1013 and TC1015. So keep in mind when receiving an S1014-xx the other side is actually sending an S1010-xx.

## Check driver's issued card, continued

- S1010-03** Enforcer checking driver's issued card – local Driver's license/non-local DriverCard – 1 MS (step2a)
- S1010-04** Enforcer checking driver's issued card – non-local Driver's license/non-local DriverCard – 1 MS (step2a)
- S1010-05** Enforcer checking driver's issued card – local Driver's license/non-local DriverCard – All MS (step2b)
- S1010-06** Enforcer checking driver's issued card – non-local Driver's license/non-local DriverCard – All MS (step2b)
- S1010-10** S1010-04 with transliteration (step3a or 3b)

Step Number	Action to perform	Expected result
step1	local CiA App interface is presented	
step2a	user enters available information (name, driver license, birth date)	Issuing member state = all
step2b	user enters available information (name, driver license, birth date)	Issuing member state = a ms
step3a	S0461-01	Latin – US transliteration is done
step3b	S0462-01	Greek – US transliteration is done
step3c	No transliteration required	
step4	S0451-xx	Phonex search keys are generated
step5	user selects submit	CiA App verifies mandatory fields
step6	CiA App sends out xml message to TCN	TCN validates xml message
step7	TCN stores request	
step8a	TCN sends out xml message to other MS	MS validates xml message
step8b	TCN sends out xml message to another MS	MS validates xml message
step9a/b	TCN receives response xml message from MS	TCN validates xml message
step10a	TCN stores responses	all ms responses are grouped into one message
step10b	TCN stores response	
step11	TCN forwards response xml message to CiA App	CiA App validates xml message

step12a	local CiA App presents driver and driver card details	card status is shown too
step12b	local CiA App presents driver details and absence of driver card details	

**S1010-07** Enforcer checking driver's issued card – local Driver's license/local DriverCard – 1 MS  
**S1010-08** Enforcer checking driver's issued card – local Driver's license/local DriverCard – All MS

Step Number	Action to perform	Expected result
step1	local CiA App interface is presented	
step2	user enters available information (name, driver license, birth date)	Issuing member state = a ms
step3a	S0461-01	Latin – US transliteration is done
step3b	S0462-01	Greek – US transliteration is done
step3c	No transliteration required	
step4	S0451-xx	Phonex search keys are generated
step5	user selects submit	CiA App verifies mandatory fields
step6	local CiA App searches for DriverCard	
step7	No DriverCard found locally	
step8	CiA App sends out xml message to TCN	TCN validates xml message
step9	TCN stores request	
step10	TCN sends out xml message to another MS	MS validates xml message
step11	TCN receives response xml message from MS	TCN validates xml message
step12	TCN stores response	
step13	TCN forwards response xml message to CiA App	CiA App validates xml message
step14a	local CiA App presents driver and driver card details	card status is shown too
step14b	local CiA App presents driver details and absence of driver card details	

**S1011-03** CiA App user checking existence & status of a DriverCard - local DL/non-local DC – 1 MS  
 Same as S1010-03  
**S1011-04** CiA App user checking existence & status of a DriverCard - non-local DL/non-local DC – 1 MS  
 Same as S1010-04

- S1011-05** CiA App user checking existence & status of a DriverCard - local DL/non-local DC - All MS  
Same as S1010-05
- S1011-06** CiA App user checking existence & status of a DriverCard - non-local DL/non-local DC - All MS  
Same as S1010-06
- S1011-10** S1011-04 with transliteration

- S1011-07** S1011-01 - no local DC - 1 MS  
Same as S1010-07
- S1011-08** S1011-01 - no local DC - All MS  
Same as S1010-08

- S1013-03** CiA App checking batch of DriverCard - local Driver's license/non-local DriverCard - 1 MS
- S1013-04** CiA App checking batch of DriverCard - non-local Driver's license/non-local DriverCard - 1 MS
- S1013-05** CiA App checking batch of DriverCard - local Driver's license/non-local DriverCard - All MS
- S1013-06** CiA App checking batch of DriverCard - non-local Driver's license/non-local DriverCard - 1 MS
- S1013-10** S1013-04 with transliteration

Step Number	Action to perform	Expected result
step1	Batch file containing separate drivers to investigated is created	
step2a	For each driver S0461-01 or	Latin - US transliteration is done
step2b	S0462-01 is executed or	Greek - US transliteration is done
step2c	none at all	
step3	For each driver S0451-xx is executed	Phonex search keys are generated
step4	CiA App sends out xml message to TCN	TCN validates xml message
step5	TCN stores request	
step6a	For each driver TCN sends out a separate xml message to other MS	MS validates xml message
step6b	For each driver TCN sends out a separate xml message to another MS	MS validates xml message
step7a	For each driver TCN receives response xml message from MS	
step7b	For each driver TCN receives response xml messages from different MS	TCN validates xml message
step8	TCN stores response(s)	all ms responses are grouped into one message

step9	TCN forwards response xml message to CiA App	CiA App validates xml message
step10	CiA App stores response for further use	

**S1014-01** CiA App checking existence & status of a DriverCard for other MS - local DL/local DC (step3a)  
**S1014-02** CiA App checking existence & status of a DriverCard for other MS - non-local DL/local DC (step3b)

Step Number	Action to perform	Expected result
step1	CiA App receives xml message from TCN	CiA App validates xml message
step2	CiA App searches for DriverCard based on provided Phonex search keys	
step3a	CiA App sends out response xml message to TCN	DriverCard found, driver details found
step3b	CiA App sends out response xml message to TCN	DriverCard found, no driver details found
step3c	CiA App sends out response xml message to TCN	no DriverCard found or issued
step4	TCN receives xml message from CiA App	TCN validates xml message
step5	TCN stores response	
step6	TCN forwards response xml message to other CiA App	

**S1015-01** CiA App checking batch of DriverCards for another MS - local DL/local DC  
**S1015-02** CiA App checking batch of DriverCards for another MS - non-local DL/local DC

Step Number	Action to perform	Expected result
step1	CiA App receives xml message batch from TCN	CiA App validates xml message
step2	CiA App searches for each DriverCard based on provided Phonex search keys	
step3a	CiA App creates response xml message	DriverCard found, driver details found
step3b	CiA App creates response xml message to TCN	DriverCard found, no driver details found
step3c	CiA App creates response xml message to TCN	no DriverCard found or issued
step4	CiA App groups all responses into one xml message	
step5	CiA App sends out response xml message to TCN	

---

step6	TCN receives xml message from Cia App	TCN validates xml message
step7	TCN stores response	
step8	TCN forwards response xml message to other Cia App(s)	

## Check tachograph card status

### Business Use Case

This use case reflects the sequence of activities performed by an enforcer when checking out the status of a driver’s card.

### Description

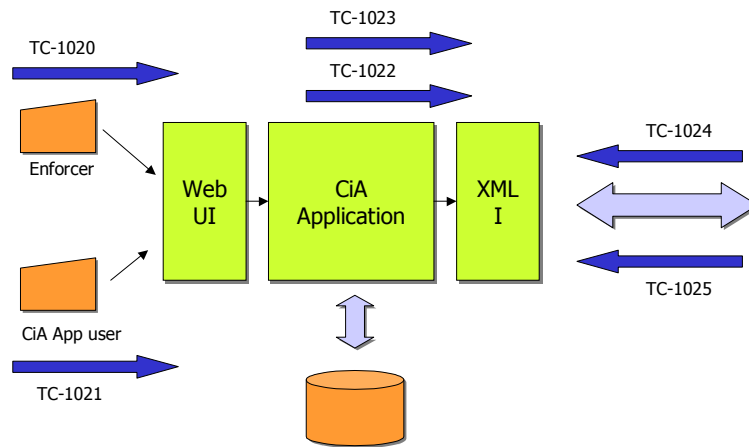
The enforcers may also use this process to check the validity of the workshop card used by the manufacturer to test and calibrate the recording equipment.

### Test Cases

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1020	Enforcer checking driver’s card status – single instance
TC-1021	CiA App user checking driver’s card status – single instance
TC-1022	CiA application checking driver’s card status – single instance
TC-1023	CiA application checking driver’s card status – batch treatment
TC-1024	CiA application checking driver’s card status – incoming request
TC-1025	CiA application checking driver’s card status – incoming batch request

### Test Cases Visualization



### Test Scenarios

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S1020-xx	Enforcer checking status of a tachograph card
S1020-01	local DL/local DC – out of scope

S1020-02	<i>non-local DL/local DC – out of scope</i>
S1020-03	local DL/non-local DC - send out to 1 MS
S1020-04	non-local DL/non-local DC - send out to 1 MS
S1020-10	S1020-04 with transliteration
S1020-11	Driver not found in foreign database
S1020-12	Phonex search keys cover multiple Drivers
S1020-13	DriverCard can not be found
S1020-14	Driver License can not be found or is not supported
S1020-15	Request timed out at one or more MS
S1020-16	ServerError due to loss of connection at one or more MS

Test Id	Description
<i>S1021-xx</i>	<i>CiA App user checking status of a tachograph card</i>
S1021-01	local DL/local DC – out of scope
S1021-02	non-local DL/local DC – out of scope
S1021-03	local DL/non-local DC – send out to 1 MS
S1021-04	non-local DL/non-local DC – send out to 1 MS
S1021-10	S1021-04 with transliteration

Test Id	Description
<i>S1022-xx</i>	<i>CiA App checking status of a tachograph card</i>
S1022-03	local DL/non-local DC – send out to 1 MS
S1022-04	non-local DL/non-local DC – send out to 1 MS
S1022-10	S1022-04 with transliteration

Test Id	Description
<i>S1023-xx</i>	<i>CiA App checking status of batch of tachograph card</i>
S1023-03	local DL/non-local DC – send out to 1 MS
S1023-04	non-local DL/non-local DC – send out to 1 MS
S1023-05	Response will combine multiple S1024-03/04/05/06 cases
S1023-06	Response will combine S1024-01/02 with S1024-07
S1023-07	Response will combine S1024-01/02 with S1024-08

Test Id	Description
<i>S1024-xx</i>	<i>CiA App checking status of a local DriverCard for other MS</i>
S1024-01	local DL - Driver found
S1024-02	non-local DL - Driver found
S1024-03	Driver not found in local database
S1024-04	Phonex search keys cover multiple Drivers
S1024-05	DriverCard can not be found
S1024-06	Driver License can not be found or is not supported
S1024-07	Request timed out
S1024-08	Request did not arrive or local server problem

Test Id	Description
<i>S1025-xx</i>	<i>CiA App checking batch of a local DriverCard status for other MS</i>
S1025-01	local DL
S1025-02	non-local DL
S1025-03	Combining multiple S1024-03/04/05/06 cases
S1025-04	Combining S1024-01/02 with S1024-07
S1025-05	Combining S1024-01/02 with S1024-08



**User Input variants**

As indicated in the above-mentioned test cases, this use case re-uses the Get Phonex Search keys and US/Ascii transliteration. The first is considered to be initiated always while the latter could be initiated, creating as such different variants for Latin-based and Greek input.

---

**Remark on Test Ids**

Test scenarios -11 through -16 are not repeated for CiA App and CiA App user because they do not test a new aspect or functionality that an actor can introduce through an interface.

Test scenarios –11 through –16 are dealing with the kind of response expected from the central system (or other MS) or cover non-standard behavior of the solution.

---

**Sending and receiving end**

As indicated by the diagram on page 60, there is a link between test cases TC1020/21/22 and TC1024 as there is a link between TC1023 and TC1025. So keep in mind when receiving an S1024-xx the other side is actually sending an S1020-xx.

---

## Check tachograph card status, continued

- S1020-03** Enforcer checking status of a tachograph card – local driver license/non-local driver card
- S1020-04** Enforcer checking status of a tachograph card - non-local driver license/non-local driver card
- S1020-10** S1020-04 with transliteration (3a/3b)

Step Number	Action to perform	Expected result
step1	local CiA App interface is presented	
step2	user enters available information	Driver card n° and issuing nation entered
step3a	S0461-01	Latin – US transliteration is done
step3b	S0462-01	Greek – US transliteration is done
step4	user selects submit	CiA App verifies mandatory fields
step5	CiA App sends out xml message to TCN	
step6	TCN stores request	
step7	TCN sends out xml message to other MS (issuing state)	MS validates xml message
step8	TCN receives response xml message from MS	TCN validates xml message
step9	TCN stores response(s)	all ms responses are grouped into one message
step10	TCN forwards response xml message to CiA App	CiA App validates xml message
step11	CiA App presents driver, driver card and card status details	

- S1021-03** CiA App user checking status of a tachograph card - local DL/non-local DC  
Same as S1020-03

- S1021-04** CiA App user checking status of a tachograph card - non-local DL/non-local DC  
Same as S1020-04

- S1021-10** S1021-04 with transliteration

- S1022-03** CiA App checking status of batch of tachograph card - local DL/non-local DC
- S1022-04** CiA App checking status of a tachograph card - non-local DL/non-local DC
- S1022-10** S1022-04 with transliteration (2a/2b)

Step Number	Action to perform	Expected result
step1	CiA App receives request to check non-local driver card	Driver card n° and issuing nation are known
step2a	S0461-01	Latin – US transliteration is done
step2b	S0462-01	Greek – US transliteration is done
step3	CiA App generates request xml message	
step4	CiA App sends out xml message to TCN	
step5	TCN stores request	
step6	TCN sends out xml message to other MS (issuing state)	MS validates xml message
step7	TCN receives response xml message from MS	TCN validates xml message
step8	TCN stores response(s)	all ms responses are grouped into one message
step9	TCN forwards response xml message to CiA App	CiA App validates xml message

- S1023-03** CiA App checking status of batch of tachograph card - local DL/non-local DC
- S1023-04** CiA App checking status of batch of tachograph card - non-local DL/non-local DC
- S1023-10** S1023-04 with transliteration

Step Number	Action to perform	Expected result
step1	Batch file containing separate driver cards to investigated is created	
step2	CiA App sends out xml message to TCN	TCN validates xml message
step3	TCN stores request	
step4	For each driver's card TCN sends out a separate xml message to other MS (issuing state)	MS validates xml message
step5	For each driver's card TCN receives response xml messages from different MS	TCN validates xml message
step6	TCN stores response(s)	all ms responses are grouped into one message
step7	TCN forwards response xml message to CiA App	CiA App validates xml message

step8 CiA App stores response for further use

- S1024-01** CiA App checking status of a local DriverCard for other MS - local DL (2c)
- S1024-02** CiA App checking status of a local DriverCard for other MS - non-local DL (2b)
- S1024-03** CiA App checking status of a local DriverCard for other MS - DC not found (2a)
- S1024-04** CiA App checking status of a local DriverCard for other MS - DL not found (2b)

**Step Number Action to perform Expected result**

- step1 CiA App receives request to check local driver card Driver card n° and driver name is known
- step2a CiA App does not find the corresponding driver card
- step2b CiA App finds driver card locally but no driver license info available
- step2c CiA App finds driver card locally and driver license info available
- step3 CiA App sends out response xml message to TCN
- step4 TCN stores response
- step5 TCN sends out xml message to other MS (requesting state) MS validates xml message

- S1025-01** CiA App checking batch of a local DriverCard status for other MS - local DL
- S1025-02** CiA App checking batch of a local DriverCard status for other MS - non-local DL

**Step Number Action to perform Expected result**

- step1 CiA App receives batch of requests to check local driver card Driver card n° and driver name is known
- step2 For each driver card n° CiA App looks for driver card status
- step3 CiA App groups all responses into one xml message
- step4 CiA App sends out response xml message to TCN
- step5 TCN stores response
- step6 TCN sends out xml message to other MS (requesting state) MS validates xml message



## Declaration of card status modification

**Use Case Description**

This use case actually covers two other use cases that could be executed in sequence. First the update of an existing driver’s card locally, followed by the declaration of the modification towards Tachonet so another member states can update its driver card database.

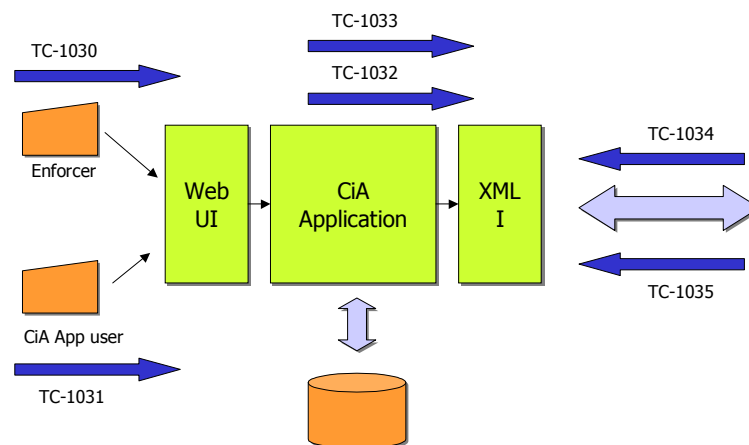
In order to declare a change in card status towards Tachonet, the enforcer or CiA application must be able to check the existence of a driver’s card (TC-101x), must be able to view the current driver’s card status (TC-102x) and must be able to change the current driver’s card status (UC-0200).

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1030	Enforcer declaring driver’s card status change – single instance
TC-1031	CiA App user declaring driver’s card status change – single instance
TC-1032	CiA application declaring driver’s card status change – single instance
TC-1033	CiA application declaring driver’s card status change – batch treatment
TC-1034	CiA application performs driver’s card status change – incoming request
TC-1035	CiA application performs driver’s card status change – incoming batch

**Test Cases Visualization**



**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
<i>S1030-xx</i>	<i>Enforcer declares card status modification towards local CiA App</i>
S1030-03	local DL/non-local DC
S1030-04	non-local DL/non-local DC
S1030-10	S1030-04 with transliteration

Test Id	Description
<i>S1031-xx</i>	<i>CiA App user declares card status modification towards TCN</i>
S1031-03	local DL/non-local DC – send out to 1 MS
S1031-04	non-local DL/non-local DC – send out to 1 MS
S1031-10	S1031-04 with transliteration
S1031-11	Driver not found in foreign database
S1031-13	DriverCard can not be found
S1031-15	Request timed out
S1031-16	ServerError due to loss of connection

Test Id	Description
<i>S1032-xx</i>	<i>CiA App declares card status modification towards TCN</i>
S1032-03	local DL/non-local DC – send out to 1 MS
S1032-04	non-local DL/non-local DC – send out to 1 MS
S1032-10	S1032-04 with transliteration

Test Id	Description
<i>S1033-xx</i>	<i>CiA App declares batch of card status modifications towards TCN</i>
S1033-03	local DL/non-local DC – send out to one or more MS
S1033-04	non-local DL/non-local DC – send out to one or more MS
S1033-05	Response will combine multiple S1034-03/04/05/07 cases
S1033-06	Response will combine S1034-03/04 with S1034-09
S1033-07	Response will combine S1034-03/04 with S1034-10

Test Id	Description
<i>S1034-xx</i>	<i>CiA App performs card status modification received from TCN</i>
S1034-03	local DL
S1034-04	non-local DL
S1034-05	Driver not found in local database
S1034-07	DriverCard can not be found
S1034-09	Request timed out
S1034-10	Request did not arrive or local server problem

Test Id	Description
<i>S1035-xx</i>	<i>CiA App performs card status modification by batch received from TCN</i>
S1035-03	local DL
S1035-04	non-local DL
S1035-05	Combining multiple S1034-03/04/05/07 cases
S1035-06	Combining S1034-03/04 with S1034-09
S1035-07	Combining S1034-03/04 with S1034-10

**User Input variants**

Besides the use cases being mentioned in the use case description, this use case re-uses also the Get Phonex Search keys and US/Ascii transliteration when dealing with a card status change where the card number is not known or available but driver name is. The first is considered to be initiated always while the latter could be initiated, creating as such different variants for Latin-based and Greek input.

A further variation is introduced by using a different card status (stolen, malfunctioning, suspended, found, etc)

---

**Remark on Test Ids**

Test scenarios -11 through -16 are not repeated for CiA App because they do not test a new aspect or functionality that an actor can introduce through an interface.

Test scenarios -11 through -16 are not repeated for Enforcer because the Enforcer himself can only change the card status locally. Depending on the local CiA App a declaration will be made (automatically) towards the central system.

Test scenarios -11 through -16 are dealing with the kind of response expected from the central system (or other MS) or cover non-standard behavior of the solution.

---

**Sending and receiving end**

As indicated by the diagram on page 66, there is a link between test cases TC1030/31/32 and TC1034 as there is a link between TC1033 and TC1035. So keep in mind when receiving an S1034-xx the other side is actually sending an S1030-xx.

---



## Declaration of card status modification, continued

- S1030-03** Enforcer declares card status modification towards local CiA App - local DL/non-local DC  
**S1030-04** Enforcer declares card status modification towards local CiA App - non-local DL/non-local DC  
**S1030-10** S1030-04 with transliteration(3a/3b)

Step Number	Action to perform	Expected result
step1	Local CiA App interface is presented	
step2	user enters available information	
step3a	S0461-01	Latin – US transliteration is done
step3b	S0462-01	Greek – US transliteration is done
step4	S0451-xx	Phonex search keys are generated
step5	User modifies driver card status and submits change to CiA App CiA App sends out xml message to TCN declaring driver's card status change	TCN validates xml message
step7	TCN stores request	
step8	TCN sends out xml message to other MS (DC issuing state)	MS validates xml message
step9	TCN receives status change acknowledgement message from MS	TCN validates xml message
step10	TCN stores response	
step11	TCN forwards response xml message to CiA App	CiA App validates xml message
step12	CiA App presents card status change acknowledgement	

- S1031-03** CiA App user declares card status modification towards TCN - local DL/non-local DC  
 Same as S1030-03  
**S1031-04** CiA App user declares card status modification towards TCN - non-local DL/non-local DC  
 Same as S1030-04  
**S1031-10** S1031-04 with transliteration  
 Same as 1030-10

- S1032-03** CiA App declares card status modification towards TCN - local DL/non-local DC
- S1032-04** CiA App declares card status modification towards TCN - non-local DL/non-local DC
- S1032-10** S1031-04 with transliteration (2a/2b)

Step Number	Action to perform	Expected result
step1	CiA App receives request to declare card status modification	Driver name and driver card n° known
step2a	S0461-01	Latin – US transliteration is done
step2b	S0462-01	Greek – US transliteration is done
step3	S0451-xx	Phonex search keys are generated
step4	CiA App sends out xml message to TCN declaring driver's card status change	TCN validates xml message
step5	TCN stores request	
step6	TCN sends out xml message to other MS (DC issuing state)	MS validates xml message
step7	TCN receives status change acknowledgement message from MS	TCN validates xml message
step8	TCN stores response	
step9	TCN forwards response xml message to CiA App	CiA App validates xml message

- S1033-04** CiA App declares batch of card status modifications towards TCN - non-local DL/non-local DC
- S1033-10** S1033-04 with transliteration (2a/2b)

Step Number	Action to perform	Expected result
step1	Batch file containing separate driver card status changes to declare is created	
step2a	For each driver/driver card status change S0461-01 or S0462-01 is executed or none at all	Latin – US transliteration is done
step2b		Greek – US transliteration is done
step3	For each driver/driver card status change S0451-xx is executed	Phonex search keys are generated
step4	CiA App sends out xml message to TCN	TCN validates xml message
step5	TCN stores request	
step6	For each driver's card change TCN sends out a separate xml message to other MS (DC issuing state)	MS validates xml message

step7	For each driver's card change TCN receives response xml messages from the other MS	TCN validates xml message
step8	TCN stores response(s)	all ms responses are grouped into one message
step9	TCN forwards response xml message to CiA App	CiA App validates xml message
step10	CiA App stores response.	

- S1034-03** CiA App performs card status modification received from TCN - local DL
- S1034-04** CiA App performs card status modification received from TCN - non-local DL

Step Number	Action to perform	Expected result
step1	CiA App receives declaration of card status modification	Driver name and driver card n° known
step2	CiA App retrieves driver card info based on provided Phonex search keys	
step3	CiA App modifies card status	
step4	CiA App sends back acknowledgement	
step5	TCN receives status change acknowledgement message from MS	TCN validates xml message
step6	TCN stores response	
step7	TCN forwards response xml message to other CiA App	CiA App validates xml message

- S1035-03** CiA App performs card status modification by batch received from TCN - local DL
- S1035-04** CiA App performs card status modification by batch received from TCN - non-local DL

Step Number	Action to perform	Expected result
step1	CiA App receives batch of declarations of card status modification	Driver name and driver card n° known
step2	For each driver card CiA App retrieves driver card info based on provided Phonex search keys	
step3	CiA App modifies card status	
step4	For each driver card CiA App sends back acknowledgement	
step5	CiA App groups all responses into xml message	
step6	TCN receives acknowledgement message from MS	TCN validates xml message

step7 TCN stores response

step8 TCN forwards response xml message to other CiA App

CiA App validates xml message

---



## Send Card/Driving license assignment

**Use Case Description**

This use case is in fact triggered when a driver card is being created for a driver that has a driving license in another member state. The CiA Application responsible for the creation of the driver’s card sends out an update through TCN to the driving license issuing member state.

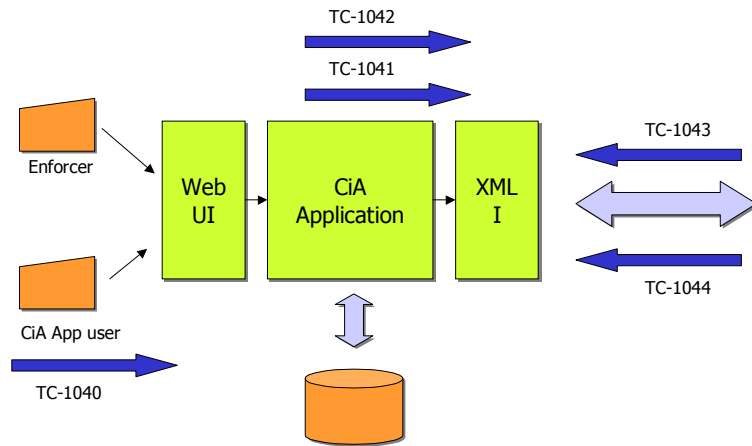
So in order to inform the other member states that a driver card is created, the CiA Application has to be able to actually create the card (UC-0100).

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1040	CiA App user declaring driver’s card creation – single instance
TC-1041	CiA Application declaring driver’s card creation – single instance
TC-1042	CiA application declaring driver’s card creation – batch treatment
TC-1043	CiA application handles driver’s card creation declaration – single instance
TC-1044	CiA application handles driver’s card creation declaration – incoming batch

**Test Cases Visualization**



**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S1040-xx	CiA App user declares creation of DriverCard towards TCN
S1040-01	non-local DL – send out to 1 MS
S1040-11	Driver not found in foreign database
S1040-12	Phonex search keys cover multiple Drivers
S1040-14	Driver License can not be found or is not supported
S1040-15	Request timed out
S1040-16	ServerError due to loss of connection

Test Id	Description
S1041-xx	CiA App declares creation of DriverCard towards TCN
S1041-01	non-local DL – send out to 1 MS

Test Id	Description
S1042-01	CiA App declares by batch creation of DriverCards towards TCN
S1042-02	Response will combine multiple S1043-01/02/03/04 cases
S1042-03	Response will combine S1043-03/04 with S1043-05
S1042-04	Response will combine S1043-03/04 with S1043-06

Test Id	Description
S1043-xx	CiA App receives declaration of DriverCard creation from TCN
S1043-01	local DL/non-local DC
S1043-02	Driver not found in local database
S1043-03	Phonex search keys cover multiple Drivers
S1043-04	Driver License can not be found or is not supported
S1043-05	Request timed out
S1043-06	ServerError due to loss of connection

Test Id	Description
S1044-01	CiA App receives batch of declarations of Driver Card creations – local DL/non-local DC
S1044-03	Combining multiple S1043-01/02/03/04 cases
S1044-04	Combining S1043-03/04 with S1043-05
S1044-05	Combining S1043-03/04 with S1043-06

**User Input variants**

Because the declaration of a driver’s card creation is a logical extension of the actually creation of the card, no web user interaction is required hence no Web User test scenarios.

No particular test scenario for transliteration because the required Phonex search key generation and transliteration is already done at creation of the driver’s card and is not required for declaration (driver license number is!).

**Remark on Test Ids**

Test scenarios -11 through -16 are not repeated for CiA App because they do not test a new aspect or functionality that an actor can introduce through an interface.

Test scenarios –11 through –16 are dealing with the kind of response expected from the central system (or other MS) or cover non-standard behavior of the solution.

**Sending and receiving end**

As indicated by the diagram on page 74, there is a link between test cases TC1040/41 and TC1043 as there is a link between TC1042 and TC1044. So keep in mind when receiving an S1043-xx the other side is actually sending an S1040-xx.







## Send Card/Driving license assignment, continued

### S1040-01 CiA App user declares creation of DriverCard towards TCN - non-local DL/local DC

Step Number	Action to perform	Expected result
step1	CiA App user creates DriverCard for Driver with foreign driver's license	
step2a	S0461-01	Latin – US transliteration is done
step2b	S0462-01	Greek – US transliteration is done
step3	S0451-xx	Phonex search keys are generated
step4	CiA App sends out xml message to TCN declaring driver card creation	
step5	TCN stores request	
step6	TCN sends out xml message to other MS (DL issuing state)	MS validates xml message
step7	TCN receives response xml message from MS	TCN validates xml message
step8	TCN stores response	
step9	TCN forwards response xml message to CiA App	CiA App validates xml message

### S1041-01 CiA App declares creation of DriverCard towards TCN - non-local DL/local DC

Step Number	Action to perform	Expected result
step1	CiA App receives confirmation of local driver card creation	
step2	CiA App sends out xml message to TCN declaring driver card creation	
step3	TCN stores request	
step4	TCN sends out xml message to other MS (DL issuing state)	MS validates xml message
step5	TCN receives response xml message from MS	TCN validates xml message
step6	TCN stores response	
step7	TCN forwards response xml message to CiA App	CiA App validates xml message

**S1042-01** CiA App declares by batch creation of DriverCards towards TCN

Step Number	Action to perform	Expected result	
step1	CiA App creates batch of driver card creation declarations	For each Driver card n° and driver name are provided	
step2	CiA App sends out xml message to TCN		
step3	TCN stores request		
step4	TCN sends out xml message to other MS (DL issuing state)		MS validates xml message
step5	TCN receives response xml message from each MS		TCN validates xml message
step6	TCN stores response		
step7	TCN groups all reponses into one xml message		CiA App validates xml message
step8	TCN forwards response xml message to CiA App		

**S1043-01** CiA App receives declaration of DriverCard creation from TCN - local DL/non-local DC

Step Number	Action to perform	Expected result
step1	CiA App receives declaration of driver card creation	MS validates xml message
step2	CiA App searches for driver info locally through provided Phonex search keys Driver info locally updated	
step3	CiA App sends out acknowledgment message to TCN	
step4	TCN stores request	
step5	TCN sends out acknowledgment message to other MS (DC issuing state)	

**S1044-01** CiA application handles driver's card creation declaration – incoming batch

Step Number	Action to perform	Expected result
step1	CiA App receives batch of driver card creation declarations	Driver info locally updated
step2	For each driver card CiA App searches for driver info locally through provided Phonex search keys	
step3	For each driver card CiA App creates acknowledgment	
step4	CiA App sends out grouped acknowledgment message to TCN	
step5	TCN stores request	

## Create Card – First Issue

**Business Use Case Description**

This use case describes the sequence of activities that has to be performed in order to create a driver’s card. This includes a check on the driver and the fact that he doesn’t yet own a driver’s card (in another MS). (TC-1010)

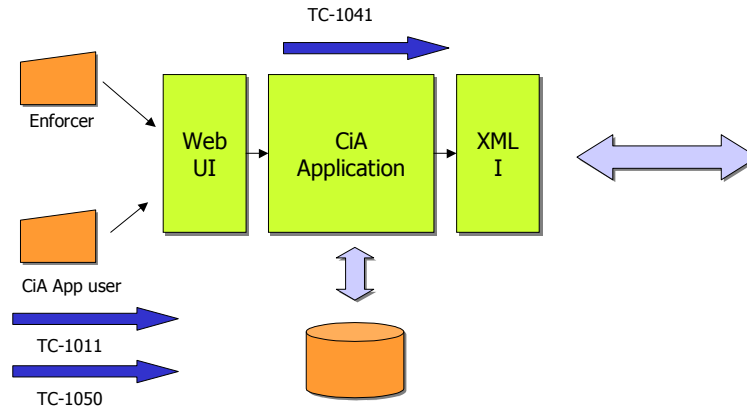
This use case can then be extended by sending out a message towards TCN and other MS that a driver’s card has been created for a driver. (TC-1040)

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1050	CiA App user creating Driver’s card – first issue – single instance

**Test Cases Visualization**



**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S1050-01	CiA App user creating DriverCard - first issue - local DL/local DC – out of scope
S1050-02	CiA App user creating DriverCard - first issue - non-local DL/local DC
S1050-03	CiA App user creating DriverCard with transliteration - first issue - non-local DL/local DC

---

**User Input  
variants**

Besides the use cases being mentioned in the use case description, this use case re-uses also the Get Phonex Search keys and US/Ascii transliteration. The first is considered to be initiated always while the latter could be initiated, creating as such different variants for Latin-based and Greek input.

The CiA application mentioned here is in fact a CiA application user working through a user interface. We are not dealing with Enforcers.

---

## Create Card – First Issue, continued

- S1050-02** CiA App user creating DriverCard - first issue - non-local DL/local DC
- S1050-03** CiA App user creating DriverCard with transliteration - first issue - non-local DL/local DC

Step Number	Action to perform	Expected result
step1	local CiA App interface is presented	
step2	S1010-xx	CiA App (user) verifies if a Driver Card is already issued
step3a	User enters required info for Driver Card creation	No Driver Card has previously been issued
step3b	User terminates creation (end of scenario)	Driver Card has previously been issued
step4a	S0461-01	Latin – US transliteration is done
step4b	S0462-01	Greek – US transliteration is done
step5	S0451-xx	Phonex search keys are generated
step6	user selects submit	CiA App verifies mandatory fields – Driver Card created locally
step7a	No declaration is done	
step7b	CiA App user send out declaration of Driver Card creation towards TCN	S1040-xx
step7c	CiA App user decides to send out declaration later (single or in batch)	S1040-xx



## Card Renewal

---

### **Business Use Case Description**

This use case covers the activities to be performed when a driver's card has to be renewed due to card expiration. Since card validity is described by a particular card status value we can say that this business use case is a variant on the Create Card (TC-1050) use case. This activity is considered a local activity and as such out of scope for this test plan document. If card renewal is defined as a card status modification a member state can opt to broadcast the status change (TC-1030).

---

### **Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1051	CiA App creating driver's card – renewal – single instance

---

### **Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description

---



# Card Exchange

**Business Use Case Description**

This use case covers the activities to be performed when a driver request that his current driver’s card be exchanged into another driver’s card. This can happen when a driver changes from member state.

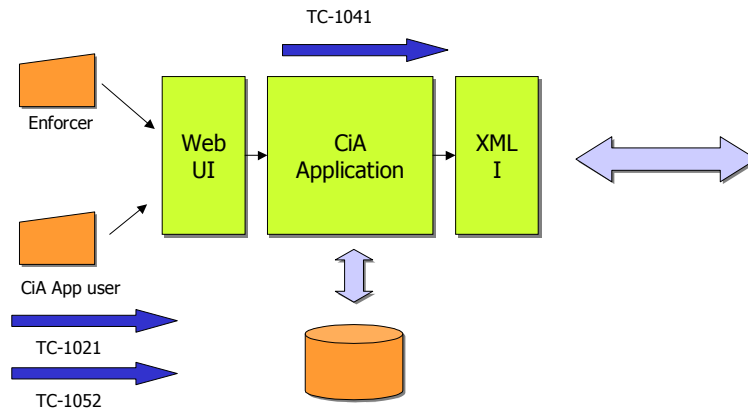
This use case reuses the Create Card (TC-1050) use case and the Check Tachograph card status (TC-1020).

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1052	CiA App user creating driver’s card – exchange – single instance

**Test Cases Visualization**



**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S1052-01	CiA App user creating DriverCard - exchange - local DL/>>local DC
S1052-02	CiA App user creating DriverCard - exchange - non-local DL/>>local DC

**User Input  
variants**

Besides the use cases being mentioned in the use case description, this use case re-uses also the Get Phonex Search keys and US/Ascii transliteration. The first is considered to be initiated always while the latter could be initiated, creating as such different variants for Latin-based and Greek input.

The CiA application mentioned here is in fact a CiA application user working through a user interface. We are not dealing with Enforcers.

The declaration of card creations and status changes can be grouped into one batch.

---

## Card Exchange, continued

- S1052-01** CiA App user creating DriverCard - exchange - local DL/>>local DC
- S1052-02** CiA App user creating DriverCard - exchange - non-local DL/>>local DC

Step Number	Action to perform	Expected result
step1	local CiA App interface is presented	
step2	S1021-xx	CiA App (user) verifies the Driver Card status
step3a	User enters required info for Driver Card creation	Driver Card status available and acceptable for exchange
step3b	User terminates creation (end of scenario)	Driver Card status unavailable or not acceptable for exchange
step4a	S0461-01	Latin – US transliteration is done
step4b	S0462-01	Greek – US transliteration is done
step5	S0451-xx	Phonex search keys are generated
step6	user selects submit	CiA App verifies mandatory fields – Driver Card created locally
step7a	No declaration is done	
step7b	CiA App user send out declaration of Driver Card creation towards TCN	S1040-xx
step7c	CiA App user decides to send out declaration later (single or in batch)	S1041-xx or S1042-xx

## Card Replacement

---

**Business Use Case Description**

This use case covers the activities to be performed when an existing card went lost and had to be replaced by a new (similar) one. Card replacement is described as a variant on the Create Card (TC-1050) use case and only when a certain card status is reached. More over this activity is considered a local activity and as such falls out of the scope of this test plan document. If card replacement is defined as a card status modification a member state can opt to broadcast the status change (TC-1030).

---

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1053	CiA App user creating driver's card - replacement – single instance

---

**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description

---

## Browse Statistics

**Use Case Description**

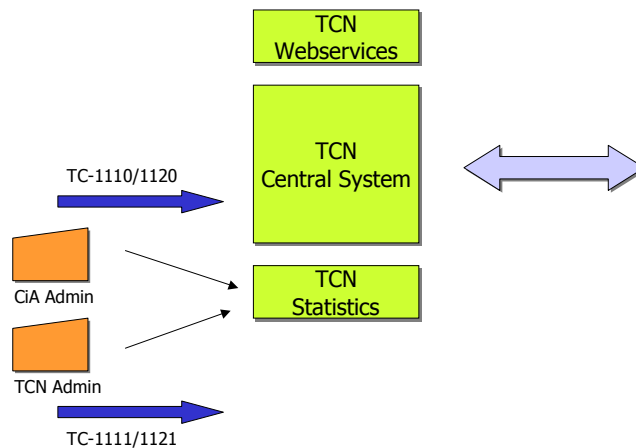
This use case allows a CiA Administrator or TACHOnet Administrator to view through a dedicated Tachonet web interface the gathered statistics for his and other member states.

**Test Cases**

Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1110	CiA Admin browse list
TC-1111	TCN Admin browse list
TC-1120	CiA Admin browse consolidation
TC-1121	TCN Admin browse consolidation

**Test Cases Visualization**



**Test Scenarios**

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
S1110-01	CiA Admin views list - incoming from - normal selection
S1110-02	CiA Admin views list - outgoing to - normal selection
S1110-03	CiA Admin selects list - filtering on date/time
S1110-04	CiA Admin selects list - filtering on mode
S1110-05	CiA Admin selects list - filtering on message type

S1110-06	CiA Admin selects list - filtering on status
S1110-07	CiA Admin selects list - export list

Test Id	Description
S1111-01	TCN Admin views list - incoming from - normal selection

Test Id	Description
S1120-01	CiA Admin views consolidation with chart - incoming from - normal selection
S1120-02	CiA Admin views consolidation with chart - outgoing to - normal selection
S1120-04	CiA Admin browse consolidation over a period
S1120-05	CiA Admin browse consolidation by mode / type / MS / status

Test Id	Description
S1121-01	TCN Admin views consolidation with chart - incoming from - normal selection

**Remark**

Complete test scenarios to be provided.

## Manage Users

---

**Use Case Description** The TCN Admin has the access rights to add a new CiA to the network and manages the access rights of those CiA's through their CiA Admins. A separate interface is available to perform a limited number of actions.

---

**Test Cases** Each use case being a generic activity can be realized, executed in a number of ways. These are termed use-case realizations or test cases. We distinguish the following test cases for this use case:

Test Case Id	Description
TC-1200	Add CiA
TC-1201	Reset Password of Admin

---

**Test Scenarios** We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Case Id	Description
S1200-01	TCN Admin adds CiA
S1201-01	TCN Admin resets Password of CiA Admin
S1201-02	TCN Admin resets Password of TCN Admin

## Non-Functional Testing

---

### Necessity

A separate document exists that describes the amount of possible messages that could be send out to the TCN central system in terms of message volume and frequency on a country and working-hour basis. During the pilot-testing phase we should get a clear view of how close our predictions are.

To anticipate possible problems we have introduced the notion of bulk checks and batch sending during the non-working hours of the local CiA clerks (or at night).

---

### Load Testing

The purpose of these tests is to start from a normal workload (a typical xml message) and to increase the size of the message being send. Because bandwidth remains the same and workload increases the speed of handling the messages will fluctuate.

Due to the fact that messages being exchanged have a fixed structure we can estimate the size of each message type and can start comparing if the change in size influence the handling.

In order to have any measurable changes the amount of messages being send could be augmented.

---

### Test Cases

The amount of test cases presented here is rather limited and generic.

Test Case Id	Description
TC-2011	CiA First Issue – online request – size 1/10/100
TC-2012	EA Check Driver’s Issued Card – online request – size 1/10/100
TC-2013	EA Check Card Status – online request – size 1/10/100
TC-2014	CiA First Issue – batch request – size 1/10/100

---

### Test Scenarios

We have further extended the available test cases into separate test scenarios. The major difference between both is in terms of possible actors (executing the activity) and slightly different inputs. We obtain the following test scenarios:

Test Id	Description
	CiA First Issue – online request – size 1/10/100 - incoming
	CiA First Issue – online request – size 1/10/100 – outgoing
	EA Check Driver’s Issued Card – online request – size 1/10/100 - incoming
	EA Check Driver’s Issued Card – online request - size 1/10/100 – outgoing
	EA Check Card Status – online request – size 1/10/100 – incoming
	EA Check Card Status – online request – size 1/10/100 – outgoing
	CiA First Issue – batch request – size 1/10/100 – incoming
	CiA First Issue – batch request – size 1/10/100 – outgoing

---

### Stress Testing

The handling of xml messages requires resources in terms of bandwidth, processor and memory. This will be tested by using a normal workload but over an extended time period allowing us to detect memory leaks, lack of space or other hardware failures or limitations.



---

**Test Cases**

The amount of test cases presented here is rather limited and generic.

Test Case Id	Description
TC-2021	CiA First Issue – online request – 7 hours non stop
TC-2022	CiA First Issue – batch request – 3 hours non stop
TC-2023	CiA First Issue – online request – 24 hours non stop

---

**Volume Testing**

The amount of member states and in particular the amount of local users (Enforcers and CiA App users) will influence the network behaviour. With these tests we will simulate concurrent users/sessions.

---

**Test Cases**

The amount of test cases presented here is rather limited and generic.

Test Case Id	Description
TC-2031	CiA First Issue – online request – 1 in/1 out/sec
TC-2032	CiA First Issue – online request – 5 in/5 out/sec
TC-2033	CiA First Issue – online request – 10 in/10 out/sec

---

**Performance Profiling**

This last set of non-functional tests are actually the most important because they will allow us to simulate a full working Tachonet network. Based on feedback received from the different member states we have calculated the amount of messages being exchanged in function of message type, and request type and peak period.

---

**Test Cases**

The amount of test cases presented here is rather limited and generic.

Test Case Id	Description
TC-2041	CiA First Issue – online request – 1 in/1 out/sec – 19 CiAs
TC-2042	CiA First Issue – online request – 5 in/5 out/sec – 19 CiAs
TC-2043	CiA First Issue – online request – 10 in/10 out/sec – 19 CiAs
TC-2044	CiA First Issue – batch request – 1 in/1 out/sec – 19 CiAs
TC-2045	CiA First Issue – batch request – 5 in/5 out/sec – 19 CiAs
TC-2046	CiA First Issue – batch request – 7 in/7 out/sec – 19 CiAs
TC-2047	CiA First Issue – online request – 10 in/10 out/sec – 29 CiAs
TC-2048	CiA First Issue – batch request – 7 in/7 out/sec – 29 CiAs

---

<End of the document/>