EUROPEAN UNION AGENCY FOR RAILWAYS

Making the railway system
work better for society.

INTEROPERABILITY UNIT			
TAF TSI - ANNEX D.2 : APPENDIX E - COMMON INTERFACE			
REFERENCE:	ERA-TD-104	**DOCUMENT TYPE:**	**TECHNICAL DOCUMENT**
VERSION:	2.2 (DRAFT)	**TAF TSI**	
DATE:	23.02.2017		

AMENDMENT RECORD

Version	Date	Section number	Modification/description
1.0	25.01.2011	All	Initial version
2.0	08.08.2013	All	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	All	Validated by the ERA TAF CCB on 11.09.2013
2.1	05.07.2016	All	Specification of the external interface of the CI incorporated

Important note

The present document belongs to the set of Technical Documents described in Appendix I - List of technical documents of the Commission Regulation 1305/2014 on the technical specification for interoperability relating to the telematics applications for freight subsystem of the rail system in the European Union and repealing the Regulation (EC) No 62/2006.

Related Documentation

The following list of documents are to be considered to

- TAF Master Plan v.4, issued on 17th January 2013 (Reference Implementation of Common Interface by TAF TSI Common Components Group)
- TAP TSI ANNEX B.62 (MASTER PLAN) (Reference Implementation of Common Interface by TAF TSI Common Components Group)
- ERA Technical Document TAP B 56, RU/IM Communication Application Guide (Chapter 12 Message header of RU/IM communication)

TABLE of CONTENTS

1. IMPLEMENTATION APPROACH (REF 4.2.12)

In relation to the Common Interface, the Telematics Application for Freight Services Sub System (TAF 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 TAF TSI document that the Common Interface is mandatory for each actor in order to join the TAF 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 4.2.12)

In Chapter 4.2.12 of TAF TSI “Networking & Communication” it is perscribed the requirements to fulfilled by the architecture of Common Interface and the Message exchange. :

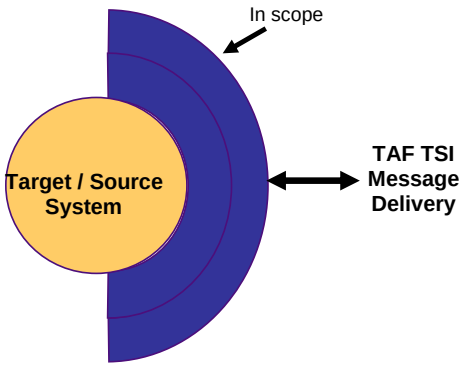
“The nature of the Information Exchange Architectures indicates a Peer-to-Peer asynchronous type of interaction between all actors, while it guarantees the overall integrity and consistency of the rail interoperability community.

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 (ref 4.2.12.6)

The context of the Common Interface is that it will provide the functionality between the TAF TSI messages and all target systems and human input exchanging data by TAF 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 TAF TSI benefits by utilising existing systems, IT platforms and communication processes whilst allowing the scalability required to implement the longer-term TAF 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.

1.2 Meeting the TSI Objective (ref 4.2.12.1 & 4.2.12.6)

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

This document describes how, using the principles of **Semantic Integration**, the service providers as quoted on section 1.3 “Technical Scope” of TAF TSI core text, Mto 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 TAF TSI message exchanges. Semantics = Data + Behavior. 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 TAF 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 TAF 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 TAF TSI.

.

To deliver value, the TAF 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 TAF TSI messages are collections of data, organised in a specific way, and grouped to provide context. To effectively use existing messages in the TAF 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 TAF TSI messages.

API Adaptors (ref 4.2.12.1 & 6)

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.

Translation and Validation (ref 4.2.12.1& 6)

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.

Metadata System (ref 4.2.14.2 & 6)

Metadata is data whose purpose is to describe other data: its definitions, structure and relationships. For the purposes of the TAF 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 TAF 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 form the Agency's website: http://www.era.europa.eu/Document-Register/Pages/Technical-Documents.aspx

Security and Transport (ref 4.2.14.3, 4, 5 & 6)

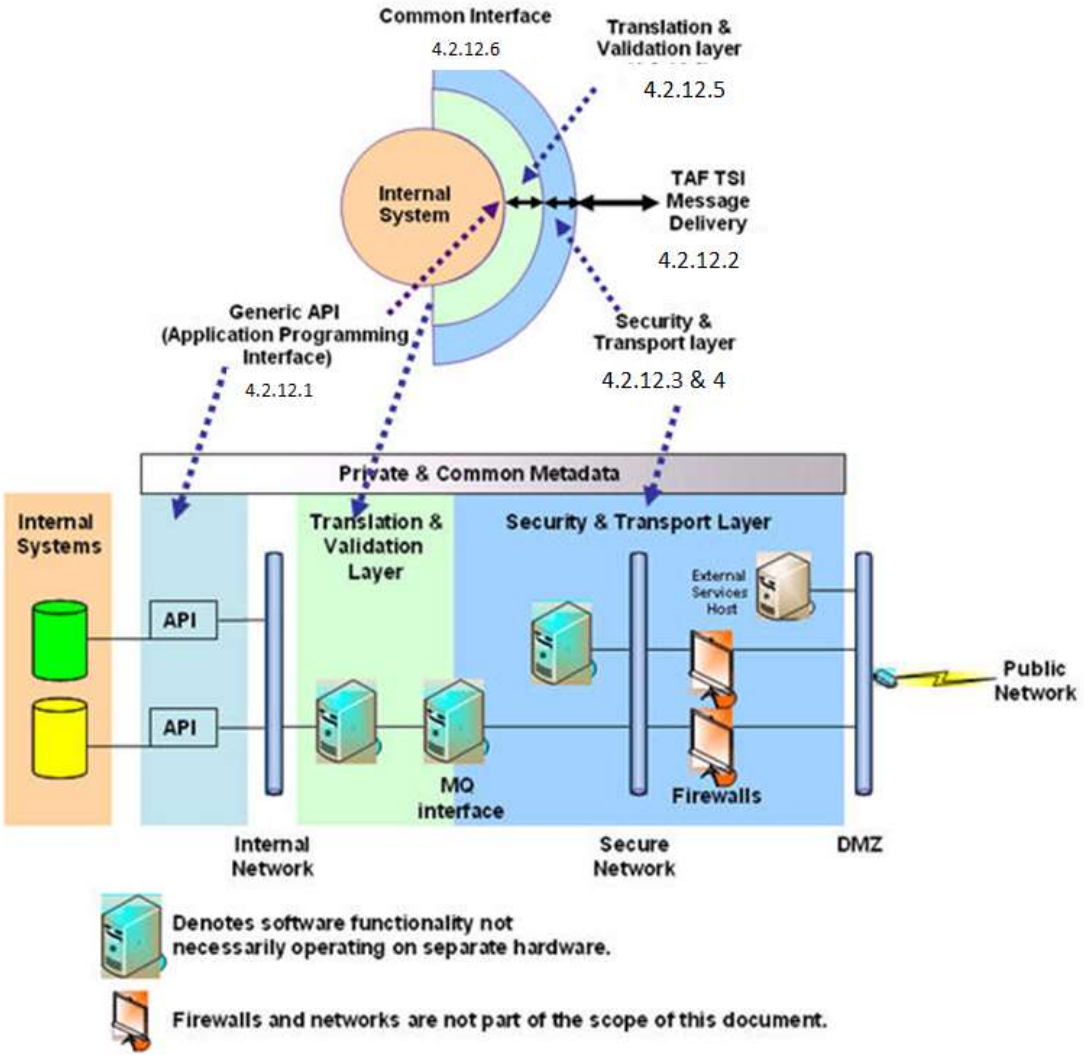
A standardised approach to the security and transportation of messages will deliver end-to-end loss-less delivery across whatever networks are available, notably the public internet.

XML(ref 4.2.12.1)

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. TAF 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 2.X. The metadata expressed in the TAF 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 TAF 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 TAF TSI XML Schemas are rich enough to describe any of non-XML data resources. The TAF 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.

1.3 Detailed Structure (ref 4.2.14.1-6)



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.

1.4 Description of the technical and operational environment

1.4.1 IT Platforms (ref 4.2.12.1)

The Reference Implementation of the Common Interface must be capable of reliable operation on open compliant IT platforms (for instance POSIX).

1.5 References

TAF TSI : Interoperability of the European rail system Technical Specification for Interoperability "Telematic Applications for Freight Services" Sub-System. Commission Regulation (EU) No 1305/2014 on the technical specification for interoperability relating to the telematics applications for freight subsystem of the rail system in the European Union and repealing the Regulation (EC) No 62/2006

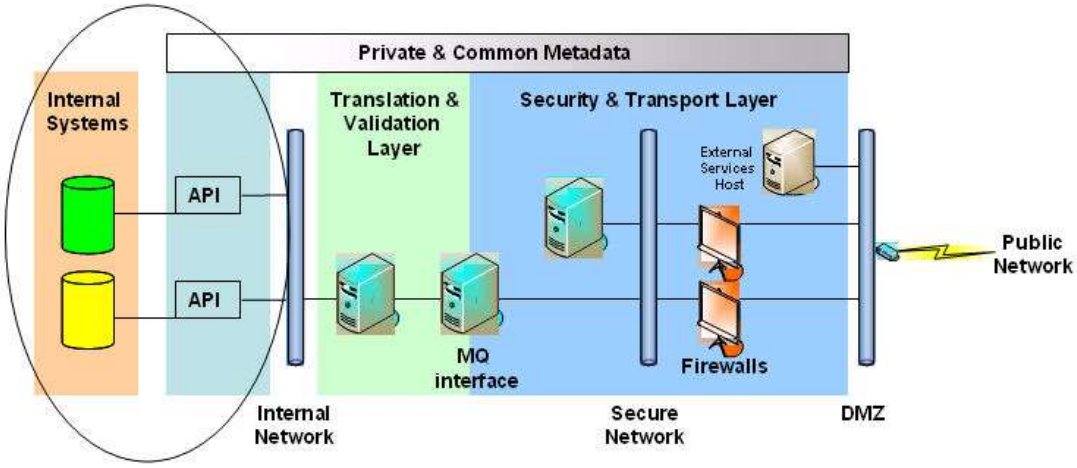
1.6 Definitions and Acronyms

See Glossary.

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.3 : To provide support for rail industry-standard programming languages
Requirement: 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 TAF TSI process, mainly RUs 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.

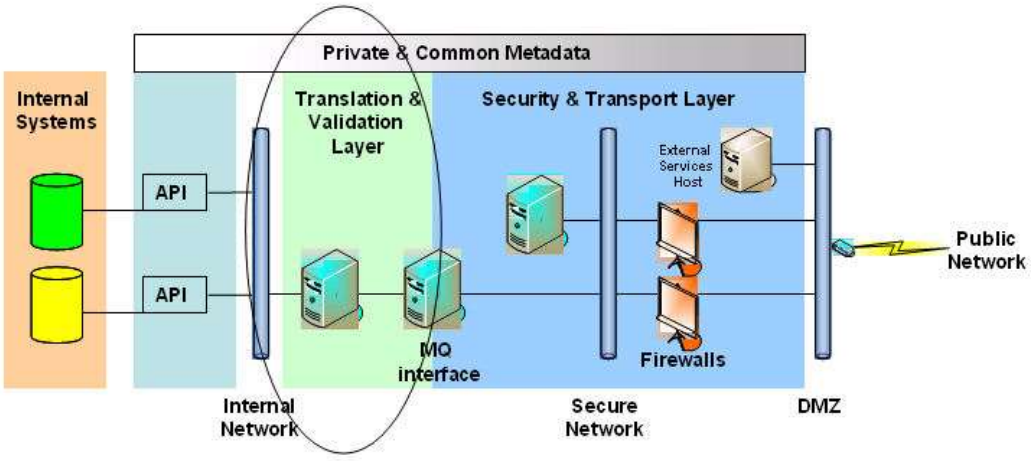
In order for the API to be successfully integrated into the target application, the following must be included:

- 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
- LDAP (for 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 TAF 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 TAF 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 TAF TSI data elements, TAF 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 TAF 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 TAF 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 TAF 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 TAF TSI metadata definitions, both in format and data quality ('annotation' and 'facets' shown in the Common Interface XSD) according to the version of TAF 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 following modules to translate existing railway messages to their nearest TAF TSI equivalent. The modules are to be selected on implementation and loaded into the Private Metadata :

Existing Message	**TAF TSI TSI message(s)**
HERMES Application 30 A	Wagon Interchange Notice Wagon ETI Wagon Interchange sub-notice
HERMES Application 30 B	Wagon Received at Interchange
HERMES Application 38-A	(enquiry & reply message to WIMO)
HERMES Application 39	Wagon Exception (possible distance data to WIMO)
HERMES messages 41	Wagon Departure Notice
HERMES messages 42	Wagon Arrival Notice
ORFEUS messages CTD, UTD,	Consignment Order
IFCSUM 97B	Consignment Order
IFTMIN 97A	Consignment Order
RCA XML (CTD)	Consignment Order
ISR IFTSTA & WSM 01	Wagon Departure Notice
ISR IFTSTA & WSM 02	Wagon Yard Arrival
ISR IFTSTA & WSM 03	Wagon Yard Departure
ISR IFTSTA & WSM 04	Wagon Interchange Notice / Sub (Border crossing)
ISR IFTSTA & WSM 05	Wagon Arrival Notice
ISR IFTSTA & WSM 06	Wagon Exception
UIC 407-1 2001	Train Running Forecast
UIC 407-1 2002	Train Running Information Train Running Interruption
UIC 407-1 2090 Europtirails	Path Request Path Details
UIC 407-1 2003	Train Delay
UIC 407-1 Generic 2004 /2201 & Europtirails 2004	Train Composition
UIC 407-1 2701	Train Running Interruption

It All these TAF messages and elements are defined in TAF TSI APPENDIX F — Data and Message Model, version 2.X.

The public metadata must hold the TAF-TSI XML Schema shown in the Common Interface XSD, allowing internal systems to process correctly formatted TAF TSI messages into and out of the Queues without Translation.

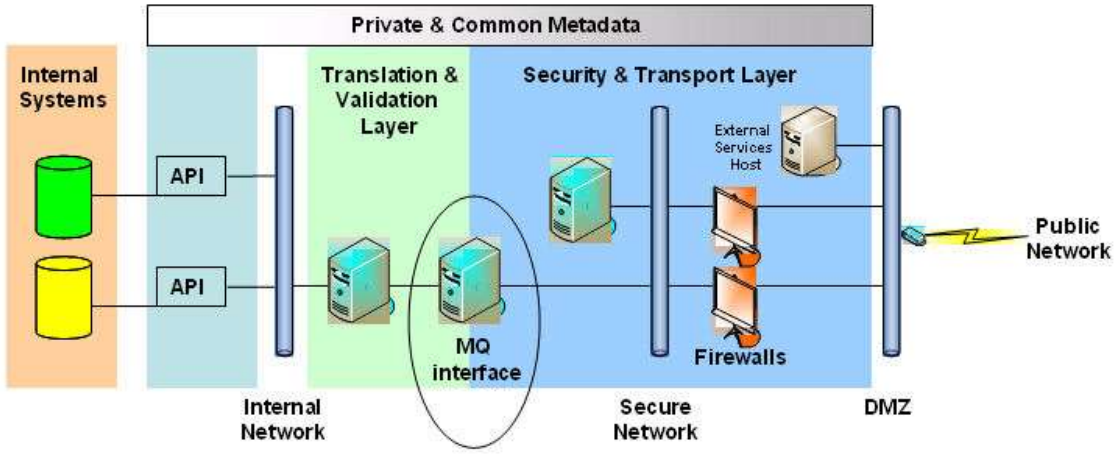
5. The translation & validation layer must handle the following validation :
    - Compliance checking of received messages from the network and logging and sending of error reports to both parties when compliance

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 (ref 4.2.12.1, 2 & 6)

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.



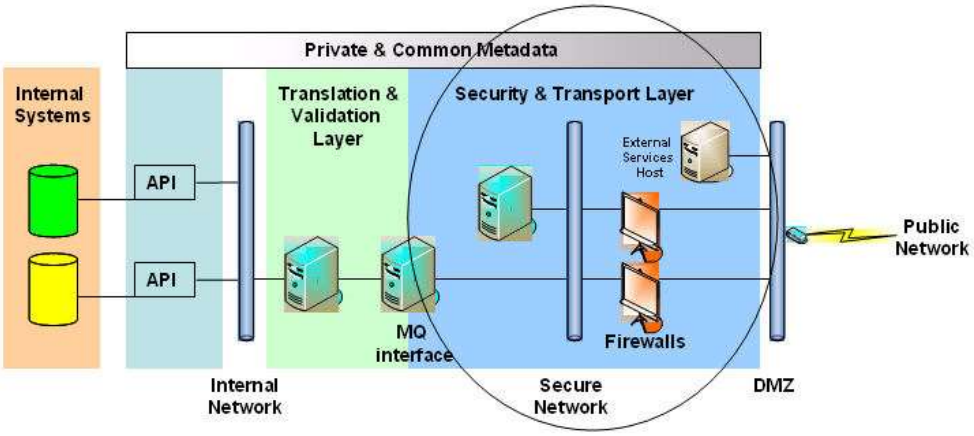
This objective will be met by the implementation of an Open Message Queue as shown. By using an open Message Queuing interface between the Translation & Validation layer and the Security & transport layer, it is possible for the following functional requirements to be met:

- 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 (ref 4.2.12.2, 3, 4 & 6)

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 TAF 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 TAF TSI messages from other Common Interface implementations elsewhere in the Rail Industry either by the TAF 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 TAF TSI messages to other Common Interface implementations elsewhere in the Rail Industry by transferring the TAF 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 TAF 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 TAF 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 TAF 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 RNE **RailNetEurope (RNE) Common Components Manager (Email contact:** support.ccs@rne.eu) Some of the data in the TAF TSI reference files will be part of the public metadata (see TAF TSI – ANNEX D.2 : APPENDIX C – REFERENCE FILES, version 2.0) and each will have its own Human-Machine Interface for administration purposes. These Human-Machine Interfaces will be separate to the Common Interface.

2.6 Metadata (ref 4.2.12.2 & 5)

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 TAF TSI schemas).

2.6.1 Private Metadata (ref 4.2.12.2 & 5)

- Specific Implementation Metadata about APIs in use,
- Translation tables from internal system data to and from TAF 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 (ref 4.2.12.2 & 5)

API Metadata (ref 4.2.12.1 & 6)

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

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

- TAF TSI Message Metadata (XSD) - ERA-TD-105: TAF TSI - Annex D.2 : Appendix F - TAF 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 (ref 4.2.12.1, 2 & 6)

The addressing of partners should be defined in the URL and the operation according chapter 4 to 6. Host names /IP Address 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 (ref 4.2.12.2 & 5 and 4.4.2)

The common metadata required for operation of the Common Interface is published on the Agency's website: http://www.era.europa.eu/Document-Register/Pages/Technical-Documents.aspx . 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, including the use of the metadata for non-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:

The following administrative functionality is required 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. PERFORMANCE AND DATA QUALITY

3.1 Sizing and performance (ref 4.4.1)

Capacity

A single instance of a Common Interface should be capable of communicating with up to 1000 other Common Interface instances.

A single instance of a Common Interface should be capable of communicating simultaneously with up to 30 “existing applications” on API layer.

Performance

A Common Interface instance should be capable of sending/receiving (together):

- nominal stress: a sustained rate of up to 100 TSI TAF messages/database accesses (in a random mix) per second;
- peak stress: a 1 minute peak of up to 200 TSI TAF messages/database accesses (in a random mix) per second.
-

The delay of any message passing through the chain Common Interface API — Common Interface external interface should be less than 2000 ms, and 90% should be within 500 ms (assuming an infinitely fast internet and a nominal stress).

Minimum unplanned unAvailability, MTBF, MTTR

A Common Interface instance should be capable of running continuously. Availability of a Common Interface should be designed to deliver at least 99.9% measured on a monthly basis (maximum total outage 525 minutes/year).

Maximum number of outages per year is 50 (MTBF=1 week)

Automatic recovery of software-related errors shall take place within 30 minutes (MTTR).

Minimum planned unAvailability, MTBF, MTTR

Automated update the public metadata within 10 seconds.

Internet resource utilisation

Under nominal stress, the communication between two Common Interfaces should not cause more than 600 kb/s (thousand bits per second) load per direction on the internet. (30 messages/s x 1000 B x 8b/B x 2 overhead factor = 480kb/s). Strongly recommend compression.

Computer resource utilisation

A Common Interface communicating with 5 existing applications on API layer and 10 other Common Interfaces under nominal stress, should not cause a load on the cpu or any other critical resource of the computer the Common Interface is running on greater than 50%.

3.2 Data Quality (ref 4.4.1)

3.2.1 Prerequisite

Chapter 4.4.1 of the Telematics Application for Freight Services Sub System (TAF 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 the 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:

3.2.2 Level 1 Compliance Checking

As the TAF 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.

3.2.3 Level 2 Application Validation

According to the TAF 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 Ident contains 4 integers, 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.

3.2.4 TAF Acknowledgments (ref 4.2.12.6 & 4.4.1) -

Receipt Confirmation Message (ref 4.2.12.6 & 4.4.1)

The TAF TSI states in the Common Interface section that “based on the results of authenticity verification of incoming messages, a minimum level of message acknowledgement can be implemented.” Message acknowledgement can be positive or negative. These messages have the ability to communicate either 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 TAF TSI catalogue. The messages are defined as follows.

ReceiptConfirmationMessage

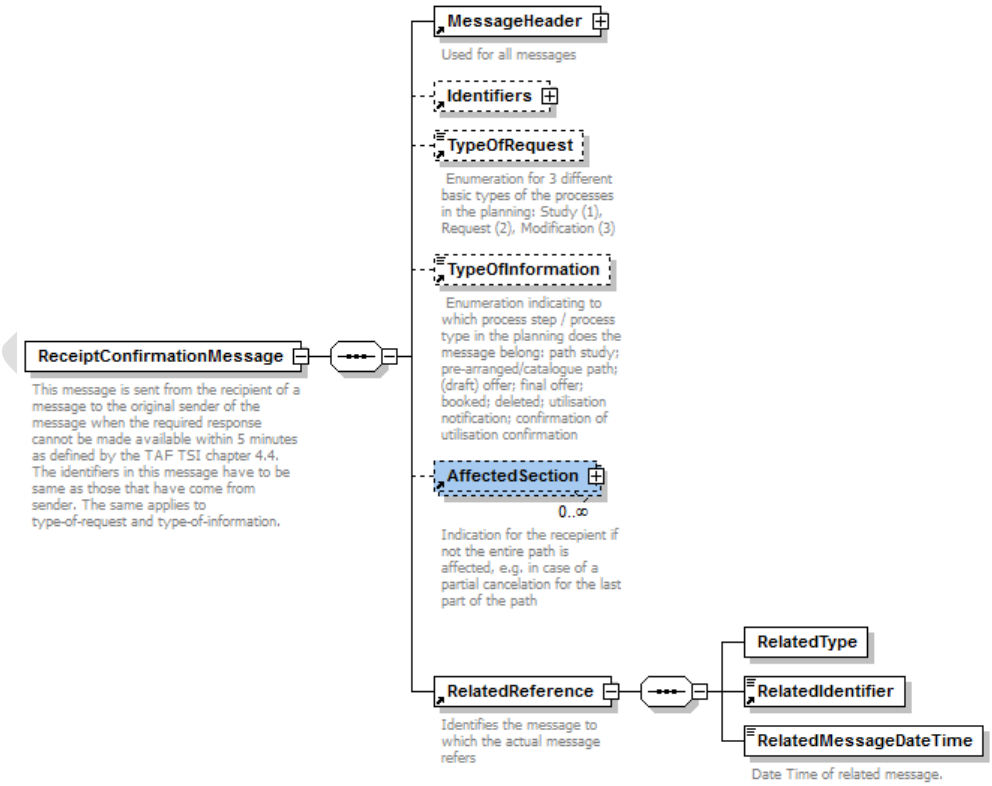


Diagram 3.1 – ReceiptconfirmationMessage

This message is sent from the recipient of a message to the original sender of the message when the required response cannot be made available within 5 minutes as defined by the TAF TSI chapter 4.4. The identifiers in this message have to be

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

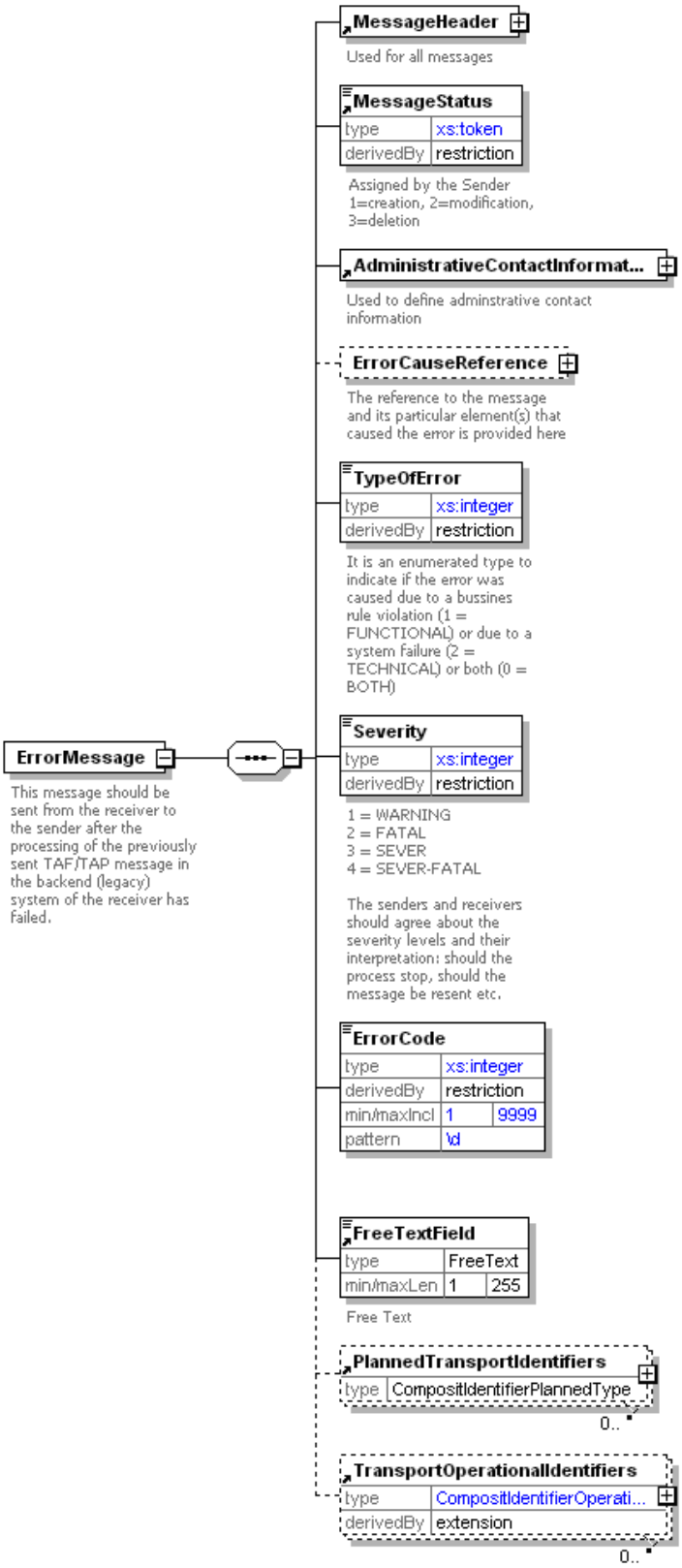


Diagram 3.2 – ErrorMessage

Elements TypeOfError, ErrorCauseReference and Severity define precisely what was the cause of the error and which consequences does it have for the message exchange. The codes for the ErrorCode element are a matter of multilateral or bilateral agreement

between the partners.

There are two levels of message checking and validation as mentioned in 3.2.2 and 3.2.3.

3.2.5 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.

3.2.6 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.

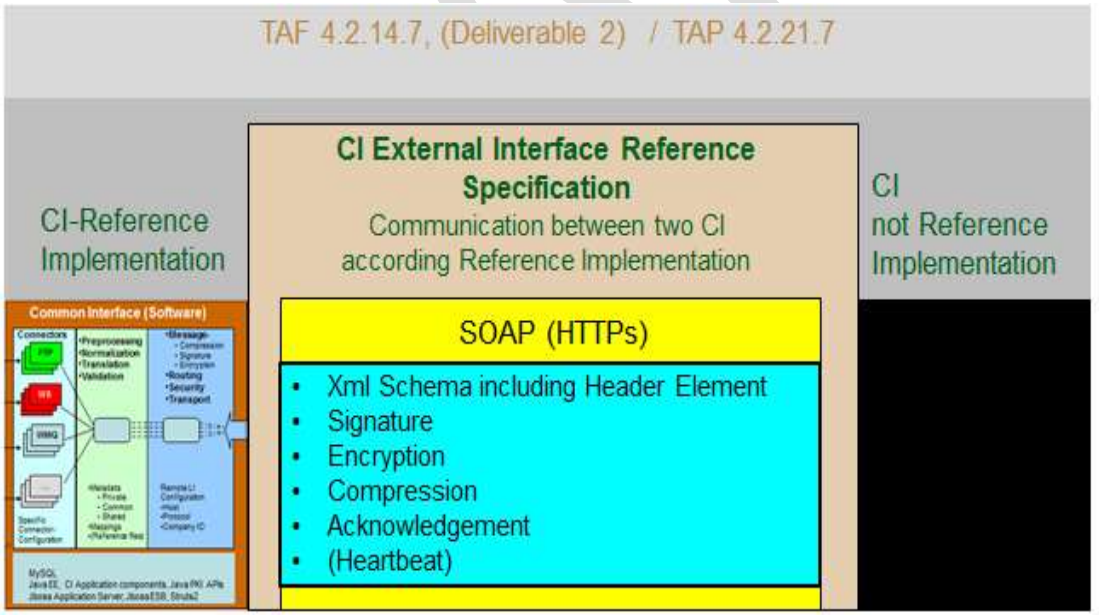
The messages Receipt Confirmation and Error Message are used to fulfil the Level 2. The Level 1 shall be fulfilled automatically by Common Interface Translation and Validation Layer, to prevent the syntactically incorrect messages already at the sending time, before such a message would have been sent 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 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. The message to be sent should be enclosed in a SOAP request envelope and the response xml will also be enclosed in a SOAP response envelope.

As in TAF/TAP 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 TAF/TAP 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/TAPTSI message. If the message is compressed, encrypted or singed, 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 from RNE. 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 SSL truststore including the senderAllias. 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 acknowledgement 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.
- AckIndentifier: 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

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. 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: **For SSL communication, you have to obtain a certificate from RailNetEurope (RNE) Common Components Manager (Email contact:** support.ccs@rne.eu) that serves as the certification authority**.**

5.5 Message Compression

The TAF-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 uses ZLIB compression libraries for message compression and decompression. ZLIB is a popular compression/decompression and is not protected by any patents. For detailed specification of ZLIB refer http://www.ietf.org/rfc/rfc1950.txt. This compression results in a byte stream. The byte stream is converted to String using Base64 encoding.

5.6 Message Encryption

The TAF-TSI message payload in the request can be encrypted for security. The encryption is optional and if it is encrypted then corresponding Header property should be set to true (refer to section 5.2.1)

The reference implementation for the CI uses Password based encryption (PBE) with DES and MD5. This algorithm follows the PKCS (Password-based encryption standard). For more details on PKCS refer http://tools.ietf.org/html/rfc2898.

In this encryption method the message is encrypted and decrypted using a using a secret key. In the reference implementation for the CI the key is derived from the SSL certificate. Therefore, for encryption you have to obtain a valid certificate from **RailNetEurope (RNE) Common Components Manager (Email contact:** support.ccs@rne.eu) that serves as the certification authority. It is the same as used for SSL on transport layer.

5.7 Message Signing

The TAF-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 SSL certificate and verified using the public key. The signing algorithm used is MD5withRSA.

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

6. 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 happen 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

6.1 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.

6.1.1 SOAP Header

The SOAP Header do not contain specific properties.

6.1.2 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

6.1.3 Sample Heart Beat Request SOAP Message

An example is annexed as Annex 7

6.2 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

6.3 Invocation

6.3.1 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 SSL communication, the certificate has to be provided from the same certification authority. The certification authority is RailNetEurope (RNE) Common Components Manager (Email contact: support.ccs@rne.eu). The same certificate as for messaging must be used.**

7. CHANGE MANAGEMENT

All change management will be in accordance with Chapter 7.2 of the TAF – TSI ("Management of Change").

Any user may propose a modification to the reference file. Basically, these change requests (CR) may be due to enhancements or corrections of the system. Enhancements are meant to add value to the system and corrections are to repair errors. Serious errors are to receive priority and be fixed immediately. Minor errors (i.e. documentation, screens, etc.) are to be processed as CRs.

CRs are logged in a dedicated change management register managed by the European Union Agency for Railways (ERA). Each CR should contain the proposed change, the reason, the urgency / importance and the originator. The ERA Core Team in charge approves or rejects CRs based on a change management procedure ("ERA_Telematics_CCM_Guide" - Telematic applications change control management"). Approved CRs will be included in the CR-plan with a description of the type of the change. For further information see document "ERA_Telematics_CCM_Guide_ - Telematic applications change control management").

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 TAF-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:xs="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="xs:anyType"/>
                    <xs:element minOccurs="0" name="signature" type="xs:anyType"/>
                    <xs:element minOccurs="0" name="senderAlias" type="xs:anyType"/>
                    <xs:element minOccurs="0" name="encoding" type="xs:anyType"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="UICMessageResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="xs: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 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

        MQ9ABJzZqttYdc65vUNCuDedwMR33RnZYYWFUaBqB+UHrDksMrVgcwB8J1gw9J1ulhWeucp4SsL0
                        V+VveADK9hAFOpZDDI05PKmKqtdTL/vfkEaJfy8cI2CZQcS3j53d6jd+GSC0RMibtkH9iexrgjQ5
                        tSZblAmyimCgaiNweCMiBEJqOVCnIq09jBMP5nR6bE074CKZ5RxeOjFNqizWDYyt//Z8Z8OnUku1
                        Pk8ixC8tWWo6DtRdCP0O0uyauzP5S8HVvl63IK7hip5cGIRD0vkKlx5ySu4e6EmrBZA1E3FavXGj
                        +JKMzY/LLxjJfc0hjFN44Ce2Mw8v8dUouoX9SRUvLIwdvbvLfCWvBXE1AJmwwSBZMFH0/axC8vcp
                        b+8jx+lZwTaST5sFfQyOHr04tzCUZimouEU0TBEMTfzDWrksQrS+3M26i5t/Vc2k9f7he3CEomZL
                        cvvzlg9B+bjePEqimm9jfGXR8UDW7h5llotcfMNHb1OfR4BPbgVOxDPE6cLPmSUklEBAx1wgj026
                        ETI7382s2L5RbPY5tu74N+DAWVqsl+qD3ImaqlW/UjtSgs9+9JL4kZEJiSsfV7OKoKvC5nZ21SwG
                        3zMLbqsEZN4ulnncJstx57tjby3IDSSeg7QeJYGchFWUvw2sv3xGrhgF5HWQkBm+Li4g1xEmrp0L
                        ZXf8oA3jtfF+bLeo0ecEtDkozg5JaqN7TfCbo/o7Z8/3A4OuzibIlwmOrDPIlxSF5wofDCQO6RsI
                B4bLHQFc5iRHTv1TNCwm7biDzcc4dLIH6M+Px38MdIOzCcu7D+8kkVWc4NXjNaztqMu2IMJD3GZ5
                        XCvCts1XKmnR6puLS3wyTUoImj2zHlFBv6dCvGpbIcD4/tD79Y/mhCqrsPloNF33EGvsvuEnHijN
                        NTYdxRSKKp8=
                </message>
        <signature>vu3MQA/Oi8M3GqGodD+FApM2Oq4j2UVK3z2jvvS95JaxgOopDUr7d/EL83/92mhTATG/szvZcoFu
                        RNEec6xZNQGzp1ISnb7XY7AjLT7k2S0AuCRkgJoHZ6hJtDF7eMo/Xwy+81pe/dFHr2TZNJGRMp0t
                        tgAxhmDdLX4rIjIC2Pg=</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-
3bd73b29ddfa
                </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 />
                <TrainNumber>
                        <PathIdent>48119</PathIdent>
                        <ScheduledDepartureDateTime>2011-04-11T20:05:00+02:00
                        </ScheduledDepartureDateTime>
                        <PathDepartureLocation>
                                <CountryCodeISO />
                                <LocationPrimaryCode />
                                <PrimaryLocationName />
                                <LocationSubsidiaryCode
                                        LocationSubsidiaryTypeCode="" />
                        </PathDepartureLocation>
                </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>
                <LocoIdent>
                        <TractionType />
                        <TractionIdent />
                        <TractionMode />
                </LocoIdent>
                <TrainRunningData>
                        <TrainRunningTechData>
                                <TrainWeight>461900</TrainWeight>
                                <TrainLength>452</TrainLength>
                                <LocoIdent>
                                        <TractionType />
                                        <TractionIdent />
                                        <TractionMode />
                                </LocoIdent>
                                <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>
                                <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="AckIndentifier"/>
				<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="AckIndentifier" 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:documentation>
            </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:documentation>
            </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="Numeric4-4">
                <xs:minInclusive value="0001"/>
                <xs:maxInclusive value="9999"/>
            </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: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>
                    <AckIndentifier>ACKID937cb438-4fb3-429a-bf27-1476818d1742</AckIndentifier>
                    <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 WSD

```
<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="https://SLW1095/LIServices/LIHBMessage"/>
            </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>
```

1.1 Acronyms and Abbreviations

The following list provides the acronyms and abbreviations used in the technical document.

Acronym / Abbreviation	Definitions
CI	Common Interface also LI Local Instance
CSV	Comma separated values
DES	Data Encryption Standard
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
MD5	Message-Digest Algorithm, cryptographic hash function
MD5withRSA	Refers to a signature algorithm where hashing is performed using MD5 and encryption/decryption is performed using RSA.
MQ	Message Queuing
NACK	Negative Acknowledge
PKI	Public Key Infrastructure
PBE	Password Based Encryption
RA	Registration Authority
RSA	public-key cryptosystem
RNE	RailNetEurope
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
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