

Moving Europe towards a sustainable and  
safe railway system without frontiers.

## Common interface

*Telematics TSI - Technical document - TD104*

*Version 4.0*

## Contents

A.	Document management .....	4
A.1	Document properties .....	4
A.2	Change management .....	4
A.3	Configuration management .....	4
A.4	Availability .....	4
A.5	Application and actors in the scope .....	4
A.6	Document history .....	5
B.	Acronyms, definitions and external references .....	5
B.1	Acronyms .....	5
B.2	Definitions .....	6
B.3	External references .....	6
1	Implementation approach (ref 4.2.12) .....	8
1.1	Architecture of the Common Interface .....	8
1.2	Meeting the TSI Objective .....	10
1.3	Description of the technical and operational environment.....	14
1.4	References .....	14
1.5	Definitions and Acronyms.....	14
2	Functional requirements.....	15
2.1	Logical Model – Generic API (ref 4.2.14.1 & 7) .....	15
2.2	Logical Model – Translation & Validation Layer (ref 4.2.12.1, 5 & 6).....	17
2.3	Logical Model – Interface between Translation & Validation Layer and Security and Transport Layer .....	18
2.4	Logical Model – Security and Transport Layer.....	19
2.5	Reference Files and Databases (ref 4.2.11.1).....	21
2.6	Metadata .....	21
2.7	Human-Computer-Interface (ref 4.2.12.1).....	23
3	Data quality.....	23
3.1	Data Quality (ref 4.4.1) .....	23
4	Communication layer between Common Interface .....	28
5	Web service .....	29

5.1	WSDL.....	29
5.2	Request.....	29
5.3	Response .....	31
5.4	Invocation.....	32
5.5	Message Compression.....	33
5.6	Message Encryption .....	33
5.7	Message Signing.....	33
6	REST .....	34
6.1	Request.....	34
6.2	Response .....	35
7	Web service for Remote LI heartbeat check.....	36
7.1	Request.....	36
7.2	Response .....	37
7.3	Invocation .....	37
ANNEX 1	MESSAGE EXCHANGE WSDL.....	38
ANNEX 2	SAMPLE MESSAGE-WITH-COMPRESSSION-SIGNING-ENCRYPTION .....	40
ANNEX 3	SAMPLE MESSAGE-WITHOUT-COMPRESSSION-SIGNING-ENCRYPTION .....	41
	LocationSubsidiaryTypeCode="" />.....	41
Annex 4	Schema for the acknowledgement XML.....	44
ANNEX 5	SAMPLE ACKNOWLEDGEMENT RESPONSE MESSAGE .....	47
ANNEX 6	Heartbeat Request WSDL .....	48
ANNEX 7	Sample heartbeat request message .....	49
ANNEX 8	ACKNOWLEDGEMENT XML FOR HEARTBEAT REQUEST.....	50

## A. Document management

### A.1 Document properties

- File name: ERA\_TD\_104.docx
- Subject and document type: Telematics TSI - Technical document - TD104
- Author: European Union Agency for Railways
- Version: 4.0

### A.2 Change management

Updates to this technical document shall be subject to Change Control Management procedure managed by the Agency pursuant:

- the applicable requirements in the reference TSI
- Art. 23(2) of the Agency Regulation

If necessary, working groups are created in line with Art. 5 of the Agency Regulation.

### A.3 Configuration management

A new version of the document will be created if new changes are considered following the Change Control Management Process led by ERA.

More specifically:

- if there is a change in the requirements which influences the implementation
- if information is added to or deleted from the technical document
- adding test cases to the field checking in messages or databases.

Modifications will have to be highlighted, so they can be easily identified.

Disclaimer:

Specific legal references to technical documents and legal acts shall be revised after the enter into force of the Telematics TSI. In some sections this text can be highlighted.

### A.4 Availability

The version in force of this document is available on Agency's Gitlab repository. Any printed copy is uncontrolled.

### A.5 Application and actors in the scope

Date of entry into force of reference TSI.

This document applies to all the actors in the scope of the reference TSI.

## A.6 Document history

Table 1 - Document history

<i>Version</i>	<i>Date</i>	<i>Modifications</i>
1.0	25.01.2011	Initial version
2.0	08.08.2013	All the chapters were revised due to the TAF TSI Revision Process and the TAF TSI CCM WP cycle 2012 – 2013.
2.0	17.10.2013	Validated by the ERA TAF CCB on 11.09.2013
2.1	05.07.2016	Specification of the external interface of the CI incorporated
2.2	23.2.2017	Validated by the ERA TAF CCB on 23.02.2017
2.5	27.05.2020	Validated by the ERA TAF CCB on 27.05.2020
3.0	15.06.2021	Version number changed to 3.0
4.0	10.06.2025	Inclusion of new elements from Telematics TSI revision and from new CI.

## B. Acronyms, definitions and external references

### B.1 Acronyms

Table 2 - Acronyms

<i>Acronym / Abbreviation</i>	<i>Definitions</i>
CI	Common Interface also LI Local Instance
CSV	Comma separated values
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
GUI	Graphical User Interface (subset of HMI)
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IP	Internet Protocol
ISO	International Standardization Organization
JMS	Java Message Service

LI	Local Interface, also Common Interface
LS	Legacy System
MQ	Message Queuing
NACK	Negative Acknowledge
PKI	Public Key Infrastructure
PBE	Password Based Encryption
RA	Registration Authority
RNE	RailNetEurope
SHA256	Shared Cybersecurity Specification SHA256withECDSA
SOAP	Simple Object Access Protocol
TXT	Text format
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
UTF-8	Unicode Transformation Format-8 (8-bit multi-character encoding)
VPN	Virtual Private Network
WAR	Web Archive format
WSDL	Web Service Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition Language

## B.2 Definitions

Terms contained in this document are defined in the ERA Ontology.

## B.3 External references

The referenced documents listed in Table 2 are indispensable for the application of this document:

- For dated references, only the edition cited applies;
- For undated references, if any, the latest edition of the referenced document (including any amendments) applies.

### Table 2 Reference documents

1. Telematics TSI : “**Draft annex** to the Commission Implementing Regulation on a technical specification relating to the telematics subsystem of the rail system in the European Union for interoperability of data sharing in rail transport and repealing Regulations (EU) No 454/2011 and (EU) No 1305/2014”



## 1 Implementation approach (ref 4.2.12)

In relation to the Common Interface, the Telematics Application Sub System (Telematics Applications (TA) TSI) documents the essential requirements for Telematics Applications (referring to 26 a) and b) of Annex II to DIRECTIVE (EU) 2016/797):

“2.7. Telematics applications for freight and passengers

### 2.7.1. Technical compatibility

The essential requirements for telematics applications guarantee a minimum quality of service for passengers and carriers of goods, particularly in terms of technical compatibility.

Steps must be taken to ensure:

- that the databases, software and data communication protocols are developed in a manner allowing maximum data interchange between different applications and operators, excluding confidential commercial data,
- easy access to the information for users.

### 2.7.2. Reliability and availability

The methods of use, management, updating and maintenance of these databases, software and data communication protocols must guarantee the efficiency of these systems and the quality of the service.”

Consequently, chapter 4.2.12.6 of the Telematics TSI document that the Common Interface is mandatory for each actor in order to join the TA TSI rail interoperability community and must have the following capabilities :

- message formatting of outgoing messages according to the metadata,
- signing and encryption of outgoing messages,
- addressing of the outgoing messages,
- authenticity verification of the incoming messages,
- decryption of incoming messages,
- conformity checks of incoming messages according to metadata,
- handling the single common access to various databases.

## 1.1 Architecture of the Common Interface

### 1.1.1 Architecture required by the TSI (ref 1.7)

In Chapter 1.7 of TA (Telematics Applications) TSI “Networking & Communication” it is prescribed the requirements to be fulfilled by the Common Interface, the application programming interfaces (‘API’) and web user interfaces (‘UI’) and the Message exchange.

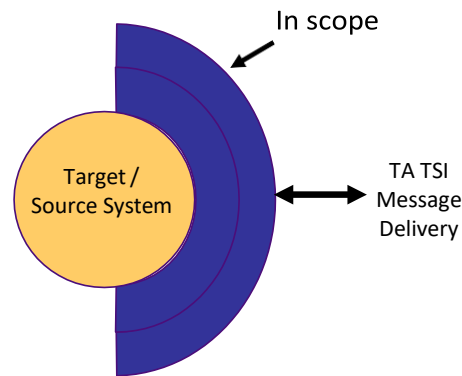


For version control, all trading partners shall continue to support current and previous versions until there is agreement to withdraw the previous version.

### 1.1.2 Common interface Implementation Architecture

The context of the Common Interface is that it will provide the functionality between the TA TSI messages and all target systems and human input exchanging data by TA TSI messages. Taking the requirements of the TSI, the Common Interface implementation architecture will therefore be structured as follows :

#### Scope of Common Interface



The purpose of structuring the Common Interface components in the implementation architecture above is to ensure that the Common Interface can meet the technical requirements prescribed in section 1.4. By achieving these, the Common Interface will deliver early TA TSI benefits by utilising existing systems, IT platforms and communication processes whilst allowing the scalability required to implement the longer-term TA TSI vision.

The Common Interface architecture allows the translation of internal data to and from TAF TSI messages and the management and transmission of the messages according to the requirements of the TSI between any actor, independent of the internal systems or business processes in use by that actor.

Messages are sent and received through an open message queue, which is the interface between the Translation & Validation layer of the Common Interface and the Security & Transport layer of the Common Interface. The Security and Transport layer manages the delivery and receipt of messages to and from the public network side of the Message Queue. The Translation and Validation layer and API layers manage the receipt of data from and delivery of data to the systems in use on the internal side of the Message Queue.

The Common metadata is used by the Common Interface for all activities that are standardised by the TSI and the Private / Shared metadata is used by the Common Interface for local activities related primarily to the API interfaces with internal systems and local operation of the Common Interface itself. The implementation of the Common Interface must be such that any modification of public /shared and private metadata can be made dynamically and mustn't have any effect on operation. Metadata changes should be applied according multilateral or bilateral agreements.

As an overall goal, the CI will support the provisions of Art 1.7 of the Telematics Applications TSI.

## 1.2 Meeting the TSI Objective

The objective of this Functional Requirements Document is to specify the Common Interface in sufficient detail for implementation, taking as its starting point, the high-level specifications described in the TAF-TSI concerning the messaging and data model.

The requirements from this document are not applicable for message exchanges for rail passenger distribution processes.

The requirements from this document are also applicable for programming interfaces ('API') and web user interfaces ('UI') referred to in Article 14 of the TSI and chapter 1.7 of its annex.

As regards cybersecurity, the provisions of chapter 1.3 of the TSI are applicable.

This document describes how, using the principles of **Semantic Integration**, the service providers as quoted on Article 2 "Technical Scope" of TA TSI core text, to exchange data and implement the Telematics Application for Freight Regulation through an Information Exchange Architecture.

In this context, **Semantics** the study of meaning, is used for understanding, implementing and managing the complex TA TSI message exchanges. Semantics = Data + Behaviour. This includes data quality assurance through the implementation of data quality checking in the Common Interface.

**Integration** is used in this context to describe creating networks of interrelated IT applications to provide benefit to the Rail Industry. The TA TSI Information Exchange Architecture for messaging requires the co-ordination of data, messages and responses from applications across Europe via multiple implementations of its Common Interface.

The TA TSI Semantic Integration framework is designed to focus on delivering high-returns quickly, by utilising existing applications as data sources.

An important secondary purpose of this Functional Requirement Specification is to utilise the Semantic Integration framework to deliver significant cost and timescale reductions implementation of TA TSI.

To deliver value, the TA TSI Semantic Integration framework for network applications is therefore required to:

- Be compatible with existing application systems;
- Use selected existing integration technologies;
- Support emerging integration and system technologies;
- Be capable of integration across organisational boundaries.

By its very nature integration is about the interactions between applications

- therefore this Common Interface Functional Requirements Specification focuses on the semantics of the message and collaboration between systems.

A direct benefit of focusing on the interaction between systems is the ability to scale because the message is the fundamental unit of integration. The TA TSI messages are collections of data, organised in a specific way, and grouped to provide context. To effectively use existing messages in the TA TSI integration framework, existing sources have been identified to avoid the expense of creating new ones wherever possible. Only those elements of existing messages that describe a reliable, consistent set of semantic properties have been incorporated. Message translation within the Common Interface will be used to reconcile differences between applications and TA TSI messages.

#### **1.2.1.1 API Adaptors (ref 1.7)**

Application Programming Interface Adaptors for linking the Common Interface to the applications/components in use by actors are required as part of the implementation process. These will provide the host application with a well-defined interface and manage the technical communications between the application and the Common Interface.

#### **1.2.1.2 Translation and Validation (ref 1.5)**

A standardized approach to the creation and reading of messages and the validation of those messages is a core component to having a workable technical implementation across many peer-to-peer actors. Common quality criteria can then be applied to the messages and data.

#### **1.2.1.3 Metadata System (ref 1.5)**

Metadata is data whose purpose is to describe other data: its definitions, structure and relationships. For the purposes of the TA TSI regulation, the Information Exchange Architecture needs different types metadata for three separate purposes:

- Concrete, prescriptive metadata for the interface metadata syntax and representation - consistent and compatible for each application.
- Descriptive metadata that concretely describes the content of the actual exchanged messages so that translations can be defined within the Common Interface.
- Abstract metadata to understanding what the data means to each application and for describing the relationships between data elements.

The Metadata required for operation of the Common Interface will be applied in two parts, Private Metadata (primarily defining the local conditions of the particular implementation of the Common Interface and the APIs which will interface with the internal systems,) Shared Metadata (primarily defining all the messages and their versions plus delivery addressing for direct actor to actor specific data exchange) and Common Metadata (primarily defining all the messages and their versions plus delivery addressing).

The TA TSI metadata defines the syntax of how the data is represented, how it is structured, the order of the elements, constraints, required quality conditions and any business rules. The actual metadata can be downloaded from the Agency's website: [https://github.com/EU-Agency-for-Railways/TSI\\_TAF/tree/master](https://github.com/EU-Agency-for-Railways/TSI_TAF/tree/master).

#### **1.2.1.4 Security and Transport (ref 1.3)**

A standardised approach to the security and transportation of messages will deliver end-to-end loss-less delivery across whatever networks are available, notably the public internet.

#### **1.2.1.5 XML (ref 1.4)**

XML is used to describe the metadata and messages because XML is the current, open, standardised syntax in widespread use used to parse a document into named elements. XML also provides a standardised format for structure metadata and constraints metadata, called an XML Schema (.xsd).

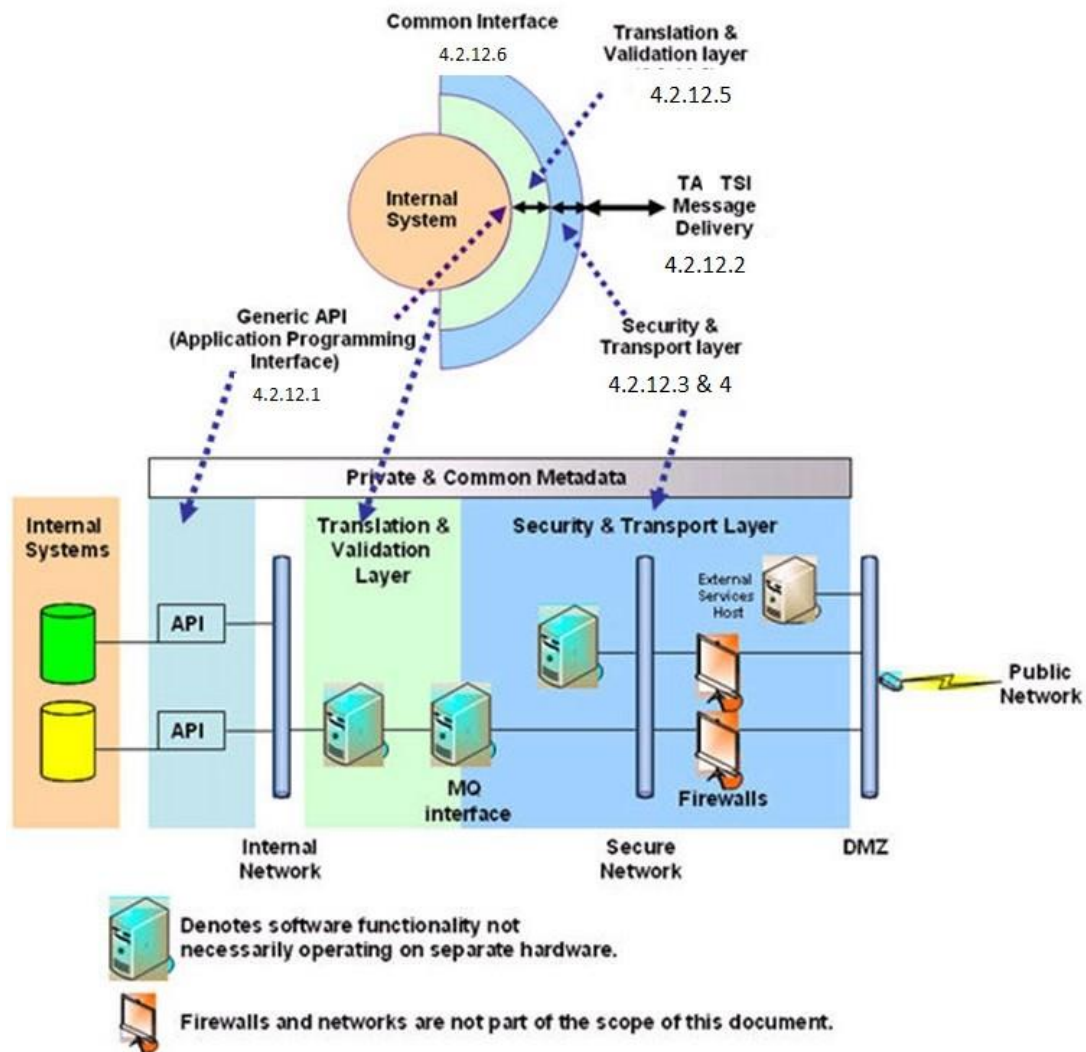
XML Schemas (.xsd) describe data precisely enough information for computers to parse any message into labeled component elements. TA TSI XML Schemas are described in the technical document ERA-TD-105: TAF TSI - Annex D.2 : Appendix F - TAF TSI Data and Message Model, Version as stored on [https://github.com/EU-Agency-for-Railways/TSI\\_TAF/tree/master](https://github.com/EU-Agency-for-Railways/TSI_TAF/tree/master). The metadata expressed in the TA TSI XML Schemas describe what each message looks like, how it is structured, which parts are optional, which are required, and value constraints. However, for semantic integration to be successful in delivering value using existing applications, the TA TSI XML Schemas also describe data found in messages from selected existing systems, CEN agreements, UIC leaflets and ERA Technical Documents. Most rail messages exchanged today are not in XML, they are in formats such as Electronic Data Interchange (EDI), flat text files or proprietary formats. The TA TSI XML Schemas are rich enough to describe any of non-XML data resources. The TA TSI regulation has focused on reconciling the differences between the messages from existing applications by describing how data elements are related in maps so that it will be possible to deploy real-world integration systems that will translate the messages as they flow between existing non-compatible applications.

#### **1.2.1.6 RDF and ERA Ontology (ref 1.4)**

On top of XSD schemas, the Common Interface will be - in its future releases - able to ensure serialization in RDF and conversion between XSD and RDF based on ERA Ontology. The ERA Ontology is described in Appendix C, index [2] and its elements (including metadata with validation rules or code list concepts) are published on the public ERA Ontology website of the European Union Agency for Railways. The RDF based future message exchange will be used between the Local Instances of the Common Interface.

For the sound implementation of RDF based future message exchange a transition period should be agreed for the introduction of RDF

### 1.2.2 Detailed Structure



The diagram above shows the detailed structure of the Common Interface with the constituent parts. The network(s) and Firewall(s) are shown for completeness but are not part of this Functional Requirement Specification. Additionally, the Common Interface must be operable on a PC for low volume installations and be scalable to separate hardware, if required, for higher volume installations.

### **1.3 Description of the technical and operational environment**

#### **1.3.1 *IT Platforms***

The Reference Implementation of the Common Interface must be capable of reliable operation on open compliant IT platforms.

### **1.4 References**

See section B.3

### **1.5 Definitions and Acronyms**

See section B.1

## 2 Functional requirements

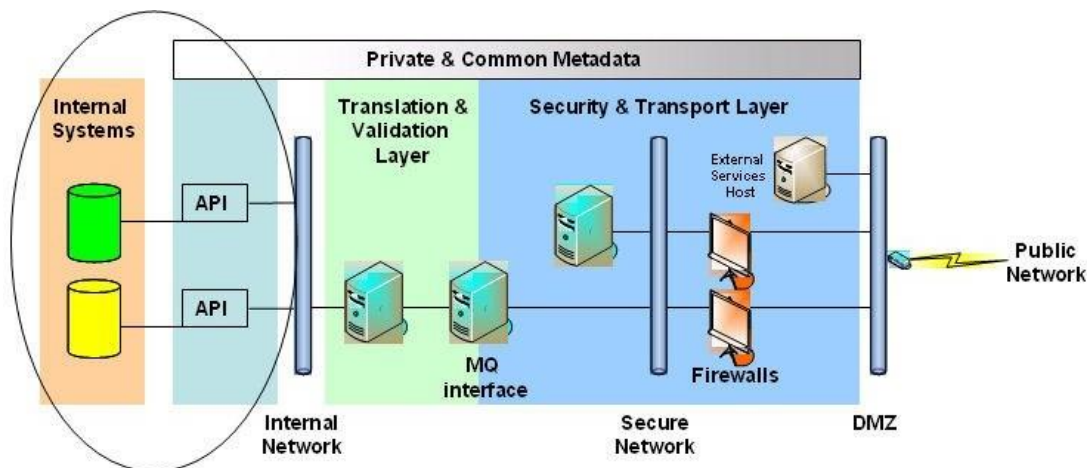
### 2.1 Logical Model – Generic API (ref 4.2.14.1 & 7)

**Requirement 2.1.1 :** To make it possible to connect the Common Interface to existing and future systems.

**Requirement 2.1.2 :** To have a Platform independent API

**Requirement 2.1.3 :** To provide support for rail industry-standard programming languages

**Requirement 2.1.4:** To report the success or raise the exception with reasons in case of failure conditions.



Generic APIs of the Common Interface will allow individual actors in the TA TSI process, mainly RUs, Combined Transport Operators, WKs and IMs, to connect internal existing or new systems to the Common Interface.

The APIs shall not address the functionality of the Transport Layer with calls such as Encryption, network protocols, etc.

The functionality that must be supported in the generic APIs is as follows :

The APIs must provide all necessary open-standard functionality for sending and receiving messages to internal RU/IM systems.

In order for the API to be successfully integrated into the target application, the following must be included:

- Description of how messages are sent and received
- Description of services provided by the interface
- Services for the following functions of the application interface covering
  - Opening and closing the session
  - Sending and receiving messages
  - Request and Response
  - Message Destination

- Name and Value Elements
  - Error Handling
- Object and Class References
  - Base Classes, Help Classes and Exception Classes
- Installation
  - Defining Services, Policies and Policy Handlers
- Authenticating infrequent access
  - Security
  - Problem Determination
  - Monitoring

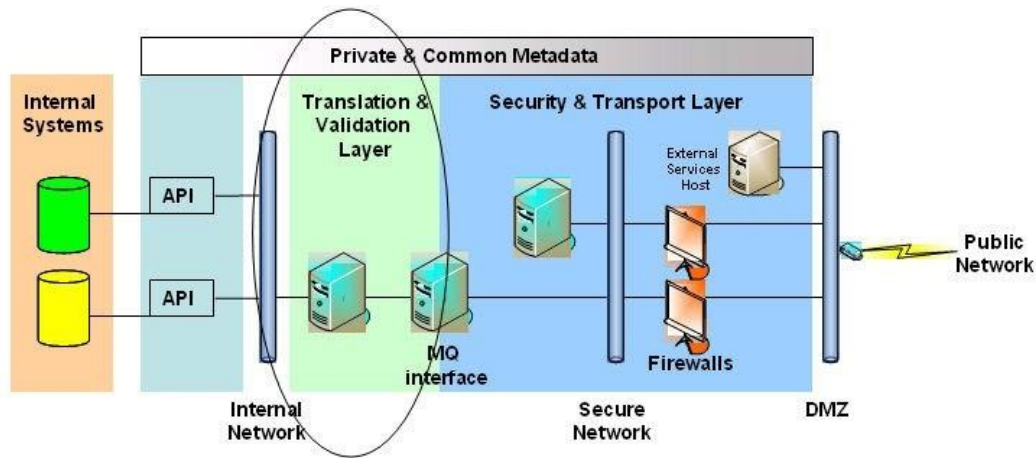


## 2.2 Logical Model – Translation & Validation Layer (ref 4.2.12.1, 5 & 6)

**Requirement 2.2.1** : To translate from API data to XML-formatted TA TSI messages and vice versa

**Requirement 2.2.2** : To apply multiple levels of syntactic and/or semantic validation of API data against rules and reference data

**Requirement 2.2.3** : To report on data quality, service quality, manageability and volumes.



The Translation and Validation layer of the Common Interface receives data from and sends data to the API layer on the one side and receives from and presents TA TSI messages to the Security and Transport layer of the Common Interface utilising the Common Interface Metadata for its translation and validation rules.

The Translation and Validation layer must be able to handle each of the TA TSI data elements, TA TSI messages and Metadata shown in the Common interface XSD.

The Translation and Validation layer must be able to provide the following functionality :

1. Receive all the TA TSI messages shown in the Common Interface XSD from the Security and Transport layer
2. Present the relevant, translated and validated data elements from these TA TSI messages to the APIs in use at the particular implementation of the Common Interface (as recorded in the Private Metadata).
3. Create all the TA TSI messages from the data elements passed to it by the APIs in use (recorded in the Private Metadata), ensuring that each data element is conformant to TA TSI metadata definitions, both in format and data quality ('annotation' and 'facets' shown in the Common Interface XSD) according to the version of TA TSI messages in use at the destination Common Interface (registered in the Common Metadata) and to pass completed messages to the open Message Queue interface between the Translation & Validation layer and the Security & Transport layer.
4. To reduce implementation effort and encourage early adoption, the translation & validation layer must provide the possibility to include modules to translate existing railway messages to their nearest TAF TSI equivalent such as UIC 407-1 into TA TSI Train Running Information etc (and vice versa). The public metadata must hold the TA-TSI XML Schema shown in the Common Interface XSD, allowing internal

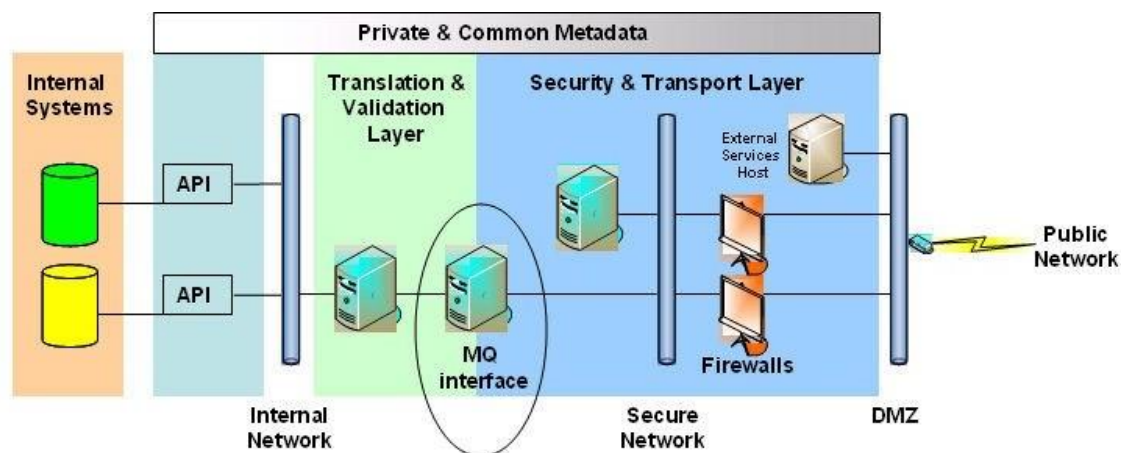
systems to process correctly formatted TA TSI messages into and out of the Queues without Translation.

5. The translation & validation layer must handle the following validation :

- Compliance checking of received messages from the network and logging and sending of error reports to both parties when compliance checks have failed. The software should allow configuration at installation.
- Compliance checking of incoming messages from the internal systems and logging and sending of error reports to the internal party only when compliance checks have failed. The software should allow configuration at installation.
- Management of message rejection at the Translation and Validation layer should make the cause of the rejection clear.
- Management of message rejection at the remote Common Interface, including identification of the cause of rejection.
- Entire rejection of a non-compliant message.

### 2.3 Logical Model – Interface between Translation & Validation Layer and Security and Transport Layer

**Requirement 2.3.1** : Provide a clear message queue interface between the T&V and S&T layers for inbound and outbound messages in order to provide a platform for end-to-end delivery between an application and the trading partner.



This objective will be met by the implementation of an **Open Message Queue** as shown. By using an open Message Queuing interface between the Translation & Validation layer and the Security & transport layer, it is possible for the following functional requirements to be met:

- Only correctly formatted messages are placed on the queue.
- The solution is scalable from a PC to a larger system.

- Inbound messages must be directed to a standard-name inbound public queue which includes company ID.
- Outbound messages must be directed to a standard-name outbound queue which includes the company ID of the recipient.
- All private queue names must be recorded in the Private Metadata.
- Messages received into the inbound queue must then be directed (according to local implementation of the security and transport layer) for example to a queue per application as defined in the private metadata.
- Properties of queues must be configurable as part of the installation process.
- Appropriate security capability must be proposed during development and/or tendering phases.

## 2.4 Logical Model – Security and Transport Layer

**Requirement 2.4.1 :** To provide appropriate security against specific issues

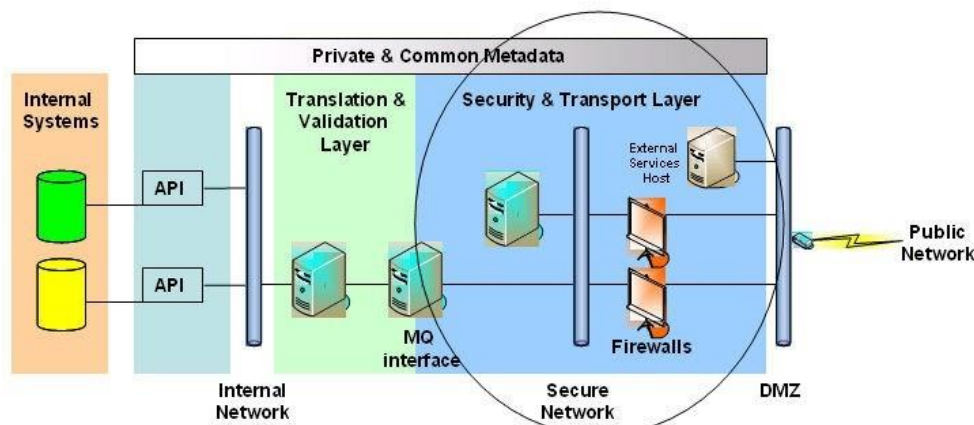
**Requirement 2.4.2 :** To provide a transaction mechanism that gives the authenticated sender of a TA TSI-formatted message on an outbound queue the guarantee that the message is received in the inbound queue of the destination Common Interface.

**Requirement 2.4.3 :** To ensure that messages are sent to the correct destination Common Interface.

**Requirement 2.4.4 :** To handle the delivery of messages through IP networks.

**Requirement 2.4.5 :** To implement asynchronous message exchange.

**Requirement 2.4.6 :** Message delivery from one Common Interface to another Common Interface must be achieved via any available IP network.



The Common Interface must ensure security addressing the following specific issues :

- privacy & confidentiality

- authentication
- integrity
- non-repudiation
- denial of service attack or flooding of queues

Inbound functionality: The Security and Transport layer of the Common Interface receives TA TSI messages from other Common Interface implementations elsewhere in the Rail Industry either by the TA TSI message being placed on the public queue if it is received from a regular trading partner whose connection details are permanently stored in the local firewall, or via an external services host if the message is received from an irregular trading partner whose network connection details are not permanently stored in the local firewall. The Security and Transport layer also dynamically manages the inbound queue to ensure that each queue entry is processed onto the correct receiving queue within 5 seconds, for further processing by the Translation & Validation layer and the API layer.

Outbound functionality: The Security and Transport layer sends TA TSI messages to other Common Interface implementations elsewhere in the Rail Industry by transferring the TA TSI messages presented to it from the Translation and Validation layer of the Common Interface from the outbound message queue within 5 seconds, using the Common Interface Metadata.

External Services Gateway must authenticate casual users using the commonly accepted appropriate security processes. The external services gateway must deny service to non-authenticated users.

The following error handling and reporting is required :

- Queue size
- Request for opening a non-existent Queue
- Message lifetime per queue

#### **2.4.1 Data Compression (ref 4.2.12.1)**

Since the files with Telematics messages may have large size, it is appropriate to perform data compressing. For XML messages, this procedure ensures significant decrease of the data volumes. Therefore the Security & Transport layer of the Common Interface will perform compression of the outgoing and decompression of the incoming TA XML messages. An open industry standard mechanism has to be used for compression/decompression.

Encryption or data compression in Message Queuing is available at 'channel exit time'. The encryption and compression shall be made as defined on 5.5 and 5.6.

## 2.5 Reference Files and Databases (ref 4.2.11.1)

Access to the TA TSI Location and Company Reference files will be granted and managed through the Common Interface by dedicated secure Web service. The URL, WSDL and X509 Certificate can be retrieved from [the](#) body designated for their management according to the provisions of the TA TSI and the Technical Document 106 on PKI. Some of the data in the TA TSI reference files will be part of the public metadata (see TA TSI – ANNEX D.2 : APPENDIX C – REFERENCE FILES, version 2.0) and each will have its own Human-Machine Interface for administration purposes.

## 2.6 Metadata

The Metadata required for operation of the Common Interface is in two parts, Private Metadata (defining the local conditions of the particular implementation of the Common Interface) and Common Metadata (defining all the public information available to all actors and centrally managed and distributed, the TA TSI schemas).

### 2.6.1 Private Metadata

- Specific Implementation Metadata about APIs in use,
- Translation tables from internal system data to and from TA TSI Messages
  - one-to-one mapping
  - code-to-code mapping
  - units translation (accuracy, rounding)
- Translation of message header information
- Audit and logging settings
- Administrative rights to amend the metadata (User & system profiles, access rights, authentication details, security)
- Local Transport protocol & system information linking the APIs and Transport & Validation layer (refer to documentation regarding systems referred to in section 2.2)
- Single or multiple instances of Private Metadata may be required.

### 2.6.2 Common Metadata API Metadata

- Generic information describing the implementation of APIs
- API audit and logging requirements

### Translation & Validation Metadata (ref 4.2.12.1, 5 & 6)

- TA TSI Message Metadata (XSD) - ERA-TD-105: TA TSI – Annex D.2 : Appendix F - TA TSI Data and Message Model

- Data quality (validation)
- Registered Common Interface installations (addresses and Information about message versions in use at each destination Common Interface)

### **Security & Transport layer Metadata (ref 4.2.12.2, 3, 4 & 6)**

- Partner definitions including partner queue names, TCP/IP definitions for each partner, network(s) to be used for each partner, service times for each partner.
- Open Message Queue definitions, names, restart/recovery, persistency, time outs, dead letter queue, maximum size, heartbeat checking
- Channel definitions queue manager to queue manager
- Logging definitions, locations, size, cyclic parameters etc
- Security definitions
- Message definitions (these override defaults above on a per message basis) expiry, triggering, priorities
- Error handling
- Naming standards
- LDAP usage, queue names
- Performance of Common Interface throughput, response times
- Technical implementation, scalability
- Support details
- Security information (Authentication and security data for messages, Reference file, WIMO access information and encryption keys)

Details of Transport layer are described in chapter 4 to 6

#### **2.6.3 Queue Naming**

The addressing of partners should be defined in the URL and the operation according chapter 4 to 6. Host names /IP Address / CI Instances needs to be agreed bilaterally. For security reason there will be no central Register. The Common Interface should not require the internal application to know the public queue names – the application should receive a message - or request that a message is sent - to/from the actors, not the public queues.

Private queues may be named as required for the operation of each specific implementation of the Common Interface. Private Queue names will be stored in the Private Metadata.

### 2.6.4 Metadata Management & Distribution

The common metadata required for operation of the Common Interface is published on the Agency's website: [https://github.com/EU-Agency-for-Railways/TSI\\_TAF/tree/master](https://github.com/EU-Agency-for-Railways/TSI_TAF/tree/master). The common metadata must be implemented using a secure, replicated metadata repository capable of secure, automatic distribution in near real-time to the Common Interface metadata instances in use by the actors. Change management for the metadata for TAF TSI messages, is the responsibility of the TAF TSI Change Control Management Working Party run under the aegis of the European Union Agency for Railways.

Private metadata may be managed by each individual actor implementing the Common Interface.

## 2.7 Human-Computer-Interface (ref 4.2.12.1)

The following administrative functionality is required as a minimum for each layer in the Common Interface

- Start/stop/reset
- Parameterised process activity logging (entry, translation/validation & exit of the Translation & Validation layer)
- Monitoring of activity in real-time
- Automatic Queue Recovery
- Version management of the Metadata.

This functionality must be available via a standard internet browser.

## 3 Data quality

### 3.1 Data Quality (ref 4.4.1)

#### 3.1.1 Prerequisite

Chapter 4.4.1 of the Telematics Application Services Sub System (TA TSI) documents the essential requirements for Data Quality. This is a prerequisite for effective data exchange and comprises the following elements:

- Completeness
- Accuracy
- Consistency
- Timeliness

The sender of each message will be responsible for the correctness of the data sent and must verify that it is in compliance with the guidelines stipulated for that message. This means that the data must not only be complete and conform to the metadata requirements (syntax-level), but must also be accurate, timely and consistent for the receiving application to effectively import the message. This requires two distinct levels of validation, as described below.



### 3.1.2 Level 1 Compliance Checking

As the TA TSI messages are defined using WC3 XSDs according to **Recommendation 28**, the schema contain all metadata needed for strict Level 1 compliance checking. This syntactical-level check validates the interchange, or part of it, for compliance with the schema. This checking normally happens at the translation and validation layer, before the data is treated by the API. It includes validation for field lengths, data types, codification (where enumerations exists), presence or absence of required data, valid payload entries where defined and the order of data transmitted. The schema validation is more robust and provides a higher level of compliance checking than traditional EDI.

The XSD metadata provides a perfect solution to meet the needs for Completeness and some of the Accuracy requirements as defined above.

As a minimum, Level 1 Compliance Checking shall be implemented. This shall be part of the Common Interface Translation and Validation Layer.

### 3.1.3 Level 2 Application Validation

According to the TA TSI the originator of the message must ensure a data quality assurance check using their own resources. Data quality assurance includes comparison of data from reference file databases provided as part of the TSI plus, where applicable, logic checks to assure the timeliness and continuity of data and messages.

Data must be of high quality if they are fit for their intended uses, which means they

- Are Error free: accessible, accurate, timely, complete, consistent with other sources, etc., and
- Possess desired features: relevant, comprehensive, proper level of detail, easy-to-read, easy-to-interpret, etc.

For example, while the Schema can validate that a Company (Organisation) Ident contains 4 AN digits, it cannot assess the validity of that code against a common reference file in the translation and validation layer. It is therefore up to the sender to assure that the information is valid in his own application before generating the message. The receiver must also perform the same validity check before the data is imported into his system. Additionally, Level 2 Application Validation should also provide consistency and timeliness checks according to the requirements defined by the target application.

This level of validation is a function of the internal systems as it presupposes that the necessary reference data are in place and applied consistently in the senders' systems.

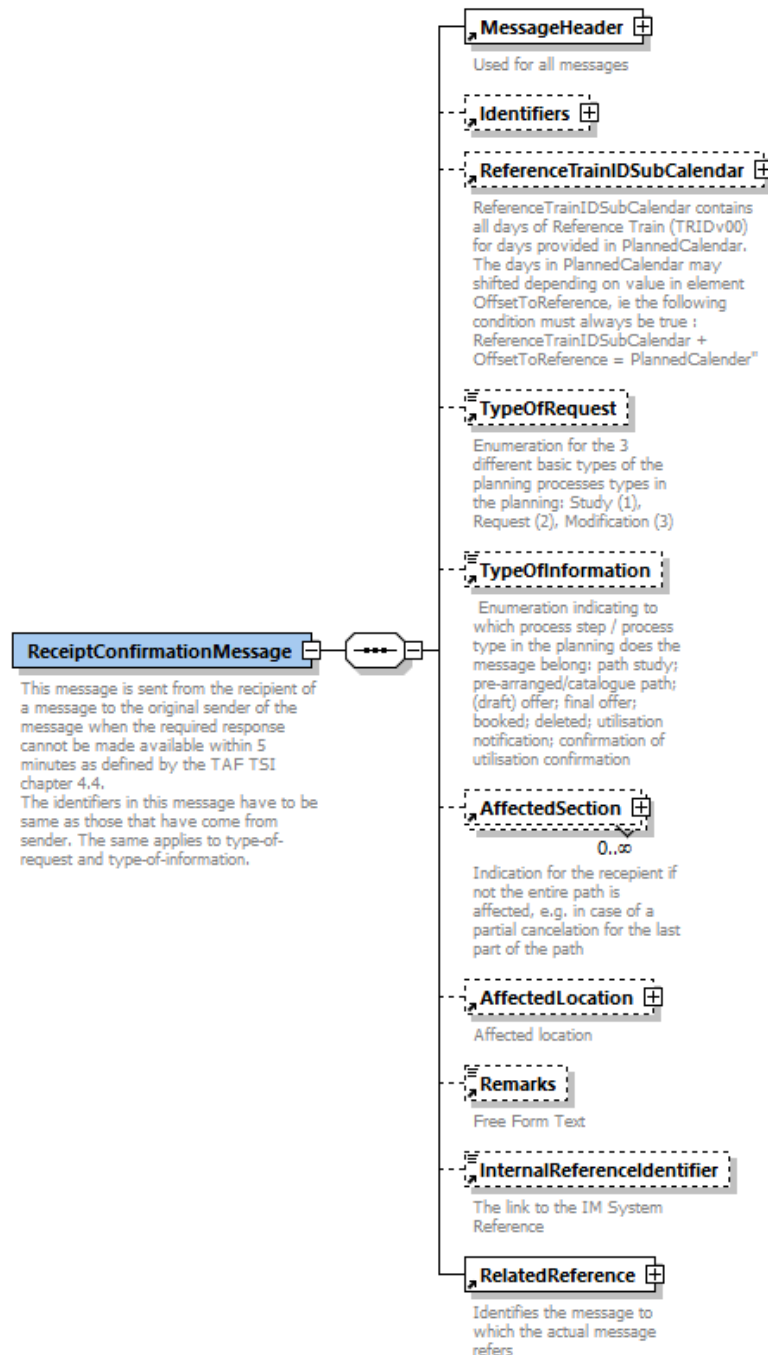
### 3.1.4 TA Acknowledgments - Receipt Confirmation Message (ref 2.3 and 2.5)

Message acknowledgement can be positive or negative. These messages have the ability to communicate specific application syntax error back to the sender. The messages also have the capability to inform the sender whether the message have been accepted or rejected by the application.

Messages that require acknowledgement are those which update and delete database entries (excluding event reporting) and those which have application or technical errors.



Based on these requirements, messages have been developed to fulfil the application level requirements. For the Common Interface this message is as any of the messages of the TA TSI catalogue. The message and its use are defined in Appendix C, index [2] and Appendix C, index [100] of the TSI.

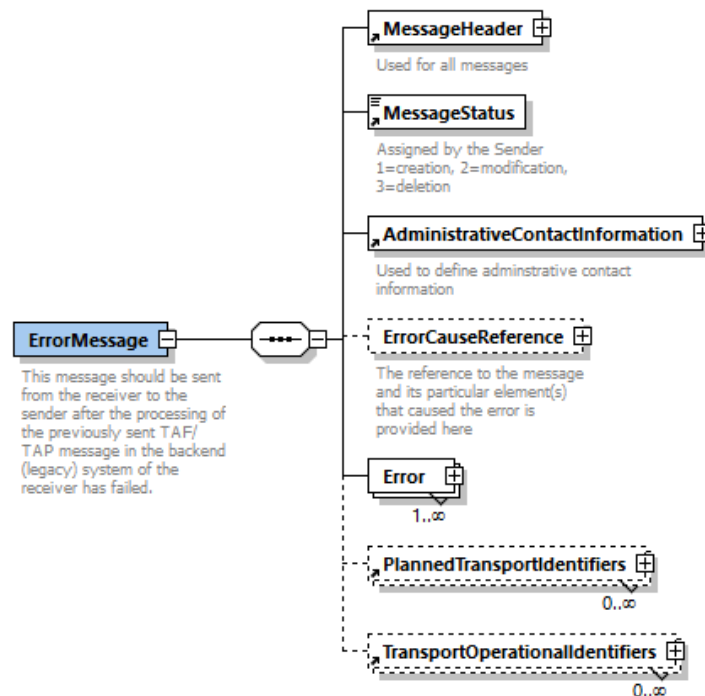


This message is sent from the recipient of a message to the original sender of the message when the required in the TA TSI chapter 2.3 and 2.5. The identifiers in this message have to be the same as those that have come from sender. The same applies to type-of-request and type-of-information. This message serves also as a link back to the original message being acknowledged.

## Error Message

The message is used in any situation when an error has occurred:

- technical causes when, for example, the legacy system is down
- functional causes: identifier sent in the message does not match the object in the receiving system; the wrong data sent in the parameters list in the message; the legacy system has detected a logical error in the data payload of the message etc.
- both technical and functional



The message and its use are defined in Appendix C, index [2] and Appendix C, index [100] of the Telematics Applications TSI.

There are two levels of message checking and validation as mentioned in 3.2.5 and 3.2.6.

### 3.1.5 Level 1 – Compliance Checking (ref 4.2.12.6 & 4.4.1)

This action (acknowledgement or rejection) indicates the result of a syntactical check of the complete received XML document. This shall be done automatically within Common Interface Translation and Validation Layer.

### 3.1.6 Level 2 – Application Validation (ref 4.2.12.6 & 4.4.1)

This action (acknowledgement or rejection) indicates the result of an application validation of the complete received XML document.

The messages Receipt Confirmation and Error Message are used to fulfil the Level 2. The Level 1 shall be fulfilled automatically by Common Interface Translation and Validation Layer, to prevent the syntactically

incorrect messages already at the sending time, before such a message would have been sent to the recipient.

## 4 Communication layer between Common Interface

The communication between Common Interface is determined by

- Content
  - Message Metadata including Header,
  - Encryption, signature, compression
- Functional
  - Push principle
  - Heartbeat between CI
- Technology
  - Protocol
  - Security

The communication between CI is done on SOA (Service-Oriented Architecture) principles using Webservice on communication Layer with transport protocol Https

Chapter 5 describes the communication Layer between two Common Interface to be compliant to the CI Reference Implementation developed by the TAF TSI Common Components Group.

Further “LI” (local instance of an CI) is also used as synonym of CI (LI=CI)

## 5 Web service

The communication between LI is possible in Web Service. The Web Service communication is SOAP over HTTP(S) where the request message and response are both in XML or JSON. The message to be sent should be enclosed in a SOAP request and the response xml will also be enclosed in a SOAP response.

As in TA the message exchange is in push mode the Web service has always to be available on the receiving common interface to be invoked from the sending Common Interface.

For the sending LI it is an outbound communication for the receiving CI this is an inbound communication. Therefore, the Web service to be published is for inbound Communication

This Web Service is only for the purpose of sending TA TSI standard messages to Remote partner using a CI.

Before sending Messages communication details must be agreed bilaterally:

- Sender Company
- Receiver Company
- Public host names/IP Address for sending and receiving (may be different)
- CI Instance Number
- Communication mode (default is Web Service)
- Protocol (default is HTTPs)
- Port (default is 443)
- Sender LI IP Address (This is a string value and may be different for as public host names, is value "uicmh: messageLiHost" in SOAP Header properties according 5.2.1.

From functional part Message types and Version needs to be agreed and if messages are compressed, encrypted or signed.

### 5.1 WSDL

The WSDL to be used is described in Annex 1

### 5.2 Request

The xml is to be sent by the SOAP request. The request is a SOAP envelope which contains a SOAP Header and a SOAP body. The Header contains some custom properties which defines metadata related to the message. The SOAP body contains the actual xml message and information related to signature and encoding. Both the header and the body are described in details in the sections below.

The xml message payload in the request can be optionally compressed for performance, and encrypted and signed for security purposes.

### 5.2.1 SOAP Header Properties

As it can be seen in the WSDL the SOAP Header contains four custom headers. These headers are used to define the message. The namespace used for all the headers is

**xmlns:uicmh="http://uic.cc.org/UICMessage/Header"**

- uicmh:messageIdentifier

This header property should contain a string value, which uniquely identifies the message. To conform to TAF/TAP-TSI standard, the unique value should be 16 byte GUID expressed as 36- character string (32 hexadecimal numbers and 4 minus characters). This value should be same as MessageIdentifier used in TAF-TSI message.

Example	45d0c713-111f-4a84-bdfd-3a32c9012860
---------	--------------------------------------

- uicmh:compressed

This is a Boolean value (true/false). If the message is compressed, then it should be set to `true` else it is set to `false`.

For more details on compression, refer to the section 5.5 on message compression.

- uicmh:encrypted

This is a Boolean value (true/false). If the message is encrypted, then it should be set to `true` else it is set to `false`.

For more details on encryption, refer to the section 5.6 on message encryption.

- uicmh:signed

This is a Boolean value (true/false). If the message is signed with a certificate, then it should be set to `true` else it is set to `false`.

For more details on signing, refer to the section 5.7 on message signing.

- uicmh: messageLiHost

This is a string value which should contain the Sender LI IP Address (should be public IP of LI). For example: 122.109.101.100

### 5.2.2 SOAP Message Body

The SOAP Body contains the TAF/TAP-TSI message payload, the encoding definition and optional digital signature.

As it can be seen in the wsdl, the Soap body contains the type uicm:UICMessage (xmlns:uicm=http://uic.cc.org/UICMessage ). The UICMessage contains four elements as described below.

- message

This element contains the actual TAF/TAP TSI message. If the message is compressed, encrypted or signed, then it should be base64 encoded.

- signature and senderAlias

Message can be digitally signed for security purpose. For signing, you have to obtain a client certificate (including private key) from the body designated for their management according to the provisions of the TA TSI and the Technical Document 106 on PKI. If messages are signed, then the signature field should contain the digital signature.

The client certificate (public key) and the alias name you obtain needs be shared with your CI partner. The CI partner needs import that certificate into their trusted keystore (TrustStore) including the senderAlias. This alias name should be filled in the **senderAlias** field.

For more details on signing, refer to the section 5.7 on message signing.

- encoding

The encoding field should contain the char encoding used for the message. CI supports the following char encodings:

- UTF-8

### 5.2.3 Sample SOAP Messages

Two sample messages are annexed to this document. One is with compression (Annex 2), encryption and signing, and one without them (Annex 3).

## 5.3 Response

The response of the Web service is an acknowledgement XML enclosed in a SOAP envelope. The acknowledgement xml in the response contains the response status information about the message, and sender and receiver information. The schema for the acknowledgement xml and a sample acknowledgement response message is annexed to this document as annex 4 and 5. The elements of the acknowledgement are described below.

- **ResponseStatus:** The value of this field is either ACK or NACK meaning positive and negative acknowledgment respectively. A positive acknowledgment means the partner CI has received the message and accepted it for processing. A negative acknowledgment means, the partner CI has received the message but rejected it as the CI is not partner configured to receive message from the sender using Web Service.
- **AckIdentifier:** This is a unique Identifier for the acknowledgement. In the reference implementation for the CI it is generated by prefixing ACKID to the message Id from received Message.
- **MessageReference:** This element contains four child elements MessageType, MessageTypeVersion, MessageIdentifier, MessageDateTime. These elements contain values from the corresponding fields in the Header part of the received TAF/TAPTSI message.
- **Sender:** Id of the sender company found in the TAF-TSI Message header

- Recipient: Id of the receiver company found in the TAF-TSI Message header
- RemoteLIName: The Name of the remote CI as configured in reference implementation for the CI, max 50 characters CI.
- RemoteLIInstanceNumber: The instance Id of the Remote CI as configured the reference implementation for the CI, max 2 numericCI.
- MessageTransportMechanism: The transport mechanism which is always WEBSERVICE in this case.

Note: The acknowledgement only indicates if the message is accepted by the CI for processing or not. It does not indicate if the message is successfully processed or not.

An example is annexed as Annex 5

## 5.4 Invocation

### 5.4.1 HTTPS Invocation

The CI must use https protocol with predefined HTTPS profile to achieve interoperability (with HTTPS version, certificate profile, allowed ciphers, mutual authentication)

and the following URL can be used to invoke the Web Service where the <CI-IP> is the IP address or host name of the CI and <CI\_HttpsPort> is the port CI is listening for the request. For <CI\_HttpsPort> default is 443. The IP address have to be exchanged in Dataexchange agreement between the partners

`https://<CI- IP>:<CI_HttpsPort>/LIMessageProcessing/http/UICCCMessageProcessing/UICCCMessageProcessingInboundWS`

It should be possible to get the WSDL on the following URL:


`https://<CI- IP>:<CI_HttpsPort>/LIMessageProcessing/http/UICCCMessageProcessing/UICCCMessageProcessingInboundWS?wsdl`


Note: TLS should be used to encrypt the communication channel based on a client certificate (including private key) from the body designated for their management according to the provisions of the TA TSI and the Technical Document 106 on PKI, .


For secure HTTP communication the HTTP/2 version or higher should be used in line with the following specifications:



## 5 Secure Communication for HTTP

**SP-SEC-Comm-5-1**,  - The HTTP endpoint shall use TLS for HTTP communication according to 3 - End-to-End Security Layer (TLS).

, **SP-SEC-Comm-5-2** - Exceptions for the usage of TLS for HTTP communication are defined in the Shared Cybersecurity Services Specification.

, **SP-SEC-Comm-5-3** - Requirements for human user authorization in HTTP communication are defined by the corresponding interface specification. (SP-SEC-SERV User Authentication Service)

### 5.5 Message Compression

The TA-TSI message payload in the request can be compressed in order to save bandwidth and for faster communication. The compression is optional and if it is compressed the corresponding Header property should be set to true (see section 5.2.1)

CI can use commonly available compression algorithms for message compression and decompression (such as ZLIB or similar to ensure interoperability and backwards compatibility between the involved parties) not protected by any patents.

### 5.6 Message Encryption

To ensure backwards compatibility, the message must be encrypted and decrypted using a secret key. In the reference implementation for the CI the key is derived from the TLS certificate (older encryption methods are permitted to ensure backwards compatibility of the Common Interface). Therefore, for encryption you have to obtain a valid client certificate (including private key) from [the](#) body designated for their management according to the provisions of the TA TSI and the Technical Document 106 on PKI. It is the same as used for TLS on transport layer secure HTTP communication the HTTP/2 version or higher should be used in line with the following specifications:

### 5.7 Message Signing

The TA-TSI message payload in the request can be digitally signed for security. The signing is optional and if it is signed, then the corresponding Header property should be set to true (refer to section 5.2.1).

In the reference implementation for the CI the message is signed using the private key of the TLS certificate and verified using the public key. The signing is done with SHA256 and Elliptic Curves (public-key cryptography based on the algebraic structure of elliptic curves over finite fields)..

If messages are signed an agreement between partners is mandatory, as also to provide the client certificate with public key and sender Alias name to the receiving Partner to enable the receiver to verify the signature.

## 6 REST

Alternatively to WSDL, the REST Web Service can be used which is an implementation of a web-based system that follows the principles of Representational State Transfer (REST) to enable communication to the local legacy systems.

REST web services provide a lightweight, flexible way to exchange data using standard HTTP methods such as GET, POST, PUT or DELETE.

The REST webservice to be used is described below:

### 6.1 Request

REST API is needed with the necessary endpoint GET / WagonNumberFreight/{id} to retrieve a specific rolling stock entry by ID.

```
GET /api/RSRD HTTP/1.1

Host: RSRD.com
Content-Type: rsrd/json
Authorization: Bearer <your_token>

{
  "MessageHeader": {
    "timestamp": "2025-04-13T17:38:00Z",
    "source": "system_X"
  },
  "WagonNumberFreight": [
    {
      "wagonId": "128734567890",
      "type": "freight",
    },
  ]
}
```

## 6.2 Response

Above REST GET example delivers following response from RSRD (example):

```
HTTP/1.1 200 OK
Content-Type: RSRD/json

{
  "AdministrativeDataSet": {
    "WagonNumberFreight": "128734567890",
    "PreviousWagonNumberFreight": "128734567899",
    "RegistrationCountry": "FR",
    "DatePutIntoService": "2020-05-20",
    "AuthorisationValidUntil": "2025-12-31",
    "SuspensionOfAuthorisation": false,
    "DateSuspensionOfAuthorisation": null,
    "MultilateralAuthorisationCountries": [
      "FR",
      "DE",
      "IT"
    ],
    "ChannelTunnelPermitted": true,
    "QuieterRoutesExemptionCountry": [
      "GB"
    ],
    "KeeperShortNameVKM": "VKM12345",
    "ECM": "Example ECM Inc.",
    "PlannedChangeOfECM": {
      "CurrentECMAssignedUntil": "2023-12-31",
      "SubsequentECM": "Next ECM Inc."
    },
    "ECMCertificate": {
      "EINNumber": {
        "CountryCodeISO": "FR",
        "TypeDocumentEIN": "31",
        "CounterAccreditedRecognizedBody": "02",
        "EINYear": "20",
        "EINCounter": 100
      },
      "ECMCertificateValidFrom": "2021-01-01",
      "ECMCertificateValidTo": "2026-01-01",
      "CoversTankWagonsForDangerousGoods": false
    },
    ...
  }
}
```

Each request from a client to a server must contain all the information needed for the server to understand and process it. This means that the server does not store any client context between requests, which makes the system scalable and simplifies the server-side design.

REST is used by the Common Interface only for legacy connectors and not for connection to other Common Interface Local Instances.

## 7 Web service for Remote LI heartbeat check

This web service is used for checking the status of Remote LI. This is to know on an early stage if no message exchange is happening that the Remote LI is alive.

An LI sends a SOAP request to RLI and the Remote LI will send a SOAP response to sender LI. The request is optional. The response is mandatory.

The minimum time between two requests is 5 minutes.

Response has to be given within a maximum time frame of 5 seconds. WSDL

The WSDL is attached as Annex 6.

### 7.1 Request

The request is a SOAP envelope which contains a SOAP body. The SOAP body contains the actual heartbeat check message and few other parameters. SOAP message body is described in detail in the sections below.

#### 7.1.1 SOAP Header

The SOAP Header do not contain specific properties.

#### 7.1.2 SOAP Message Body

The SOAP Body contains the message payload

As it can be seen in the WSDL, the Soap body contains the type UICHBMessage. The UICHBMessage contains two elements as described below:

- message - This element contains the heart beat check message – “Are you alive?”
- Properties

Below is the list of properties used:

- Remote LI's Instance number
- Sender Host IP
- LI Instance Number
- LI to LI Transport Mode
- LI Name
- Remote LI IP
- Remote LI port
- Remote LI name

### 7.1.3 *Sample Heart Beat Request SOAP Message*

An example is annexed as Annex 7

## 7.2 Response

The response of the Web service is an acknowledgement XML enclosed in a SOAP envelope. The acknowledgement xml in the response contains the status information about the remote LI as “HEART\_BEAT\_WS\_RECEIVED”.

An example is annexed as Annex 8

## 7.3 Invocation

### 7.3.1 *HTTPS Invocation*

If CI can be reached with https protocol, then the following URL can be used to invoke the Web Service where the <CI-IP> is the IP address or host name of the CI and <CI\_HttpsPort> is the port CI is listening for the request. For <CI\_HttpsPort> default is 443. You can communicate with CI admin to know the IP and the port.

To the get the WSDL, use the following URL:

`https://<CI-IP>:<CI_HttpsPort>/LIServices/LIHBMessage?wsdl`

Note: For TLS communication, the certificate has to be provided from the same certification authority. The certification authority is the body designated for their management according to the provisions of the TA TSI and the Technical Document 106 on PKI. The same certificate as for messaging must be used.

For REST services Health Check Endpoint (/health, /status) can be used towards an endpoint that returns system status.

## ANNEX 1 MESSAGE EXCHANGE WSDL

```

<wsdl:definitions name="LIReceiveMessageService" targetNamespace="http://uic.cc.org/UICMessage"
xmlns:ns1="http://uic.cc.org/UICMessage/Header" xmlns:ns2="http://schemas.xmlsoap.org/soap/http"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://uic.cc.org/UICMessage"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:documentation>This WSDL describes the communication
    protocol for sending TA-TSI standard messages to partner
    using Common Interface (CI).For deatailed Documentation
    please refer to the CI Inbound Web Service Description
    Document</wsdl:documentation>
  <wsdl:types>
    <xs:schema elementFormDefault="unqualified" targetNamespace="http://uic.cc.org/UICMessage"
version="1.0" xmlns:tns="http://uic.cc.org/UICMessage" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xs:element name="UICMessage" type="tns:UICMessage"/>
      <xs:element name="UICMessageResponse" type="tns:UICMessageResponse"/>
      <xs:complexType name="UICMessage">
        <xs:sequence>
          <xs:element minOccurs="0" name="message" type="xsd:anyType"/>
          <xs:element minOccurs="0" name="signature" type="xsd:anyType"/>
          <xs:element minOccurs="0" name="senderAlias" type="xsd:anyType"/>
          <xs:element minOccurs="0" name="encoding" type="xsd:anyType"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="UICMessageResponse">
        <xs:sequence>
          <xs:element minOccurs="0" name="return" type="xsd:anyType"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
    <xsd:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified"
targetNamespace="http://uic.cc.org/UICMessage/Header"
xmlns="http://uic.cc.org/UICMessage/Header"
xmlns:tns="http://uic.cc.org/UICMessage"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="messageIdentifier" nillable="true" type="xsd:string"/>
      <xsd:element name="messageLiHost" nillable="true" type="xsd:string" />
      <xsd:element name="encrypted" nillable="true" type="xsd:boolean"/>
      <xsd:element name="compressed" nillable="true" type="xsd:boolean" />
      <xsd:element name="signed" nillable="true" type="xsd:boolean"/>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="UICMessage">
    <wsdl:part element="tns:UICMessage" name="parameters">
      </wsdl:part>
    <wsdl:part element="ns1:messageIdentifier" name="messageIdentifier">
      </wsdl:part>
    <wsdl:part element="ns1:messageLiHost" name="messageLiHost">
      </wsdl:part>
    <wsdl:part element="ns1:compressed" name="compressed">
      </wsdl:part>
    <wsdl:part element="ns1:encrypted" name="encrypted">
      </wsdl:part>
    <wsdl:part element="ns1:signed" name="signed">
      </wsdl:part>
  </wsdl:message>

```

```

</wsdl:message>
<wsdl:message name="UICMessageResponse">
  <wsdl:part element="tns:UICMessageResponse" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:portType name="UICReceiveMessage">
  <wsdl:operation name="UICMessage">
    <wsdl:documentation>This operation is used to send the
      message to the Remote CI.</wsdl:documentation>
    <wsdl:input message="tns:UICMessage" name="UICMessage">
    </wsdl:input>
    <wsdl:output message="tns:UICMessageResponse" name="UICMessageResponse">
    </wsdl:output>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="LIReceiveMessageServiceSoapBinding" type="tns:UICReceiveMessage">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="UICMessage">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="UICMessage">
      <soap:header message="tns:UICMessage" part="messageIdentifier" use="literal">
      </soap:header>
      <soap:header message="tns:UICMessage" part="messageLiHost" use="literal">
      </soap:header>
      <soap:header message="tns:UICMessage" part="compressed" use="literal">
      </soap:header>
      <soap:header message="tns:UICMessage" part="encrypted" use="literal">
      </soap:header>
      <soap:header message="tns:UICMessage" part="signed" use="literal">
      </soap:header>
      <soap:body parts="parameters" use="literal"/>
    </wsdl:input>
    <wsdl:output name="UICMessageResponse">
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="LIReceiveMessageService">
  <wsdl:port binding="tns:LIReceiveMessageServiceSoapBinding" name="UICReceiveMessagePort">
    <soap:address location="https://10.10.200.213/LIServices/LIReceiveMessage"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

## ANNEX 2 SAMPLE MESSAGE-WITH-COMPRESSION-SIGNING- ENCRYPTION

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:uicm="http://uic.cc.org/UICMessage">
  <soap:Header>
    <uicmh:signed xmlns:uicmh="http://uic.cc.org/UICMessage/Header">true</uicmh:signed>
    <uicmh:encrypted xmlns:uicmh="http://uic.cc.org/UICMessage/Header">true</uicmh:encrypted>
    <uicmh:compressed xmlns:uicmh="http://uic.cc.org/UICMessage/Header">true</uicmh:compressed>
    <uicmh:messageIdentifier xmlns:uicmh="http://uic.cc.org/UICMessage/Header">d990c1c8-a869-478f-9ca0-
830f786c1b86</uicmh:messageIdentifier>
    <uicmh:messageLiHost
xmlns:uicmh="http://uic.cc.org/UICMessage/Header">122.109.101.100</uicmh:messageLiHost>
  </soap:Header>
  <soap:Body>
    <uicm:UICMessage>
      <message>eNoADUDyv9Ab1S8z7EuwlNjyeXfsc3E428xW8Hf7mMyAi+08ufECDInjs1ES3AM05W1owgmhVIGi
u8mCwEmVaybcBB5pLp90Yh5tBQGVZcLBBQwizZLGfeQbs7SAK6ZJfEt43Ial4KTCb4y06wYmYfDa
MQ9ABJzZqtYdc65vUNCuDedwMR33RnZYYWFUaBqB+UHRdksMrVgcwB8J1gw9J1ulhWeucp4SsL0
V+VveADK9hAFOpZDDI05PKmKqtdTL/vfkEajfy8cl2CZQcS3j53d6jd+GSC0RMibtkH9iexrgjQ5
tSZbIAmyimCgaiNweCMIbEJqOVcNlq09JBMP5nR6bE074CKZ5RxeOjFNqizWDYyt//Z8Z8OnUku1
Pk8ixC8tWWo6DtRdCP000uyauzP5S8HVvl63IK7hip5cGIRD0vkKlx5ySu4e6EmrBZA1E3FavXGj
+JKMzY/LLxjJfc0hjFN44Ce2Mw8v8dUouoX9SRUvLIwdvblvLFCWvBXE1AJmwwSBZMFH0/axC8vcp
b+8jx+lZwTaST5sFfQyOHR04tzCUZimouEU0TBEMTfzDWrksQrS+3M26i5t/Vc2k9f7he3CEomZL
cvvzlg9B+bjePEqimm9jfgXR8UDW7h5llotcfMNHb1OfR4BPbgVOxDPE6cLPmSUKIEBAX1wgj026
ETI7382s2L5RbPY5tu74N+DAWVqsl+qD3ImaqlW/UjtSgs9+9JL4kZEJiSsfV7OKoKvC5nZ21SwG
3zMLbqsEZn4ulnncJstx57tjby3IDSSeg7QeJYGchFWUvw2sv3xGrhgF5HWQkBM+Li4g1xEmrp0L
ZXf8oA3jtfF+bLeo0ecEtDkoZg5JaQn7TfCbo/o7Z8/3A4OuzibllwmOrDPIlxSF5wofDCQO6Rsl
B4bLHQFc5iRHTv1TNCwm7biDzcc4dLIH6M+Px38MdlOzCcu7D+8kkVWw4NXjNaztqMu2IMJD3GZ5
XCvCts1XKmnR6puLS3wyTUolmj2zHIFBv6dCvGpbld4/tD79Y/mhCqrsPlonF33EGvsuEnHijN NTYdxRSKKp8=
      </message>
    <signature>vu3MQA/Oi8M3GqGodD+FApM2Oq4j2UVK3z2jvS95JaxgOopDUr7d/EL83/92mhTATG/szvZcoFu
RNEec6xZNQGzp1ISnb7XY7AjLT7k2S0AuCRkgJoHZ6hJtDF7eMo/Xwy+81pe/dFhr2TZNJGRMp0t
tgAxhmDdLX4rljIC2Pg=</signature>
    <senderAlias>sdw1458</senderAlias>
    <encoding>UTF-8</encoding>
  </uicm:UICMessage>
</soap:Body>
</soap:Envelope>
```



## ANNEX 3 SAMPLE MESSAGE-WITHOUT-COMPRESSION-SIGNING- ENCRYPTION

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:uicm="http://uic.cc.org/UICMessage">
  <soap:Header>
    <uicmh:signed xmlns:uicmh="http://uic.cc.org/UICMessage/Header">false</uicmh:signed>
    <uicmh:encrypted xmlns:uicmh="http://uic.cc.org/UICMessage/Header">false</uicmh:encrypted>
    <uicmh:compressed xmlns:uicmh="http://uic.cc.org/UICMessage/Header">false
    </uicmh:compressed>
    <uicmh:messageIdentifier xmlns:uicmh="http://uic.cc.org/UICMessage/Header">796cc4ee-cd51-49c2-a8cf-
3bd73b29
ddfa    </uicmh:messageIdentifier>
    <uicmh:messageLiHost
xmlns:uicmh="http://uic.cc.org/UICMessage/Header">122.109.101.100</uicmh:messageLiHost>
    </soap:Header>
    <soap:Body>
    <uicm:UICMessage>
      <message>
        <TrainCompositionMessage>
          <MessageHeader>
            <MessageReference>
              <MessageType>2201</MessageType>
              <MessageTypeVersion>5.1.8</MessageTypeVersion>
              <MessageIdentifier>796cc4ee-cd51-49c2-a8cf-3bd73b29ddfa
              </MessageIdentifier>
              <MessageDateTime>2011-12-
08T14:25:30.910+05:30</MessageDateTime>
            </MessageReference>
            <Sender CI_InstanceNumber="">3025</Sender>
            <Recipient CI_InstanceNumber="">0074</Recipient>
          </MessageHeader>
          <PathIdentity>
            <PathIdent>48119</PathIdent>
            <PathDeparturePoint>
              <CountryCodeISO />
              <LocationPrimaryCode />
              <PrimaryLocationName />
              <LocationSubsidiaryCode
                LocationSubsidiaryTypeCode="" />
            </PathDeparturePoint>
            <PathDepartureTime>2011-04-11T20:05:00+02:00</PathDepartureTime>
            <PathDestinationPoint>
              <CountryCodeISO />
              <LocationPrimaryCode />
              <PrimaryLocationName />
              <LocationSubsidiaryCode
                LocationSubsidiaryTypeCode="" />
            </PathDestinationPoint>
            <PathDestinationTime />
          </PathIdentity>
          <TrainNumber>
            <PathIdent>48119</PathIdent>
            <ScheduledDepartureDateTime>2011-04-11T20:05:00+02:00
            </ScheduledDepartureDateTime>
            <PathDepartureLocation>
              <CountryCodeISO />
              <LocationPrimaryCode />
              <PrimaryLocationName />
              <LocationSubsidiaryCode
                LocationSubsidiaryTypeCode="" />
            </PathDepartureLocation>
          </TrainNumber>
        </MessageHeader>
      </message>
    </uicm:UICMessage>
  </soap:Body>
</soap:Envelope>

```

```

</TrainNumber>
</PathIdentity>
<TrainCompositionJourneySection>
  <JourneySection>
    <IntermediateDestination>
      <CountryCodeISO />
      <LocationPrimaryCode />
      <LocationSubsidiaryCode
        LocationSubsidiaryTypeCode="" />
    </IntermediateDestination>
    <IntermediateArrivalTime>2011-04-12T04:30:00+02:00
    </IntermediateArrivalTime>
    <IntermediateDepartureTime />
    <ResponsibilityActualSection>
      <ResponsibleRU>0083</ResponsibleRU>
      <ResponsibleIM />
    </ResponsibilityActualSection>
    <ResponsibilityNextSection>
      <ResponsibleRU>IT</ResponsibleRU>
      <ResponsibleIM />
    </ResponsibilityNextSection>
  </JourneySection>
  <Locolent>
    <TractionType />
    <TractionIdent />
    <TractionMode />
  </Locolent>
  <TrainRunningData>
    <TrainRunningTechData>
      <TrainWeight>461900</TrainWeight>
      <TrainLength>452</TrainLength>
      <Locolent>
        <TractionType />
        <TractionIdent />
        <TractionMode />
      </Locolent>
      <TrainCC_System />
      <TrainRadioSystem />
      <TrainMaxSpeed />
      <MaxAxleWeight>9.366666666666665</MaxAxleWeight>
      <BrakeType />
      <BrakeWeight />
    </TrainRunningTechData>
  </TrainRunningData>
  <LivestockIndicator />
  <TrainWagonOrder>
    <WagonNumberFreight>238742727825</WagonNumberFreight>
    <WagonTrainPosition>1</WagonTrainPosition>
  </TrainWagonOrder>
  <WagonData>
    <WagonNumberFreight>238742727825</WagonNumberFreight>
    <WagonTechData>
      <WagonLength>2650</WagonLength>
      <WagonNumberOfAxles>3</WagonNumberOfAxles>
      <BrakeType>3</BrakeType>
      <BrakeWeight>28</BrakeWeight>
      <WagonMaxSpeed>100</WagonMaxSpeed>
      <MaxAxleWeight>9.366666666666665</MaxAxleWeight>
      <WagonWeightEmpty>26700</WagonWeightEmpty>
    </WagonTechData>
  </WagonData>

```

```
        <ActivityType />
        <ExceptionalGaugingInd />
        <TotalLoadWeight />
    </WagonData>
    <ActivityType />
    <ExceptionalGaugingInd />
    <TotalLoadWeight />
</TrainCompositionJourneySection>
</TrainCompositionMessage>
</message>
<encoding>UTF-8</encoding>
</uicm:UICMessage>
</soap:Body>
</soap:Envelope>
```

## Annex 4 Schema for the acknowledgement XML

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="LI_TechnicalAck">
    <xs:annotation>
      <xs:documentation>This schema describes the UIC-LI-LI ACK message</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ResponseStatus"/>
        <xs:element ref="AckIdentifier"/>
        <xs:element ref="MessageReference"/>
        <xs:element ref="Sender"/>
        <xs:element ref="Recipient"/>
        <xs:element ref="RemoteLIName"/>
        <xs:element ref="RemoteLIInstanceNumber"/>
        <xs:element ref="MessageTransportMechanism"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ResponseStatus">
    <xs:annotation>
      <xs:documentation>ResponseStatus describes the message received status, ACK represents message
received successfully and NACK represents message received but rejected</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="ACK"/>
        <xs:enumeration value="NACK"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MessageReference">
    <xs:annotation>
      <xs:documentation>This element identifies the message with Message Type (Name) , Message Version
and with MessageIdentifier from received Message</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="MessageType"/>
        <xs:element ref="MessageTypeVersion"/>
        <xs:element ref="MessageIdentifier"/>
        <xs:element name="MessageDateTime" type="DateTime">
          <xs:annotation>
            <xs:documentation>Message Received Local Time of the Remote LI
</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="MessageType" type="FreeText">
    <xs:annotation>
      <xs:documentation>This message Type is same as the Common Schema Root Element name: Eg:
TrainCompositionMessage</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="MessageIdentifier" type="FreeText">
    <xs:annotation>
      <xs:documentation>Identification of the Message through out the lifecycle in CI. It is 36 bytes text
</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="AckIdentifier" type="FreeText">

```

```

        <xs:annotation>
            <xs:documentation>Identification of the ACK ID which is generated by prefixing "ACKID" to the message
id of the received message</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="MessageTypeVersion">
        <xs:annotation>
            <xs:documentation>This message Type Version is same as as the message Type Version from received
Message</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:maxLength value="25"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="Sender" type="CompanyCode">
        <xs:annotation>
            <xs:documentation>This element is the original sender of the message received which is company Id
</xs:documen
tation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Recipient" type="CompanyCode">
        <xs:annotation>
            <xs:documentation>This element is original Receiver of the message received which is company Id
</xs:documen
tation>
        </xs:annotation>
    </xs:element>
    <xs:simpleType name="CompanyCode">
        <xs:annotation>
            <xs:documentation>Identifies the company which is sending/receiving the messages</xs:documentation>
        </xs:annotation>
        <xs:restriction base="String4-4">
            <xs:pattern value="[0-9A-Z]{4}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="RemoteLIName" type="FreeText">
        <xs:annotation>
            <xs:documentation>This element identifies the receiving Remote LI Name</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="RemoteLIInstanceNumber" type="RLI-Number1-2">
        <xs:annotation>
            <xs:documentation>This element identifies the receiving Remote LI Instance Number</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:simpleType name="FreeText">
        <xs:annotation>
            <xs:documentation>Clear Text in ISO Unicode character set</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:maxLength value="255"/>
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="RLI-Number1-2">
        <xs:annotation>
            <xs:documentation>Identifies the type of message</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="99"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Numeric4-4">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0001"/>

```

```

        <xs:maxInclusive value="9999"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="String4-4">
      <xs:restriction base="xs:string">
        <xs:minLength value="4"/>
        <xs:maxLength value="4"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:element name="MessageTransportMechanism">
      <xs:annotation>
        <xs:documentation>This element identifies the message transport mechanism WS or
JMS</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="WEBSERVICE"/>
          <xs:enumeration value="JMS"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:simpleType name="DateTime">
      <xs:annotation>
        <xs:documentation>All DateTime are in local time plus UTC offset, time difference according the time
zones must be handled in the individual systems .</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:dateTime"/>
    </xs:simpleType>
  </xs:schema>

```

## ANNEX 5 SAMPLE ACKNOWLEDGEMENT RESPONSE MESSAGE

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:uicm="http://uic.cc.org/UICMessage">
  <soap:Body>
    <uicmr:UICMessageResponse xmlns:uicmr="http://uic.cc.org/UICMessage">
      <return>
        <LI_TechnicalAck>
          <ResponseStatus>ACK</ResponseStatus>
          <AckIdentifier>ACKID937cb438-4fb3-429a-bf27-1476818d1742</AckIdentifier>
          <MessageReference>
            <MessageType>TrainRunningInformationMessage</MessageType>
            <MessageTypeVersion>5.1.8</MessageTypeVersion>
            <MessageIdentifier>937cb438-4fb3-429a-bf27-1476818d1742</MessageIdentifier>
            <MessageDateTime>2011-12-22T12:56:20.338</MessageDateTime>
          </MessageReference>
          <Sender>3025</Sender>
          <Recipient>0074</Recipient>
          <RemoteLIName>CCG-LI</RemoteLIName>
          <RemoteLIInstanceNumber>21</RemoteLIInstanceNumber>
          <MessageTransportMechanism>WEBSERVICE</MessageTransportMechanism>
        </LI_TechnicalAck>
      </return>
    </uicmr:UICMessageResponse>
  </soap:Body>
</soap:Envelope>
```

## ANNEX 6 Heartbeat Request WSDL

```

<wsdl:definitions name="LIHBMessageService" targetNamespace="http://uic.cc.org/UICMessage"
xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://uic.cc.org/UICMessage"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <xs:schema elementFormDefault="unqualified" targetNamespace="http://uic.cc.org/UICMessage" version="1.0"
xmlns:tns="http://uic.cc.org/UICMessage" xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="UICHBMessage" type="tns:UICHBMessage"/>
      <xs:element name="UICHBMessageResponse" type="tns:UICHBMessageResponse"/>
      <xs:complexType name="UICHBMessage">
        <xs:sequence>
          <xs:element minOccurs="0" name="message" type="xs:anyType"/>
          <xs:element minOccurs="0" name="properties" type="xs:anyType"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="UICHBMessageResponse">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="return" type="xs:anyType"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="UICHBMessageResponse">
    <wsdl:part element="tns:UICHBMessageResponse" name="parameters">
  </wsdl:part>
  </wsdl:message>
  <wsdl:message name="UICHBMessage">
    <wsdl:part element="tns:UICHBMessage" name="parameters">
  </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="UICHBMessage">
    <wsdl:operation name="UICHBMessage">
      <wsdl:input message="tns:UICHBMessage" name="UICHBMessage">
  </wsdl:input>
      <wsdl:output message="tns:UICHBMessageResponse" name="UICHBMessageResponse">
  </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="LIHBMessageServiceSoapBinding" type="tns:UICHBMessage">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="UICHBMessage">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="UICHBMessage">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="UICHBMessageResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="LIHBMessageService">
    <wsdl:port binding="tns:LIHBMessageServiceSoapBinding" name="UICHBMessagePort">
      <soap:address
location="http://host:port/LIMessageProcessing/http/UICCCMessageProcessing/UICCCMessageProcessingHeartBeatWS"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```



## ANNEX 7 Sample heartbeat request message

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:uicm="http://uic.cc.org/UICMessage"
xmlns:uicmh="http://uic.cc.org/UICMessage/Header">
  <soap:Body>
    <uicm:UICHBMessage>
      <message>Are you alive?</message>
      <properties>
        <org.uic.cc.esb.message.rli.instance.no>101
      </org.uic.cc.esb.message.rli.instance.no>
        <org.uic.cc.esb.message.li.host>10.10.207.213
      </org.uic.cc.esb.message.li.host>
        <org.uic.cc.esb.message.li.instance.no>01
      </org.uic.cc.esb.message.li.instance.no>
        <org.uic.cc.esb.message.lili.transport.mode>WEBSERVICE
      </org.uic.cc.esb.message.lili.transport.mode>
        <org.uic.cc.esb.message.li.name>TESTLI
      </org.uic.cc.esb.message.li.name>
        <org.uic.cc.esb.message.rli.ip>10.10.207.64
      </org.uic.cc.esb.message.rli.ip>
        <org.uic.cc.esb.message.rli.port>8080
      </org.uic.cc.esb.message.rli.port>
        <org.uic.cc.esb.message.rli.name>TEST_REMOTE_LI
      </org.uic.cc.esb.message.rli.name>
      </properties>
    </uicm:UICHBMessage>
  </soap:Body>
</soap:Envelope>
```

## ANNEX 8 ACKNOWLEDGEMENT XML FOR HEARTBEAT REQUEST

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:uicm="http://uic.cc.org/UICMessage" xmlns:uicmh="http://uic.cc.org/UICMessage/Header">
```

```
  <soap:Body>
```

```
    <uicmr:UICHBMessageResponse xmlns:uicmr="http://uic.cc.org/UICMessage">
```

```
      <return>HEART_BEAT_WS_RECEIVED</return>
```

```
    </uicmr:UICHBMessageResponse>
```

```
  </soap:Body>
```

```
</soap:Envelope>
```