Horizon 2020
European Union Funding
for Research & Innovation

In2Rail

In2Rail

Project Title:	**INNOVATIVE INTELLIGENT RAIL**
Starting date:	01/05/2015
Duration in months:	36
Call (part) identifier:	H2020-MG-2014
Grant agreement no:	635900

Deliverable D8.4
Interface Control Document for Integration Layer Interfaces, external/Web interfaces and Dynamic Demand Service

Due date of deliverable	Month 36
Actual submission date	16-04-2018
Organization name of lead contractor for this deliverable	HC
Dissemination level	PU
Revision	Final

Authors

		Details of contribution
Author(s)	**HaCon Ing. (HC)** Sandra Kempf Rolf Gooßmann	Coordination and Document structure of D8.4 Contributions to Chapter 1-12 Author of Chapters 9.1, 9.2, 9.4 and 9.5
Contributor(s)	**Siemens (SIE)** Stefan Wegele	Author of Chapters 6, 8 and 10.2 Contributions to Chapter 1-12
	Bombardier Transportation (BT) Roland Kuhn Martin Karlsson Zbiewniew Dyksy	Author of Chapters 9.3 Contributions to Chapter 1-12
	CAF Signalling (CAF) Carlos Sicre Vara de Rey Manuel Castro Viñas	Contributions to Chapters 1-12
	Ansaldo STS (ASTS) Gian Luigi Zanella Matteo Pinasco	Author of Chapter 7 Contributions to Chapter 1-12
	AZD Praha s.r.o. (AZD) Martin Bojda Michal Žemlička Martin Růžička	Contributions to Chapter 1-12
	Thales (THA) Christoph Bücker Vishal Bhatt	Contributions to Chapter 1-12

Executive Summary

The overall aim of the In2Rail project is to set the foundation for a resilient, cost-efficient, high capacity, and digitalised European Rail Network.

Intelligent Mobility Management (I2M), a sub-project of I2R, is one of the three technical sub-projects and comprising Work Package 8 (WP8). WP8 addresses and develops a standardised integrated ICT environment capable of supporting diverse TMS dispatching services and operational systems. It also includes standard interfaces to external systems outside TMS/dispatching (for other railway management systems and transport modes) with a plug-and-play framework for TMS/dispatching applications.

WP8 represents the part of I^2R lighthouse project to Shift2Rail IP2 and CCA which addresses works which are key inputs to S2R TD2.9 "Evolution of Traffic Management System" and CCA WA4.2 Integrated Mobility. All deliverables from WP8 will form the base for proceeding works in X2RAIL-2. WP6 "Traffic Management System" (IP2) and IMPACT-2 WP7 "Integrated Mobility" (CCA).

The current document corresponds to the last two deliverables inside WP8, and is focused in the description of the required Data structure for the external and internal Interfaces of the Integration Layer.

The research has been conducted by all partners of WP8, and the inputs have been consolidated in this document.

TABLE OF CONTENTS

Abbreviations and acronyms

Abbreviation / Acronyms	Description
AF	Application Framework
API	Application Programming Interface
ATO	Automated Train Operation
CCS	Control & Command System (Signalling)
CDM	Canonical Data Model
ETA	Estimated Time of Arrival
ETCS	European Train Control System
ICT	Information and communications technology
ICD	Interface Control Document
IF	Interface
IL	Integration Layer
ILS	Interlocking
IM	Infrastructure Manager
IMDG	In-Memory Data Grid
LX	Level Crossing
MA	Movement Authority
OffPP	Offline Production Plan
OnPP	Online Production Plan
P2P	Point-to-Point
QoS	Quality of Service
RBC	Radio Block Centre
RTTP	Real Time Traffic Plan
S2R	Shift2Rail
SiB	Signal Board
SQL	Structured Query Language
S&P	Subscribe and Publish
TAF TSI	Telematics applications for freight service
TAP TSI	Telematics applications for passenger service
TMS	Traffic Management System
TRL	Technical Readiness Level
TSR	Temporary Speed Restriction
UML	Unified Modelling Language
WP	Work Package
XML	Extensible Markup Language

1 Background

The products and systems for Traffic Management available on the market from the various supply sources usually do not use standardized data structures and interfaces for communicating within a TMS or to external services. Standardization in this area can only be observed for certain interfaces with external services such as, e.g., TAF/TAP TSI or UIC 407. This leads to enormous one-time efforts and cost to link sub-systems and products of different suppliers.

Cost savings linked to the reduction of these non-recurrent cost are considered to reach up to 10% of the total project cost if combinations of sub-systems would use a standardized ICT structure are applied within the overall system.

Therefore, the design of a communication platform with standardised interfaces to connect the TMS with internal and external services or systems is a key target for In2Rail and the preceding S2R activities.

In the frame of specifying and developing a new integrated and standardized ICT structure for rail operation services the standardization this deliverable is the first step towards the required data structure and message definition syntax for the interfaces between the TMS and internal and external systems.

2 Objective / Aim

WP8 constitutes one of the issues in the framework of the Project titled “Innovative Intelligent Rail” (Project Acronym: In2Rail; Grant Agreement No 635900).

The overall objective of WP8 is to address and develop a standardised integrated ICT environment capable of supporting diverse TMS dispatching services and operational systems. Additionally, WP8 deals with standard interfaces to external systems outside TMS/dispatching and with a plug-and-play framework for TMS/Dispatching applications.

The WP8 includes two areas; the Integration Layer (IL) and the Application Framework (AF) for applications. Each area is devoted to specific subtopics, which are shown in Figure 2.1.

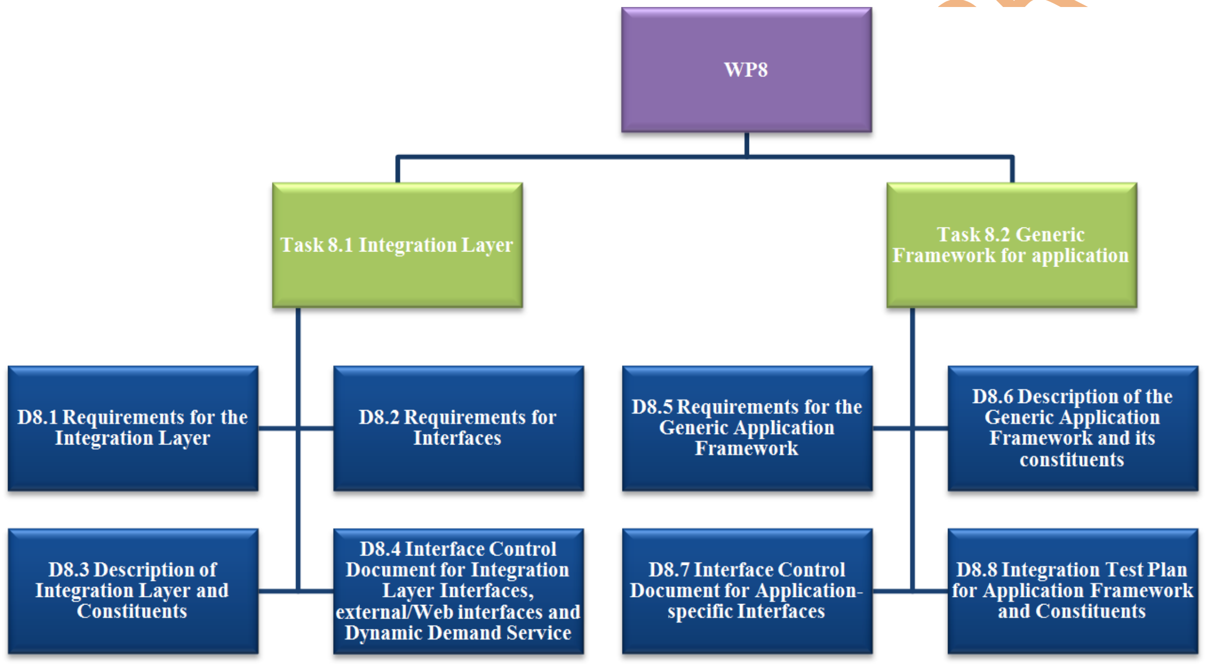

WP8
Task 8.1 Integration Layer
Task 8.2 Generic Framework for application
D8.1 Requirements for the Integration Layer
D8.2 Requirements for Interfaces
D8.3 Description of Integration Layer and Constituents
D8.4 Interface Control Document for Integration Layer Interfaces, external/Web interfaces and Dynamic Demand Service
D8.5 Requirements for the Generic Application Framework
D8.6 Description of the Generic Application Framework and its constituents
D8.7 Interface Control Document for Application-specific Interfaces
D8.8 Integration Test Plan for Application Framework and Constituents

Figure 2.1: Subtopics of WP8

The long term objective for the project is to provide a standardised integrated ICT environment supporting TMS applications connected to other multimodal operational systems (see Figure 2.2).

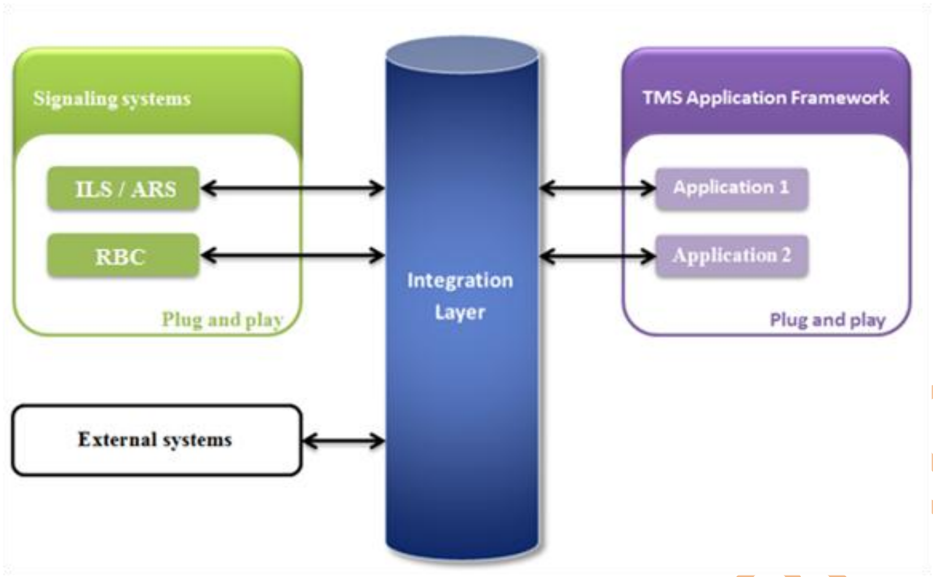


Figure 2.2: Overview of Integration Layer as communication platform

The objective of this Interface Control Document (ICD) is to describe the both-way interfacing between the Integration Layer and external and internal systems and hence avoiding complex and costly function and data mapping processes within the Interface structure.

Deliverable 8.4 in combination with [D8.7] is the first step towards standardisation Interfaces in the traffic management and will be followed from proceeding activities in X2RAIL-2, WP6 and IMPACT-2, WP7 project of Shift2Rail including development of prototypes up to TRL6.

3 Purpose and structure of the document

The aim of this document is to provide a formal description of the required Data structures for the external and internal Interfaces of the Integration Layer.

This document together with [D8.3] shall enable fulfilment of the requirements specified in [D8.1]. The deliverable [D8.3] provides functional description of the Integration Layer with reasoning for architectural decisions. The underlying data model definition, also called the "Canonical Data Model" (CDM), can be found in [CDM].

This document is structured as follows:

- Chapter 6 describes the Integration Layer API; how systems/modules can interact with the IL;
- Chapter 7 contains a description of the security that should be used for the IL;
- Chapter 8 gives an overview about the sandbox and versioning of the Integration Layer;
- Chapter 9 focuses on the different Topics used by the Integration Layer for communication;
- Chapter 10 consists of the conclusion of the document.

4 Integration Layer API

The Integration Layer has two main functionalities:

- reliable management of object states with no single point of failure;
- allow modification of objects by the clients and notify subscribed clients about any objects modifications.

There are several products and projects on the market [ZooKeeper] [Hazelcast] [Redis] [OpenSplice][ConnexDDS]. Most of them use specific API – either precompiled library with interface or using generated source code.

To allow using any suitable product from the market and avoid a vendor lock-in one generic API shall be created, covering only functionality, required by Integration Layer. The API shall be provided as re-usable library or a set of. In this case services connected to the Integration Layer can be created and offered on the market as products, to be used without adaptation with any compatible implementation of IL.

The proposed API is relatively simple and consists of the classes listed in Figure 4.1. The term IL_ is used as a prefix.

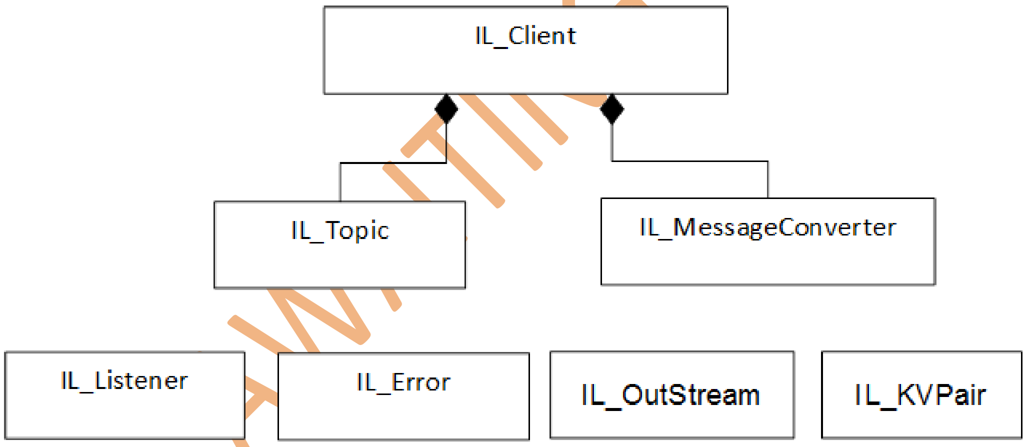


Figure 4.1: Class diagramm of API

The process of subscription is shown in Figure 4.2.

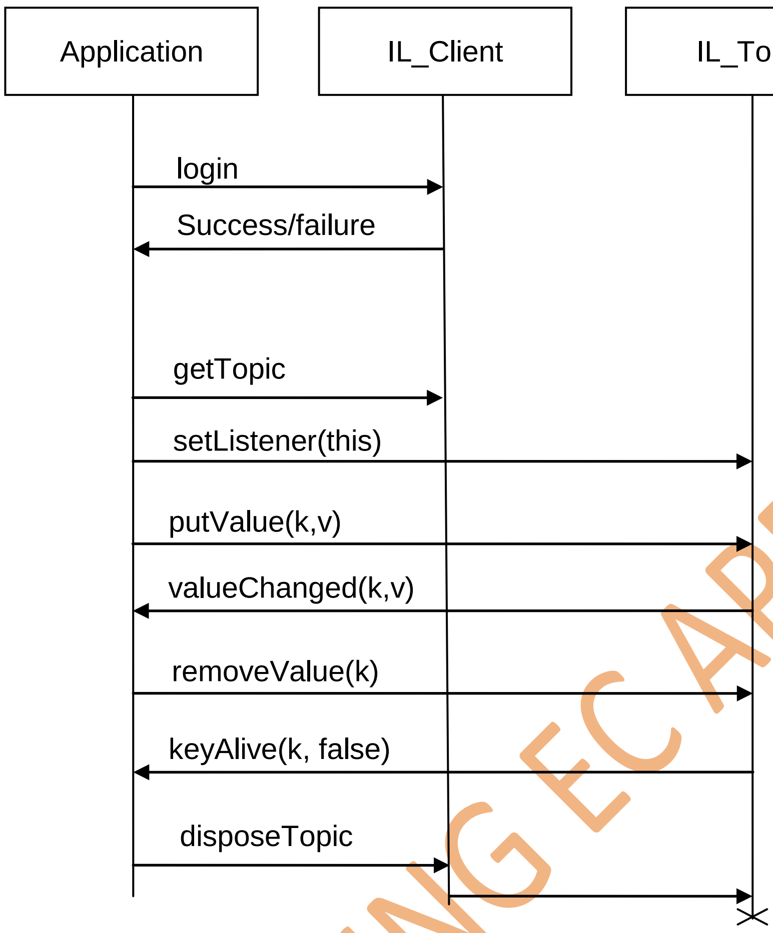


Figure 4.2: Sequence diagram for communication with API

4.1 IL_Client

The main responsibility of `IL_Client` class is to setup access to Integration Layer in method login and to be a container for the other classes of the API.

```
typedef enum {   MSG_TYPE_RAW,
                  MSG_TYPE_PROTOBUF,
                  MSG_TYPE_JSON,
                  MSG_TYPE_XML,
                  MSG_TYPE_ASN1} IL_EncodingFormat;


class IL_Client
{
public:
  virtual bool login(const char *jsonConfig, IL_Error & error) = 0;
  virtual IL_Topic* getTopic(const char *topicId,
                       const char* topicConfig, IL_Error & error) = 0;
  virtual void dispose(IL_Topic* topic) = 0;
  virtual bool registerDataType(const char *cdmModuleName, const char
*cdmModuleContent, IL_Error & error) = 0;
  virtual bool getMessageSchema(const char *cdmModuleName,
IL_EncodingFormat fmt, IL_OutStream & os, IL_Error & error) = 0;
  virtual IL_MessageConverter* messageConverter(IL_EncodingFormat fmt) = 0;
};
```

4.1.1 Login to Integration Layer

bool IL_Client::login(const char *jsonConfig, IL_Error & error);

Description

Authorises the client to Integration Layer to start subscriptions and data modifications.

Parameters

Parameter	Type	IN/Out	Description
jsonConfig	const char * (zero terminated string)	In	Contains a data structure with login parameters and product specific information (like IP addresses of the cluster nodes).
error	IL_Error &	Out	Provides an error code and the error message in case of failure.

Return value

Value	Description
True	Login successful.
False	Login unsuccessful, error results are in error attribute.

4.1.2 Get topic to receive and send data

```
IL_Topic* getTopic(const char *topicId, const char* topicConfig, IL_Error &
error);
```

Description

Request of a topic from Integration Layer.

Parameters

Parameter	Type	IN/Out	Description
topicId	const char * (zero terminated string)	In	Id of the topic
topicConfig	const char*	In	Json representation of the data structure TopicConfig shown in the table below.
error	IL_Error &	Out	Returns the error code and the error message in case of failed request.

Return value

Pointer to IL_Topic in case of success.

Nil-pointer in case of failure.

Data structure defining `topicConfig` parameter of `IL_Client::getTopic`.

```
<struct name="TopicConfig">
```

```
      <attr name="id" type="string" key="true" id="1"/>
      <attr name="dataType" type="string" id="2"/>
      <attr name="qos" type="QoS" containment="true" id="3"/>
      <attr name="modelAddressPrefix" type="string" id="4"/>
</struct>
```

Attribute	Description
id	Topic id. To support the OMG standard DDS, the id is allowed to contain only ASCII alpha-numeric values: [0-9][a-zA-Z].
dataType	Represents the data type in module.message notation, defined in a proto-File.
qos	Quality of services is a structure described below. It is influenced by OMG DDS specification, version 1.4.
modelAddressPrefix	According to the canonical data model, the data objects managed in Topics build an object tree. Any object of the tree can be addressed starting from the root node. The address of each object in the topic is represented as modelAddressPrefix/key which allows the clients to find any object in any topic by its address.

The data structure Quality of Service (QoS) allows configuration the topic behaviour.

```
<struct name="QoS">
  <attr name="reliableTransport" type="boolean" default="true" attrId ="1"/>
  <attr name="durability" type="Durability" default="VOLATILE" attrId ="2"/>
   <attr name="latencyBudget" type="uint32" attrId="3" default="0"/>
   <attr name="transportPriority" type="int32" attrId ="4" default="0"/>
   <attr name="lifespan" type="uint32" default="0" attrId ="5"/>
</struct>
```

Attribute	Description
reliableTransport	There are two kinds of data transport protocols “Reliable” and “best effort”. If the value is “true”, the transport protocol contains acknowledge and resend functionality ensuring, that each subscriber receives a message. With “best effort” some messages can be dropped with the advantage of increased performance. A typical use case for “best effort” could be a Voice-over-IP application.
durability	This is a key attribute defining Topic behaviour. Single values are described below.
latencyBudget	The attribute is defined in OMG DDS. It specifies the maximum acceptable delay in milliseconds from the time the data is written until the listener is notified about it. It is a hint for the Integration Layer, which is used to combine several messages into one bulk often increasing transfer performance by an order of magnitude.
livespan	Defines the maximum duration of validity of the data

Attribute	Description
	after it was written by the writer. The default value is 0, which means “infinite”. As any object contains a timestamp from the sender side, a reader can use this attribute to decide, if the object contains valid data. Integration Layer uses this attribute to find and remove outdated data objects by a garbage collector service.
transportPriority	The attribute provides a hint to the Integration Layer how to set the priority of the underlying transport.

The durability attribute of QoS is the main field specifying behaviour of the topic. It describes, if the data should outlive their writing time.

```
<enum name="Durability">
      <enumerator name="VOLATILE" value="0"/>
      <enumerator name="TRANSIENT_SESSION" value="1"/>
      <enumerator name="TRANSIENT" value="2"/>
      <enumerator name="PERSISTENT" value="3"/>
</enum>
```

Attibute	Description
VOLATILE	Means that the data can be removed from Integration Layer as soon as all existing subscribers received it. The typical use case is notification about some operations. Using this attribute the Integration Layer provides a typical publish-subscribe communication topic for message based communication.
TRANSIENT_SESSION	The data will have the same live-time as the writer of the data – as long as the writer is connected to the Integration Layer the objects are provided to late-joiners. This attribute annotates, which data shall be managed by a session management service and removed as soon as the writer disconnects from IL.
TRANSIENT	The data will be managed in memory of the Integration Layer and will survive disconnection/crash of the writer application. If all the nodes building Integration Layer will go down, the data will be lost.
PERSISTENT	The data is kept on permanent storage, so it will survive a system down time. Depending on Integration Layer it can be a function of the writer or a special service for storage and recovering of the Integration Layer.

The dispose method of the IL_Client is used to dispose Topics created by this IL_Client. After disposal the Topic will stop receiving notification and sending them to the listener.

The methods `login` and `getTopic` have the attribute of type `IL_Error`. Its purpose is the “emulation” of missing exception support over dynamic libraries. In case of an error, the method would return false/null and in the IL_Error-data structure would contain the type of the error and an error message. The implementation of the IL_Error is as follows.

```
class IL_Error {
public:
  int code;
  char *message;
  size_t messageCapacity;
      /**
      * Constant indicates that an operation was successfully performed.
      */
  static int const SUCCESS = 0;
      /**
      * Constant indicates that a generic error occured.
      */
  static const int ERROR_GENERIC = 1;
      /**
      * The buffer provided to the method is not big enough
      */
  static const int ERROR_NOT_SUFFICIENT_BUFFER = 2;
};
```

Additional types of errors will be identified and extended during further Shift2rail projects.

4.1.3 Dispose topic

```
void IL_Client::dispose(IL_Topic* topic);
```

Parameter IL_Topic* topic defines the topic to dispose. The pointer to IL_Topic is not allowed to be deleted in any other place.

4.1.4 Register data type

```
bool IL_Client::registerDataType(const char *cdmModuleName, const char
*cdmModuleContent, IL_Error & error) = 0;
```

Description

The IL_MessageConverter to start working require a central definition of the supported data types in xml format. Here the single files containing such definitions are sent to IL_Client and its children.

Parameters

Parameter	Type	IN/Out	Description
cdmModuleName	const char *	In	Contains the module name of the data type specification.
cdmModuleContent	const char*	In	Contains XML specification of the data format used for CDM.
error	IL_Error &	Out	Contains error code and error message in case of failure.

Return value

True – success, false – failure;

4.1.5 Retrieve message schemata

```
virtual bool getMessageSchema(const char *cdmModuleName, IL_EncodingFormat
fmt, IL_OutStream & os, IL_Error & error) = 0;
```

The Integration Layer provides automatic validation of the message content inside of the MessageConverter. To allow external validation of the messages and to simplify code generation for parsers the IL_Client provides schemata converted from the Canonical Data Model into the target serialisation format: .proto-files, Xml schema, JSON schema, ASN.1 message specification.

4.1.6Get message converter

The last method IL_Client::messageConverter gives access to the message converters (s. section 4.4). Both methods return references to objects are managed (incl. deleted) by IL_Client – they are not created at each function call.

4.2 IL_Topic

The class IL_Topic provides an access point to modify key-value-objects and to subscribe to these modifications.

```
class IL_Topic
{
public:
  virtual const char *id() const = 0;
  virtual IL_Client* parent() const = 0;
  virtual bool setListener(IL_Listener* listener, const char
*filterExpression, IL_Error & error) = 0;
  virtual IL_Listener* listener() const = 0;
  virtual const char *dataType() const = 0;

  virtual bool putValue(const IL_KVPair & kvp, IL_Error & error) = 0;
  virtual bool deleteKey(const char *key, IL_Error & error) = 0;

  virtual bool requestValue(const char *key, IL_Error & error) = 0;
protected:
  virtual ~IL_Topic() {}
};
```

Method	Description
id	Returns Id of the topic used at construction by IL_Client::getTopic.
parent	Returns pointer to the parent-IL_Client, which can be used for disposal of the topic and to access MessageConverter provided by the API.
setListener	Puts listener from the application to be notified about modifications of the objects in the Topic. filterExpression specifies a regular expression for keys – only keys matching the filterExpression will be distributed to the listener. Empty or zero value for filterExpression means “distribute all values”.
listener	Returns the current listener.
dataType	Returns the protobuf-type id provided to the Topic in IL_Client::getTopic as attribute TopicConfig::dataType.
putValue	Modifies the value, which must be provided in raw-

Method	Description
	format (see section with MessageConverter).
deleteKey	Removes the key-value with the specified key from Integration Layer. Subscribers will be notified about it.
requestValue	This is an auxiliary function. Some IL-implementations send in case of modification of a value only its key, expecting the client application to request the changed value afterwards if it is interested in it. Other IL-implementations send modified key and values together. The function requestValue shall be called only in the IL_Listener::keyAlive if the parameter "isAlive" is true, which means the value is modified, but it shall be extra requested. Providing this function allows reduction of the network traffic for big values.

4.2.1 Key-value-pair message

```
struct IL_KVPair {
      const char *key;
      const char *rawValue;
      int rawValueSize;
      int64 senderTimestamp;
      const char **attributes;
};
```

Attribute	Type	Description
key	const char *	Key used by IL to identify objects.
rawValue	const char*	Value in proprietary (raw) format.
rawValueSize	uint32	Length of the raw value.
senderTimestamp	int64	Number of microseconds since epoch (1970.01.01)
attributes	const char **	A list of pairs in form of "key", "value". The end of the list is marked with nil-key. It represents an extension point for future standardisation of the meta-message information

4.3 IL_Listener

The IL_Listener provides the only way to receive values from the Integration Layer. Its functions are called asynchronously from some threads created by the Integration Layer. The Integration Layer expects that the client application spends max 100 milliseconds inside of the methods.

```
class IL_Listener
{
public:
  virtual void valueArrived(const IL_KVPair & kvp, IL_Topic* topic) = 0;
  virtual void keyAlive(const char *key, bool isAlive, IL_Topic* topic) = 0;
  virtual void IL_Error(const IL_Error & error, IL_Topic* topic) = 0;
};
```

Method	Description
valueArrived	Provides a key-value-pair in one of the following cases: - The client application called topic.setListener (in another thread)

	- The value was created on the Integration Layer - The value was modified on the Integration Layer.
keyAlive	Provides the live-status of the key-value in Integration Layer if it changed after the call topic.setListener. If the value of the attribute isAlive is true, the client application has to call topic->requestValue(key) immediately or later if the application is still interested in it. The requested value will be delivered over valueArrived method asynchronously.
IL_Error	The call back function for any kind of errors like lost connections, memory overflows etc. Specific errors will be defined during Shift2rail projects.

4.4 IL_MessageConverter

Message converter shall create one of the standardised representations out of the proprietary raw data format used in specific Integration Layer implementation.

```
class IL_MessageConverter
{
 virtual bool raw2value(const char *dataType, const char *raw, size_t rawSize, IL_OutStream & os, IL_Error & error) = 0;
 virtual bool value2raw(const char *dataType, const char *value, size_t valueSize, IL_OutStream & os, IL_Error & error) = 0;
};
```

4.4.1Conversion from proprietary to standard representation

bool MessageConverter::raw2value(const char *dataType, const char *raw, size_t rawSize, IL_OutStream & os, IL_Error & error);

Description

Writes converted value into the IL_OutStream.

Parameters

Parameter	Type	IN/Out	Description
datatype	const char *	In	Defines the data format in Module.struct format.
raw	const char *	In	Value prepared for IL/received from IL in raw format
rawSize	size_t	In	Length of the raw message
os	IL_OutStream	Out	Output stream for the output value.
error	IL_Error &	Out	Contains error code and error message in case of failure.

Return value

True in case of success, false – otherwise.

4.4.2Convertion from standard to proprietary representation

```
bool MessageConverter::value2raw(const char *dataType, const char *value, size_t valueSize, IL_OutStream & os, IL_Error & error);
```

Description

Writes converted value into the OutStream.

Parameters

Parameter	Type	IN/Out	Description
dataType	const char *	In	Defines the data format in Module.struct format.
value	const char *	In	Value in one of the standard formats to be converted to IL-specific format.
valueSize	size_t	In	Length of the value message, required at least for binary standard formats.
os	IL_OutStream	Out	Output stream for the output value.
error	IL_Error &	Out	Contains error code and error message in case of failure.

Return value

True in case of success, false – otherwise.

4.5 IL_OutStream

```
class IL_OutStream
{
public:
  virtual bool getNextBuffer(const void ** data, int *size) = 0;
  virtual void ignoreLastBytes(int count) = 0;
  virtual int64 byteCount() const = 0;
};
```

Method	Description
getNextBuffer	Returns pointer to the next allocated buffer. Return true in case of success.
ignoreLastBytes	Notifies the IL_OutStream, that the writing is finished, and the last count-bytes of the last getNextBuffer were not used.
byteCount	Returns current size of the stream.

An example implementation of the IL_OutStream can be found on Protobuf project as e.g. StringOutputStream with a little different function names.

4.6 Factory functions

As most of the factory functions are integrated into IL_Client remain only two functions one for creating IL_Client and one for its disposal:

```
IL_Client *createIL_Client();
void disposeIL_Client(IL_Client*);
```

5 Integration Layer Security (Session Management)

This section describes session management is implemented. The session management is enabled in cooperation by the Integration Layer and the Application Framework. Sessions are provided to applications running in the context of the Application Framework; the naming convention used reflects this fact.

5.1 Definitions and assumptions

5.1.1 Definition of AF session

The AF session allows an end-user to start a sequence of related operations on the data set nodes after the end-user successfully completed its authentication on the system. In this document, the authentication is referred as login.

Rationale: This definition is useful to describe how the term session is used in this section, and to distinguish it from many other meaning the term has in current literature.

5.1.2 Definition of end-user

End-users are application running in the context of the AF and, by extension, human users interacting with the aforementioned applications.

5.1.3 Definition of reservation

A reservation defines:

- the list of access permissions of the owner of the reservation over the selected CDM Data Set node(s). The owner of the reservation may request read-only control to delegate the protection of the reserved node to the mechanism in charge of verify if node can be written on. This is useful when write request start from a third application; in this case the application doesn't need to check if the write should be executed, it simply pass it to the wrapper that shall reject the write operation;
- the list of access permission the owner of the reservation grants to other end-user(s).

5.1.4 Definition of access permission

Access permission are: full control, read only, delete. Each end-user may have an associated list of (node, access permissions…).

5.1.5 Assumptions

The following assumptions have been made:

1. the CDM Data Set nodes will be accessed only via the IL;
2. there's a specialized AF component providing services for AF reservation management; this component is referred as AF session manager. Services are provided via methods of the API. The AF session manager is either an application running in the AF or a distributed "logic" laying in the AF API used by the applications, see also [D8.6];

3. access permission will be stored and accessed in the CDM Data Set, thus they will become part of the CDM.

   Rationale: This is an attempt to be free from the technology used to manage authorization. No assumption is made on where the access permission will be stored in the CDM; a replica of the CDM structure with nodes containing access permissions is an option.

5.2 Functional description of session management

1. The end-user is able to start AF sessions after its authentication to the IL;
2. the end-user is free to start more than one AF session at the same time;
3. the AF session manager provides the end-users the following session management methods: StartSession, EndSession, Reservation, Free, WatchDog, HandOver, TakeOver;
4. StartSession – the start session method return an AF session handle. Sessions keeps tracks of all the reservations;
5. EndSession – the stop session method frees all the reserved nodes (see point 7);
6. SessionWatchDog – the watchdog method will rearm the watchdog timeout period on the specified session handle. The AF session manager decrements and monitors the watchdog of each session in order to terminate the session and to remove from data set sandboxes not yet checked in;

   Rationale: here the assumption is that sandboxes have a lifecycle similar to change sets of a versioning system. Checked in means the fact that change sets are ready to be committed on data set.
7. Reservation:
   a. Reservation – the reservation method will reserve the requested list of nodes of the Data Set, provided that the end-user is authorized to reserve them,
   b. If successful, the reservation method returns a Reservation handle,
   c. If unsuccessful, the reservation method returns the reason of failure (e.g. already reserved by another end-user, type of access not authorized, not existing node),
   d. More than one Reservation may be performed in the context of the same AF session,
   e. If the end-user reserves a node, it will reserve all of its descendants,
   f. The end-user may specify access rights for other users,
   g. Free – the free method frees the reserved nodes. Free is an ad-hoc method that can operate on reservation handles (i.e. all the nodes associated with the reservation handle) or on single nodes,

      Rationale: To help synchronization between concurrent reservation operations (i.e. manage race condition) from different application, one application may reserve an entire branch till needed, and then free some sub-branch nodes; note that this will modify the reservation.
   h. ReservationWatchDog – the watchdog method will rearm the watchdog timeout period on the specified reservation handle. The AF session manager decrements and monitors the watchdog of each reservation in order to free the nodes associated with the handle;
8. Sandbox:

   a. Sandbox – the sandbox creation method will add a sandbox node to the Data Set (please refer to Sandbox section),
   b. If successful, the sandbox creation method returns a SandboxId,
   c. Rationale: The SandboxId is an handle used in the following operations: i) automatic garbage collection when a session is terminated; ii) sandbox or session handover/takeover between users; iii) sandbox handover to AF components that merge sandboxes,
   d. If unsuccessful, the sandbox creator method returns the reason of failure (e.g. type of permissions not granted, not existing path),
   e. More than one Sandbox may be performed in the context of the same AF session;
9. HandOver – the handover is an ad-hoc method that operates on one AF session, AF session reservations, or AF session sandboxes:
   a. On AF session, it prepares the transfer of all the AF session reservations to another specified AF session – see TakeOver,
   b. On reservations, it prepares the transfer of the reservations to another specified AF session – see TakeOver,
   c. On Sandboxes, it prepares the transfer of the sandboxes to another specified AF session – see TakeOver.
10. TakeOver – the takeover method is an ad-hoc method to acquire all the reservations of another AF session, reservations or sandboxes belonging to other AF sections. Takeover shall be executed in a specified time since the handover. TakeOver may be forcefully executed by another session.

5.3 Proposed solution

In this section the following topic will be described:

- access permissions;
- reservation;
- AF session management;
- wrapper implementation hints.

The activity diagrams of this section are provided in order to show a conceptual view of the different actors and activities; their actual implementation will depend on the final architecture of the AF and IL.

5.3.1 Access permissions and Reservations on CDM

The Canonical Data Model instance consists of two parallel structures containing:

- values of data (Data Set);
- access permissions and reservation about data (Access Permissions Set).

The Access Permissions Set depends on two aspects:

- basic access rights assigned to users and groups (manged by authorization service);
- reservations assigned by the owner of the data nodes (dynamic attributes, managed by AF session manager).

The reservation parameters contained in Access Permissions Set are:

- Reservation handle list
- Users list and related access rights assigned by session owner
- Groups list and related access rights assigned by session owner

Access rights assigned by session owner must be compliant with the basic access rights assigned by authorization service.

This solution is compliant with both IMDG and queue messaging systems that can be used to implement the IL.

5.3.2AF session manager

The AF session manager holds all the AF sessions and their related properties:

- session Id;
- session Owner;
- session Name (plain text name);
- session Start Timestamp;
- session Expected Duration;
- session watchdog timeout (i.e. watchdog period);
- reservation handle list:
  - Reserved nodes list,
  - Reservation watchdog timeout,
- SandboxId list.

The following UML activity diagram shows the behaviour of AF session manager for reservations.

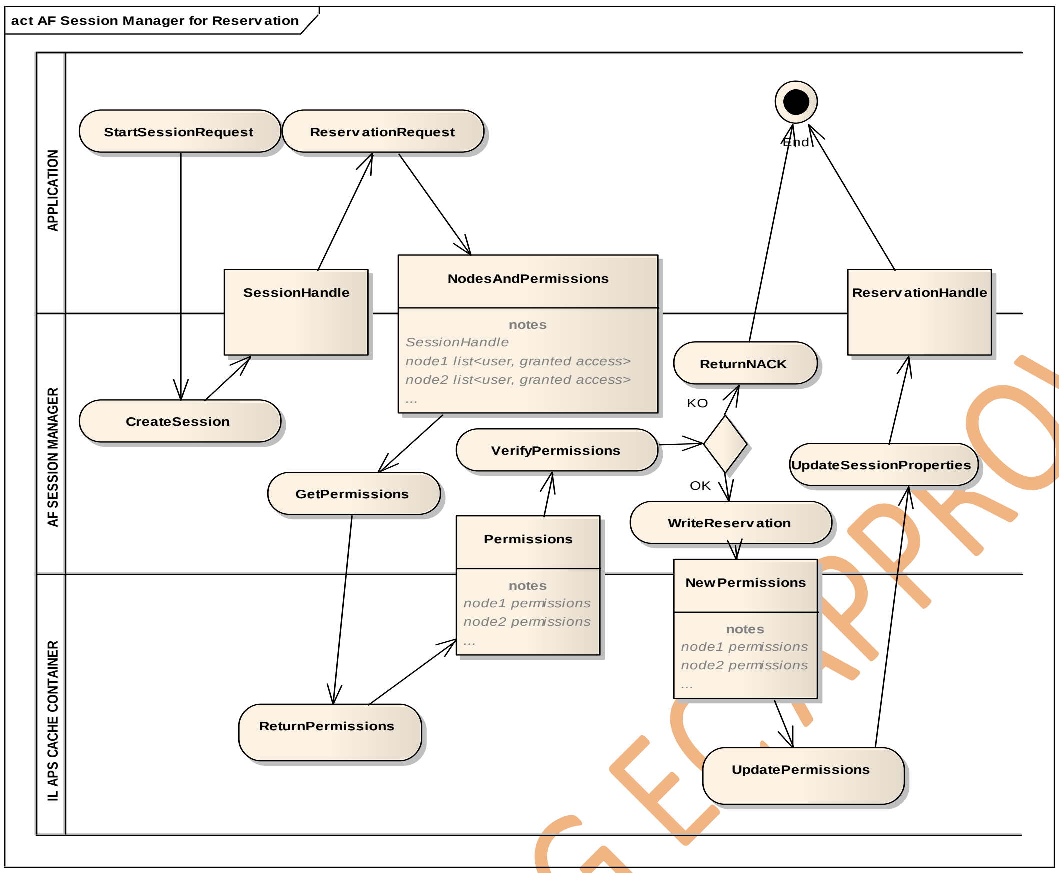


Figure 5.1: AF Session Manager – activity diagram for Reservation

In the following UML activity diagram the behaviour of AF session manager for sandbox creation is described.

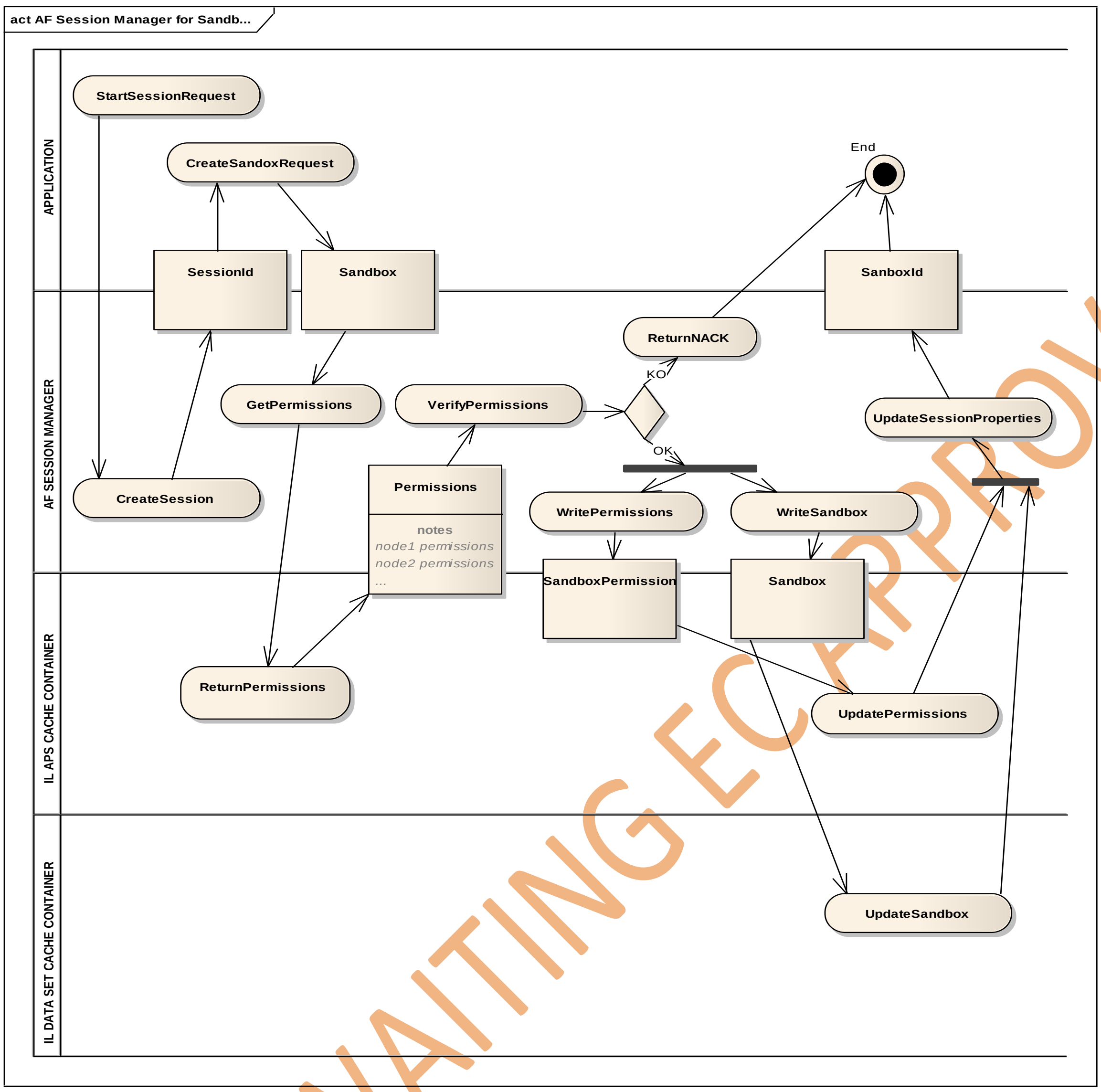


Figure 5.2: AF Session Manager – activity diagram for Sandbox creation

In the above figures, the two processes of "Reservation" and "Sandbox creation" are separately described for clarity: during normal operation, Reservation and Sandbox creation may be interleaved in the same session. Due to the fact that Sandboxes are implemented as nodes of the Data Set, reservation could be obviously applied to them.

5.3.3 Wrapper implementation hints

In the proposed solution, the wrapper of the IL is responsible to verify if the application (or end-user) has the rights to perform the required action.

The following UML activity diagram shows an example of write authorisation, supposed that:

- the IL is implemented with an IMDG implementation of the IL is shown, and;
- the end-user has started an AF session.

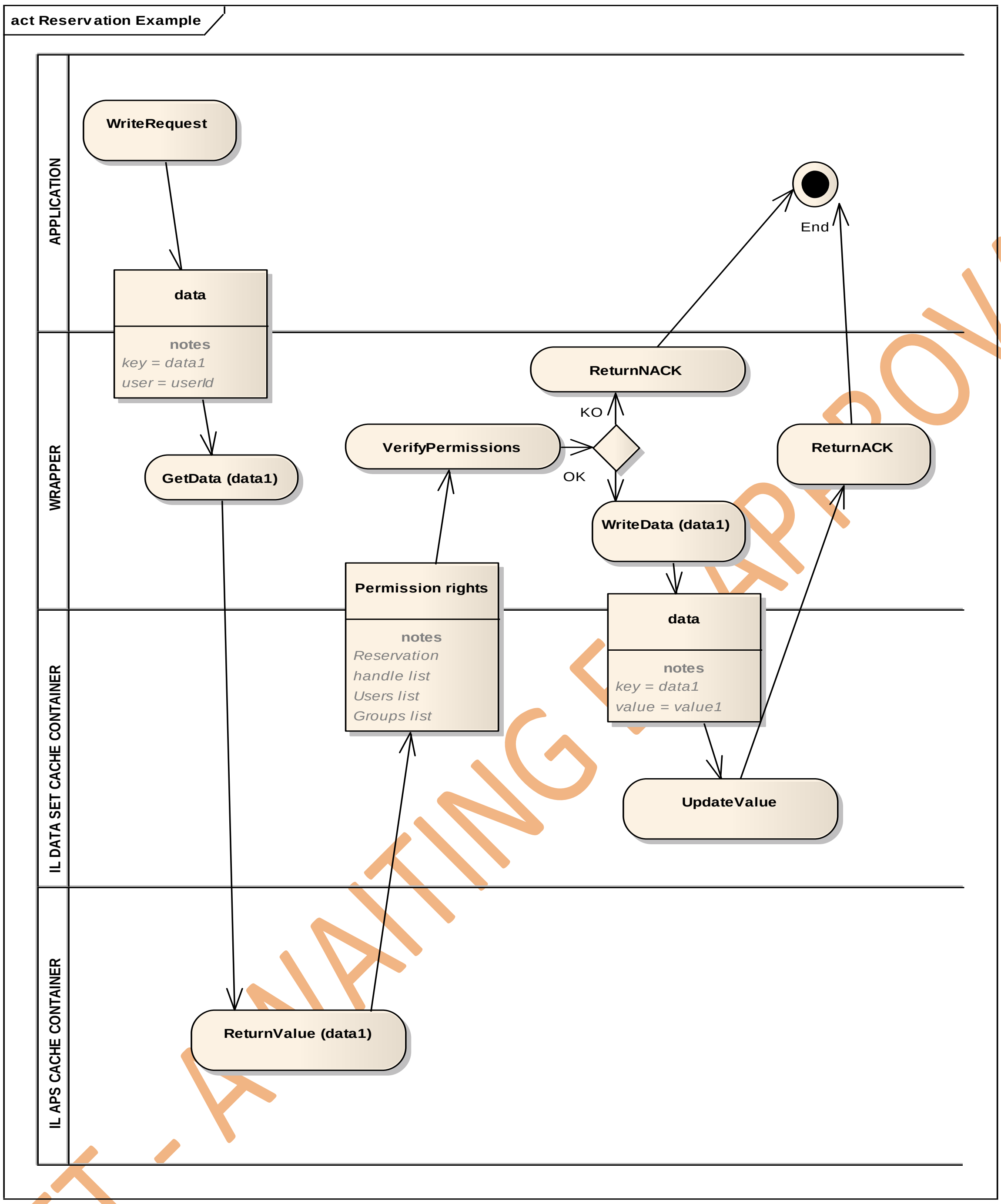


Figure 5.3: Activity diagram for Write permission checks

A generic application request the wrapper to write a new value of “data1”. The wrapper accesses to the IL cache container (i.e. the place where the IMDG stores its key-value elements) of “Access Permission Set” in order to acquire permission rights. The wrapper performs the write operation on “Data Set” cache container only if the permission rights are compliant with the required action.

5.3.4Overall description of the proposed solution

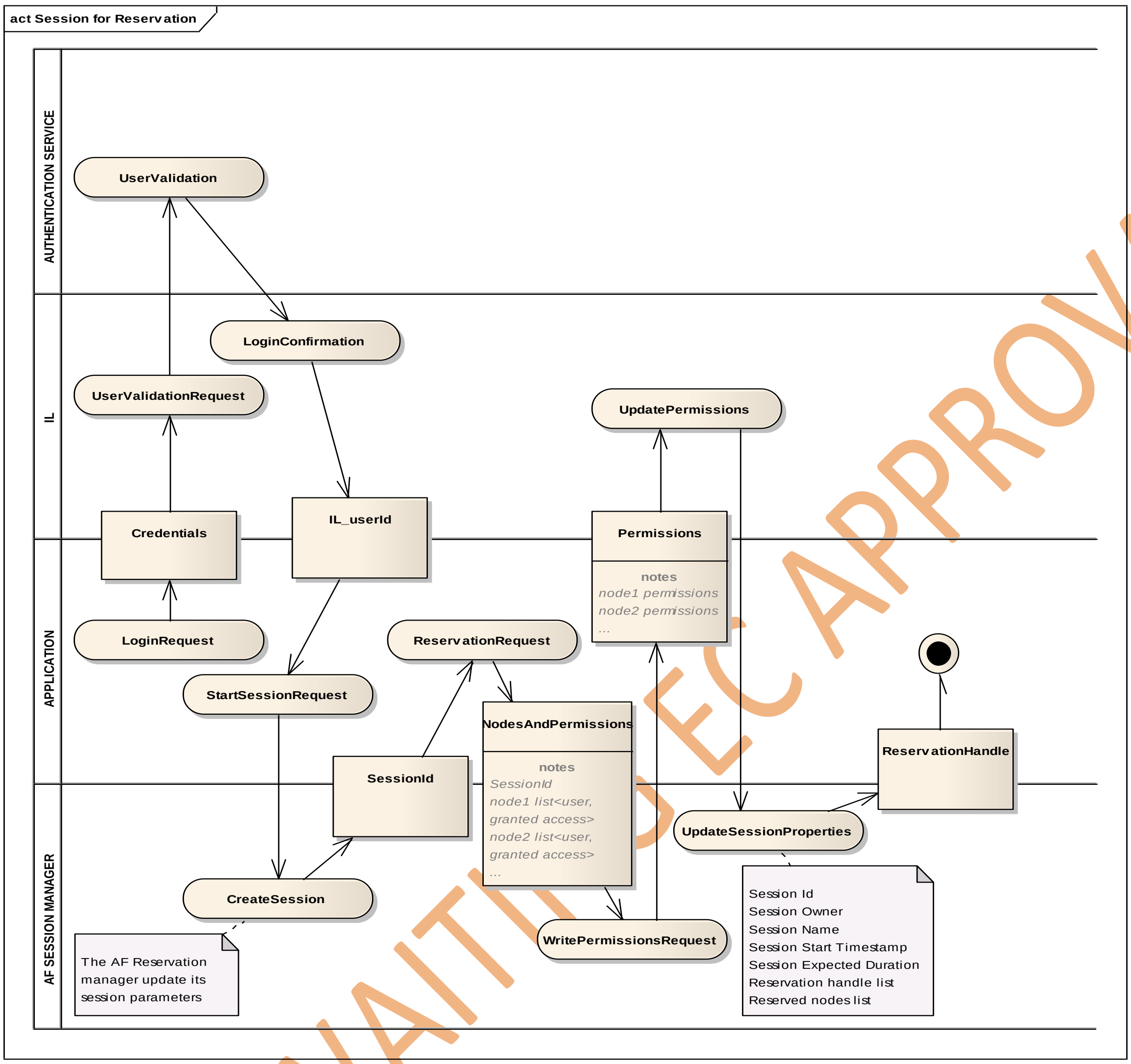


Figure 5.4: Activity diagram for the proposed solution

The above activity diagram depicts the proposed solution, encompassing: the login to the IL, the start of an AF session, and a reservation od some CDM Data Set nodes. Here, the authentication of the user (UserValidation) is provided in cooperation by the IL and an external authentication service.

6 Sandbox/Version management in IL

6.1 Introduction

In recent tenders for TMS there are requirements for support versioning of the Real Time Traffic Plan (RTTP), like:

- create a new version of RTTP with a first modification initiated by an operator;
- provide a workflow for merging the locally modified version with the “master” RTTP;
- be able to restore RTTP to any past version before merging the modified RTTP.

For covering of this functionality for RTTP a special service is required. Instead of limiting the service to “Versioning” of RTTP, a generic approach is specified, which is able to provide version management for an arbitrary part of the CDM (e.g. to cover construction works on Topology-Part of CDM).

The second aspect of version management is coverage of transactions. Integration Layer provides access to the data by a (high performance) publish-subscribe pattern. The drawback of this approach is that the client applications receive modifications asynchronously, so clients are unable to identify the transaction boundaries (point in time, when the data is claimed to be valid). Another issue represents the lack of transaction support of many IMDG-products on the market.

To cope with these issues the “version management” functionality can be used: in the context of TMS areas requiring transactional support, require explicitly “version management” as well, e.g. RTTP. But “version management” of the data set covers all aspects of the transaction in a perfect way, so that additional support of the transactions from an IL-product is not needed. This allows selection of low-cost-solutions from the market as basis for IL.

The approach behind the data-set versioning is not very different from the common Version Management Systems like SVN, Git, HG, RTC. In the following we use the concept of “Sandbox”, which comprises a sequence of changes, which applied after each other create the current state of the data set (see Figure 6.1).

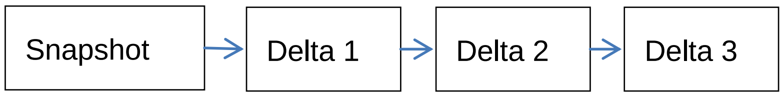


Figure 6.1: Sandbox = sequence of changes

The sandboxes can be stacked on each other creating branches. In Figure 6.2 the Sandbox B is using Sandbox A as a basis. It means that the client application first has to apply all changes in Sandbox A, and then all changes of Sandbox B.

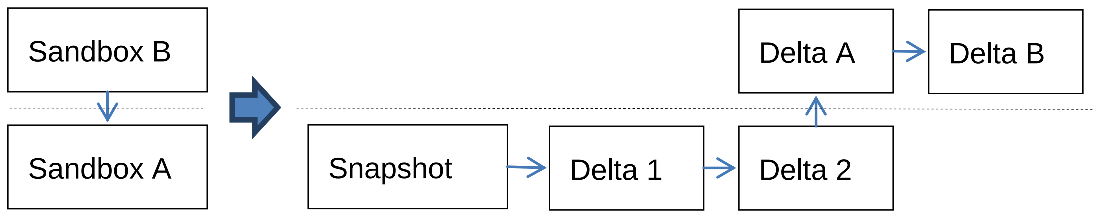


Figure 6.2: Sandbox stacking: apply additional changes to the base-sandbox

6.2 Interface specification

To handle Sandbox functionality a dedicated Sandbox management service has to be specified. On the Integration Layer several topics are required, divided into two groups:

- Sandboxes-Management-Topics handling creation and removal of the Sandboxes;
- ChangeSets-Management Topics handling change sets of a one specific Sandbox.

The list of the required topics is shown in Table 6.1.

Topic.addressPrefix	Data type	Description
/xxx/SBMgmt	SM.SBMgmtCommand	Request to create or removal of Sandboxes
/xxx/SBList	SM.SandboxInfo	Contains the list of sandboxes with their configuration.
/xxx/SBMgmtNotifications	SM.RequestNotification	Replies/notifications on management commands for creation and removal of Sandboxes.
/xxx/MySandboxA/ChangeSets	SM.ChangeSet	A sequence of change sets building a sandbox. Keys are numbers in sequence starting with 0 in hex-representation.
/xxx/MySandboxA/ChangeRequests	SM.ChangeRequest	A set of concurrent requests from client applications. Keys must be unique, e.g. random strings or patterns like /sender/id.
/xxx/MySandboxA/Notifications	SM.RequestNotification	Replys/notifications on change requests. The key is identical to the key of the change request.
/xxx/MySandboxA/Snapshots	SM.Snapshot	Contains snapshots = a set of changesets, which contains only all previous changes. The client can either apply all previous changes or the snapshot.

Table 6.1: Topics for sandbox management service

A class diagram with the part of canonical model representing Sandbox management functionality is shown in Figure 6.3.

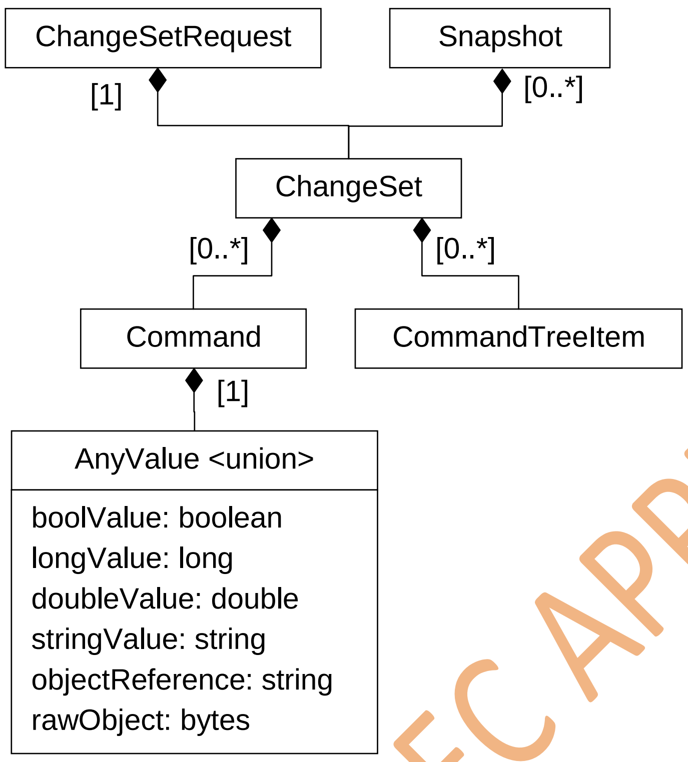


Figure 6.3: Data structures for management of one sandbox in the module SM (SandboxManagement)

For the management of the set of sandboxes additional data structures are required (see Figure 6.4).

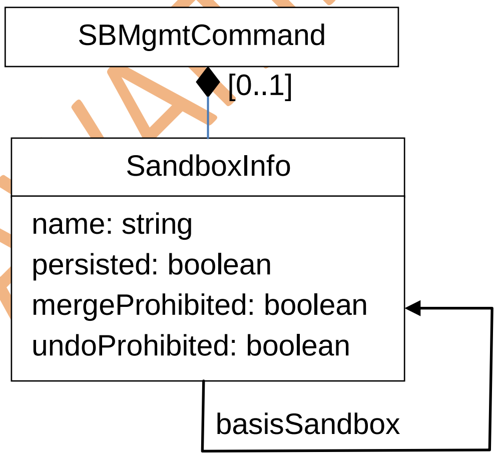


Figure 6.4: Data structures for management of a set of sandboxes in module SM (SandboxManagement)

The sandboxes with attribute persisted equal to true, the topic containing the ChangeSets will be persisted.

If the attribute mergeProhibited is false, the SandboxManagement service will try to merge new commands with previous once. The clients must be able to apply modified ChangeSets.

If the attribute undoProhibited is false, removal of the ChangeSets from the topic has a meaning of “undo” operation and the client applications shall be able to follow this request.

In this case it has to keep the ChangeSets together with previous values either in memory or locally persisted. A typical use case for enabled Undo-Sandboxes represent "small" private sandboxes of the operators, so it is safe to keep previous values in memory without a risk of a memory overflow. For the master sandbox the undo operation is typically prohibited.

A typical representation of the sandbox sequence is shown in Figure 6.5.

/xxx/SBList	
key	value
master	`Persistent:true` `undoInhibited: true`

/xxx/master/changerequests	
key	value
rnd-key-23	x7x

/xxx/master/changesets	
key	value
a23c	`Snapshot: true`
a23d	xxx
a23e	xxx
a23f	xxx
a240	x7x

/xxx/master/snapshots	
key	value
a23c	A very big changeset

/xxx/master/requestNotifications	
key	value
rnd-key-23	`Accepted: true`

Figure 6.5: Example of topics content building a "master"-sandbox short before the request rnd-key-23 is completed

6.3 Transactions management in IL

Inside of IL the key-value pairs are modified by applications. The modification types comprise removal, inserting and update of key-value pairs. IL notifies subscribed applications about modifications. There are several use cases for transactions in this context:

1. consistent data distributed among key-value pairs. If the client modifying a set of KVP crashes during the modification, the resulting data set can be invalid;
2. clients subscribing for change notifications on KVPs receive modifications asynchronously, typically the smaller KVPs arrive earlier to the client, so it does not know, if the consistent state is already reached or not (if all modifications made by one transactions already arrived). For the duration of transfer the biggest modified object (typically 0.1 sec) the receiver application could have an inconsistent data state and notify the user about inconsistencies;
3. concurrent "overlapping" modifications – if several clients concurrently modify objects in Integration Layer, the last writer wins overwriting the changes of the previous one. This is typically solved by locking objects before writing them. The locking functionality is not

specified yet in the IL. Some products on the market support locking (Hazelcast, Redis), some require a dedicated locking service (Opensplice DDS).

The use case 1 and 3 are normally covered by transaction management of an IMDG (e.g. in Infinispan, Hazelcast, Redis). The use case 2 cannot be covered by Integration Layer implicitly. Clients requiring “any-time” consistency have to subscribe to the sandbox management.

The current assumption in In2rail is that the sandbox management functionality provides sufficient support for transactional behaviour in TMS. Therefore the IL-API does not provide support for transactions. If during the prototype development will come out that real transactional support is unavoidable, either API will be extended to support transactional put and get operations or a dedicated service for transactional get operations together with sandbox-based put interface will be specified.

7 Integration Layer Topics

The following sections describe the different Information Topics used in the Integration Layer. Topics are a group of data used for the exchange of data between the different systems involved in the traffic management process, e.g. the Traffic Management System, the Interlocking, the Energy Management System, etc. Technically Topics are a representation of a composition relation in the Shift2Rail data model tree, which can be addressed by a key.

There is a wide range of information that can be distributed via the IL. For In2Rail the scope is limited to the information of the core areas that are needed to be exchanged from and to the TMS. Further developments and enhancements of the Topics and data model will be done in the Shift2Rail Projects X2Rail2 WP6 and IMPACT-2 WP7.

The considered areas are:

- Infrastructure (Railway Topology);
- Timetable;
- Traffic Control and Command (including Process Image);
- Energy;
- External Services (WEB-Interfaces).

More information about the scope of the In2Rail data model, called Shift2Rail CDM, the composition of it and initial starting points are described in [D8.3].

For identification of the Topics, the following rules served as guidelines [D8.3 Chapter 6.4.4]:

- data objects that normally belong to different organisations should be separated in different topics, e.g. infrastructure assets, rolling stock, energy system;
- data objects that interact closely should be in the same topic, e.g. signals, switches and other signalling elements;
- safety related data should be separated from non-vital data. Please note that potential impacts on assurance levels for applications will have to be assessed and considered in future Shift2Rail project deliverables (e.g., X2Rail-2, X2Rail-4);
- the three main classes of data persistence spans should be separated, i.e. static data (e.g. topology data), dynamic data (status information, which changes in real time) and historic data (e.g. event logs)

7.1 Infrastructure Topics

During the State-of-the Art analysis of existing data standards in the area of Traffic Management, further described in [D8.3] Chapter 6.2.1, also existing infrastructure models are taken into account.

The analysis encountered that the RailML standard is a good starting point for the S2R CDM infrastructure modelling. RailML provides two infrastructure models:

- RailML version 2;
- RailML version3 with RailTopoModel.

RailML version2 is already a stable version and connected to the timetable part of RailML, but it is not further developed since version 3 with the new RailTopoModel is introduced. RailML v3 in contrast is still under development by the RailML community und not yet tested in production.

Both models only contain static elements and have to be enhanced with dynamic data elements which are missing in the current model since it is used for planning purposes only.

New parts need to be developed in the RailML model to make it usable for the traffic management process. Since RailML v3 and the RailTopoModel are not available early enough for a deeper analysis, we decided in WP8 to shift the whole modelling of the Infrastructure part to the S2R projects X2Rail-2 WP6 and IMPACT-2 WP7.

7.2 Timetable Topics

A timetable represents a planning concept for the train movements on a network. The process of timetable creation undergoes several steps.

The timetable creation starts with the need of an operator to run their rolling stock on the network infrastructure. For this the operator sends a request for a train path to the responsible Infrastructure Manager. After the negotiation phase between the operator and the IM, where the real train path is generated, the train becomes a resource on the network.

The Infrastructure Manager gets several train path requests from one or different operators. This planned trains forms the Long Term Planning timetable or seasonal timetable.

This seasonal timetable is published to the Passenger at a specific time of the year. This is called the published timetable.

During the year the planned timetable and trains are changed several times due to maintenance on the infrastructure, changes from the operator on the requested train path and new train requests. This planning phase is called the Short Term Planning and the updated timetable is saved as an operational timetable, also called the Offline Production Plan (OffPP).

The Offline Production Plan (OffPP) is send by the planning department on a daily base to the traffic management department and stored as a reference for the originally planned timetable. Out of the OffPP the Online Production Plan (OnPP) is generated within a TMS containing rescheduled train timetables due to actual restrictions and other constraints. The OnPP is the basis for the operation and it is distributed to appropriate users, like operators, drivers and PICOPs.

Due to deviations in the running times and other actual state information the OnPP is rescheduled continuously and consistently followed by distribution to external modules or recipients as shown in Figure 7.1.

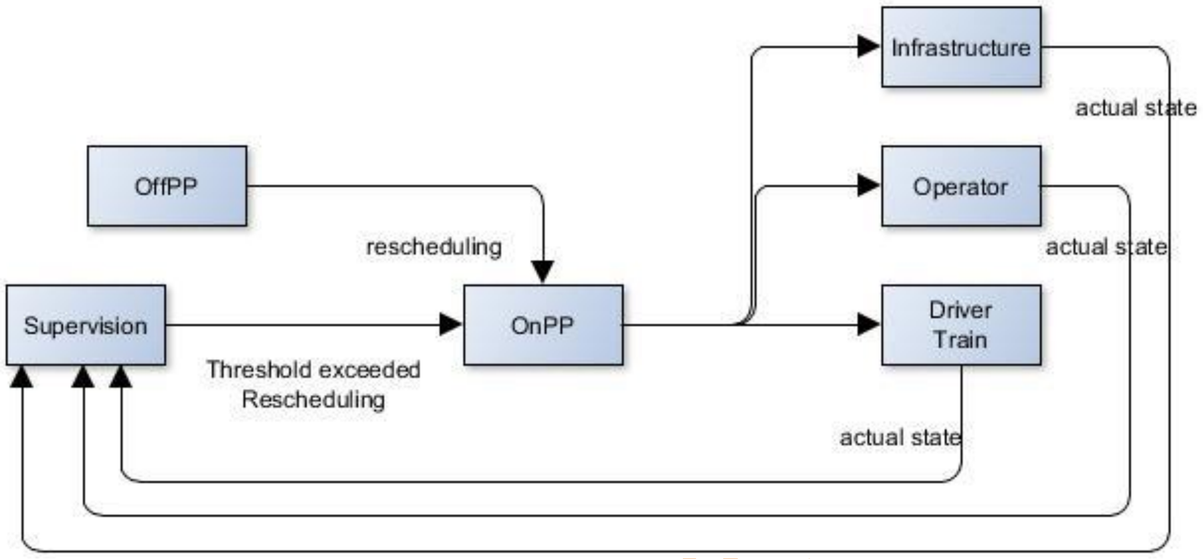


Figure 7.1: Production Plan Loop

Another timetable in the Traffic management system is the forecast timetable. The Forecast itself is the result of re-calculating the trains of the actual OnPP in order to derive exact train running based on incoming train positions and apply conflict detection. Some conflicts may not be resolvable automatically (e.g., due to complexity or missing activation) or simply because the time horizon for applying conflict resolution has not been reached yet. In these cases, the forecast calculation usually applies business rules to estimate the time required for overcoming the respective conflicts.

The actual timetable contains the actual train movements imported from different sources like the ILS or on-board unit and persisted in the TMS.

Furthermore, a TMS may contain multiple emergency timetables, simulation timetables, etc.

The traffic management system is handling the last steps of the timetable creation process – from planned to the actual timetable. The TMS staff and services have several requirements on the data managed as timetables.

7.2.1Requirements

Before specific terms will be used for definition of train movements the term train will be used to represent a building block of a timetable in order to formulate the requirements.

Requirement	Rational
The timetable shall specify location and speed of a train in time and space	This is a basic requirement used by: - operator to monitor and control the traffic; - Automatic Route Setting service to establish train routes well in advance before the train movement; - Passenger Information System to inform passengers about arrivals/departures/delays; - Post-processing system for root cause analysis for delays and further deviations.
The timetable shall specify connections to other trains at any station or track element	Connections for join and split are required by the staff to know, where to split the train and in which sequence to join incoming trains. Passenger and sequence connections are used by the dispatchers to analyse degrees of freedom in case of delays of “trigger” trip. In an ideal world without delays, these kinds of connections are not required, as they are implicitly modelled by times. In the timetable they represent “parameters” for a “decision algorithm”, which decides if the train is allowed to leave some place (e.g. station).
Transport type (passengers, goods, empty) for each station.	The purpose of this attributes is similar to connections – the operator use them to identify required modifications in case of disturbances.
Dynamic characteristics (length, mass, acceleration/deceleration profiles) which can change in any station.	This data allows a simulation algorithm to calculate technically achievable running times.
Allow various levels of precision, e.g. to specify only start and end stations vs. detailed train movement.	There are several use cases: - Ad-hoc planning requests can be represented as a “not-completely” planned train run; - Planning system often delivers a not- completely planned train runs as they have to be adjusted to the current state of the network anyway (e.g. unplanned maintenance, speed restrictions, delays etc.). In most cases a detailed specification of train runs is required.
Provides train position and speed at distinct points in time with high precision.	These times can be used by a simulation algorithm to start run time calculations from the intermediate points of the train run. Low precision would produce flipping times in the train run depending on the simulation starting point.
Compact encoding	One use case is a shared forecast timetable distributed to all involved (several hundred) clients in very short time (around one second). At bigger disturbances and timetable modifications the number of distributed train runs for a medium

Requirement	Rational
	sized railway network can be several hundred as well.
Tree structure of the data model	A timetable is typically structured as a list of train runs (or trains). As the Integration Layer asynchronously distributes timetable modifications to the subscribed clients, a modified train (trip) shall contain all the modified elements in one data structure following the composition pattern. In this case no additional transactional behaviour is required, which is not supported by the planned Integration Layer.
Compact model representation in memory	The model will be used not only for sending messages about modifications of some part of the timetable to/from IL. It shall be possible to generate client code to reduce complexity (and the costs) for development of the client applications. That means to thrifty use of optional values, as in the client code the values will be created.

A good metaphor for the timetable represents a time-distance-diagram. As a standard mean for observing and editing of the train operations the time-distance-diagram represents very well expectation of the stuff and involved services. Hence the model of the train movement should provide enough information for "drawing" of a train on the time-distance diagram.

7.2.2 Timetable schema in RailML 2.3

A simplified class diagram of the RailML 2.3 timetable schema is shown in Figure 7.2. In the following sections several aspects of the specification will analysed for their compatibility with the requirements in Chapter 7.2.1.

7.2.2.1 Train reference TrainParts

The first observation shows, that the tree structure is dedicated to the planning purpose of a seasonal timetable: the reuse of TrainParts by several Trains simplifies adaptation as a bulk-editing of the involved trains with one modification.

Considering requirements from TMS, where the timetable is expected as a list of trains, this class diagram does not fit well to the Key-Value-based representation in CDM:

- assuming the "Train" concept to be a building block of the Timetable, it does not contain enough information for description of the train movements and references to TrainParts;
- TrainParts can be referenced by several Trains, therefore it is not a composition relation,
- in an Integration Layer where each Topic has a unique data type two topics are required to model trains: TrainParts and Trains. With missing transactions and

asynchronous delivery implemented in Integration Layer it would be complicated for the clients to reconstruct the current state of a train.

Therefore the first step for modification of the class diagram would be assumption, that Train should be a composite of TrainParts.

The next observation is, that TrainParts are “cutting” one Train into pieces with the same formation – as long as the length and the weight of the train are the same, the TrainPart can continue. For most of the train movements having const length and weight, the concept of TrainPart introduces an additional level of complexity without any payload. In case of the timetable planning process this separation is reasonable to enable reuse of “train-parts” in several trains. In case of TMS each train is handled completely independent from the others, and bulk-editing functionality is just a tool to simplify the handling by the human operator.

Therefore the next step of modification would be assignment of Formation-class to OcpTT, so each modification of the train dynamics can be annotated with a new formation values. In most cases only the first OcpTT would have one formation-object.

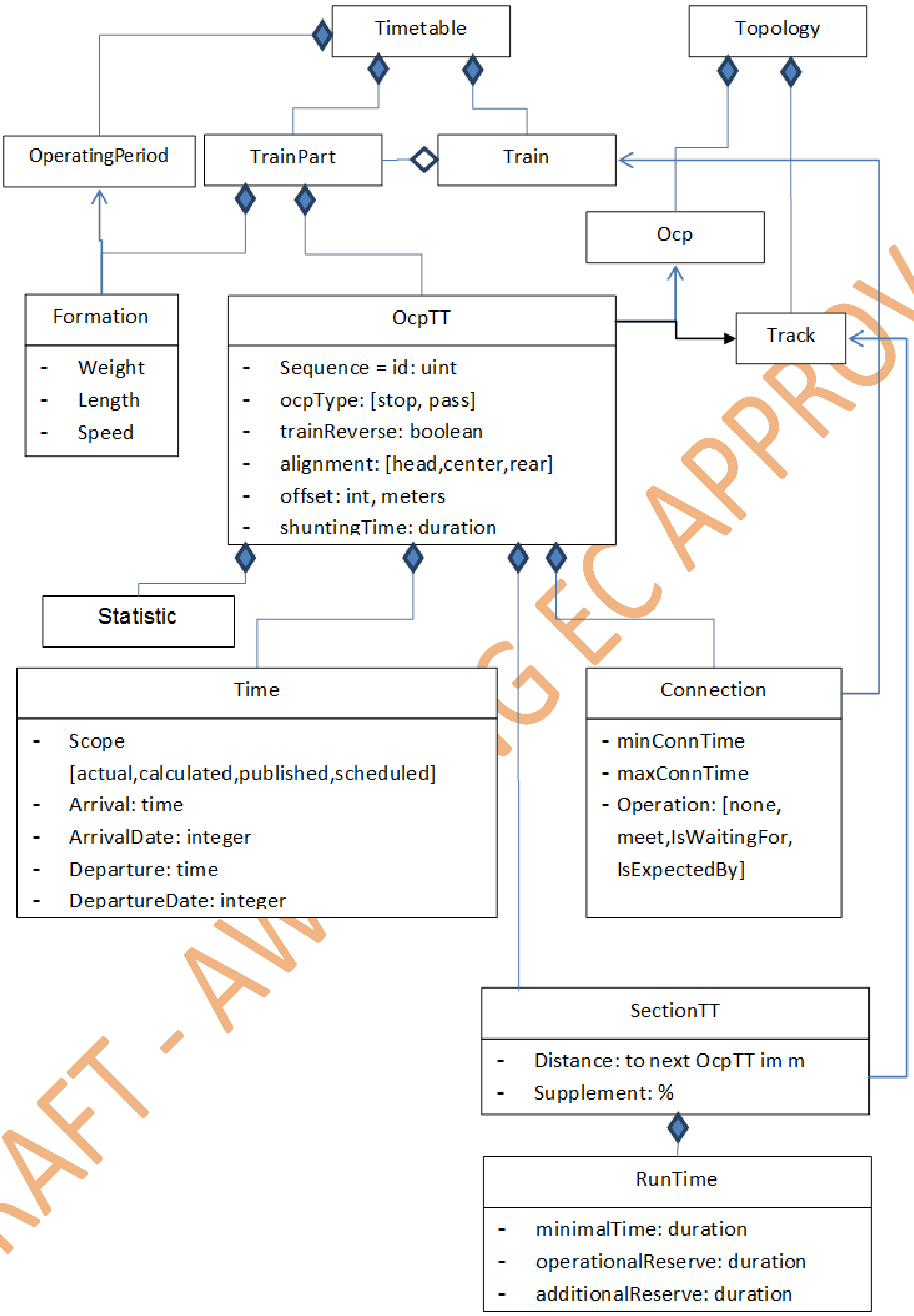


Figure 7.2: Simplified class diagram of Timetable schema in [RailML2.3]

7.2.2.2 Time scope

The next observation is the Scope-attribute of the Time-structure. It annotates the meaning of the times in the structure and the source of this information. In case of Integration Layer the Key-value-representations of some entity shall be separated by the source system:

- "actual" arrival time is coming from Train Tracking Service;
- "calculated" arrival is coming from the Forecast Service;
- "published" from Passenger Information System.

If all three sources would modify concurrently one data structure the last "writer" would overwrite changes made by other systems. Therefore the data shall be structured in a way, that it is modified by one service only.

This can be achieved if the "meaning" and "source" of the trains is encoded by the Topic where it is located (see Figure 7.4).

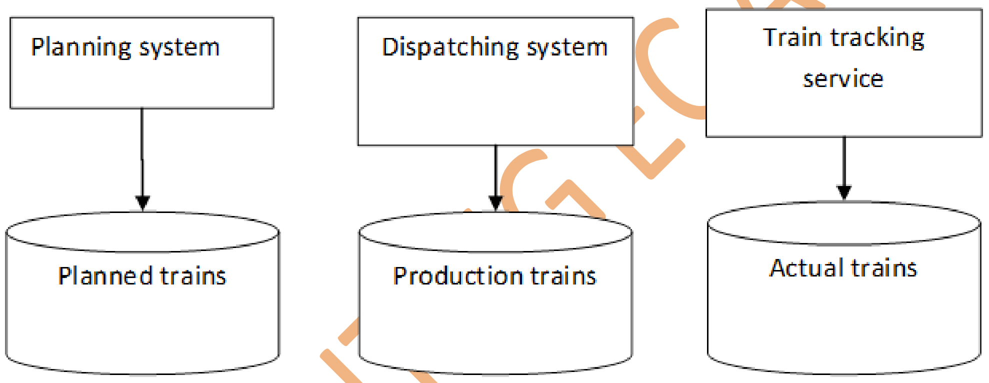


Figure 7.4: Reuse of Train-Data structure for representation of different types of timetables

A disadvantage of information duplication is compensated by a high flexibility and performance of this approach.

Therefore the modification would be to remove the "Scope"-attribute.

7.2.2.3 Referencing infrastructure

The next concept to be analysed is the referencing of the railway infrastructure. The absolute coordinates on the railway graph are specified in OcpTT as tuples <TrackReference, Ocp-Reference>, which refer to the Infrastructure schema, where the element Track.CrossSection contains OcpReference and its position on the Track. Ocp is a macroscopic node on the railway infrastructure, typically a station. It is an interesting approach which allows even abstract specification of the station to be visit, if the Track is not specified. A disadvantage is that with this approach the timetable (arrival, departures, connections etc.) can be specified only at station level, but not at signal level. Some Automatic Route Settings on the market support fine grained planning at signal level, e.g. it can ensure a train sequence in a station by assigning TrainConnections at entrance signals.

Therefore the required modification would be to allow assignment of timetable information to signals as well.

7.2.2.4 Splitting TrainPart into station and intra-station sections

The train line in time-distance diagram is splitted into two parts:

- at stations the absolute arrival and departure times are specified in OcpTT;
- between the stations relative running times (after the departure in OcpTT) and distances are specified in SectionTT-element.

This separation allows two main use cases:

- the times in OcpTT can be shown and edited e. g. a tabular timetable editor (or time distance diagram);
- a very simple algorithm can automatically adjust running times between stations using minimal(run)Time and additionalReserve specified in SectionTT. In this case the SectionTT gives an envelope for constructing the train line between stations in case of delay or editing of the source departure time or target arrival time.

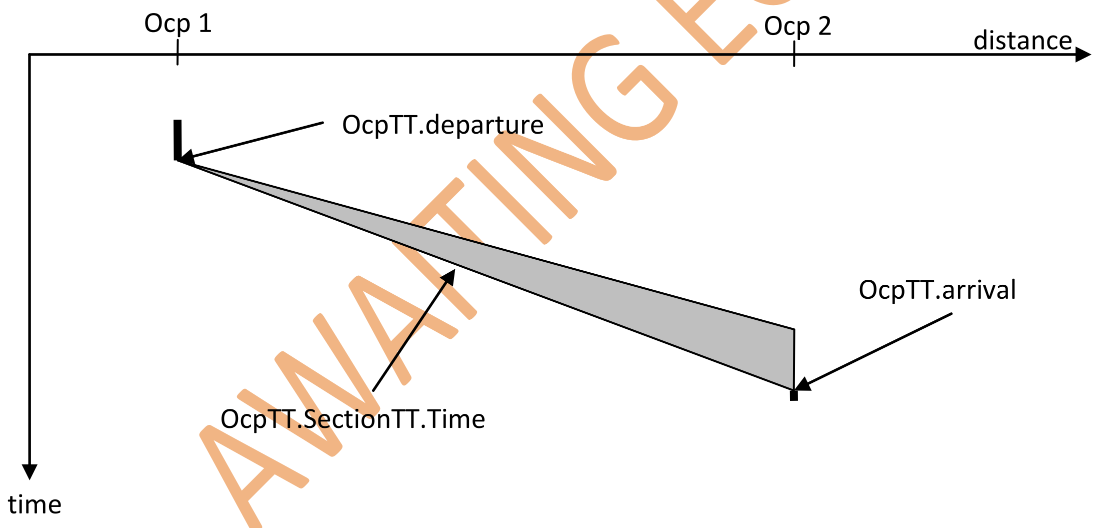


Figure 7.5: Times in RailML2.3

The running times allow specifying a “smooth” train trajectory with higher precision, than only at stations. Using additionalReserve attribute it provides an envelope for undisturbed case. The only use case for this value is a “simplified” forecast algorithm, which calculates possible arrival times of a train in case of delays by reducing the running time until the technical minimum. Unfortunately the real arrival times depend not only on train characteristics, but on the current state of the infrastructure (e.g. temporary speed restrictions) and the traffic situation (neighbour trains). In the future innovative Traffic Management System which includes ATO it is assumed that the minimum running time cannot be used properly by the involved services.

Therefore the modification would be to provide only one planned time-distance line for the train for controlling purposes.

The uncertainty of this line should be modelled by a distribution function, which typically would have the mean value representing the time-distance line (see Figure 7.6).

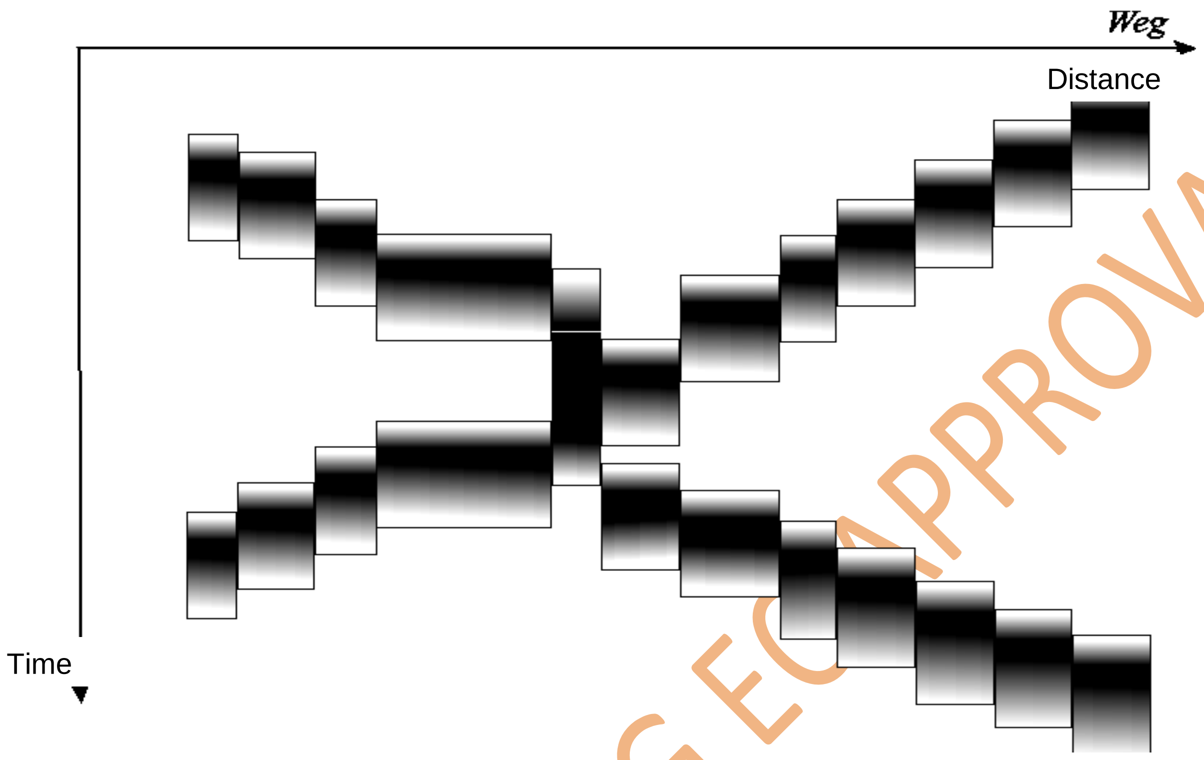


Figure 7.6: Representation of uncertainty in time-distance diagram

But even in this case only the operator's workstation is requiring this information to allow the operator to identify areas with high uncertainty and create solutions with high probability.

The next question is, if the separation of time-data between OcpTT and sectionTT required in TMS. In case of RailML which concentrate on the planning at station level it is a reasonable approach. As soon as in TMS the times and connections can be associated with signals the situation changes:

- way between them is mostly unique and;
- the distance is relatively short as well.

It depends on the selection of class hierarchy how to structure the timing points.

7.2.3 Proposed operational timetable representation

According to the CDM-definition process described in D8.3, the Timetable schema from RailML is used as a starting point for development of the Operational Timetable-part. In the first step a simplified class diagram was be derived (see Figure 7.7).

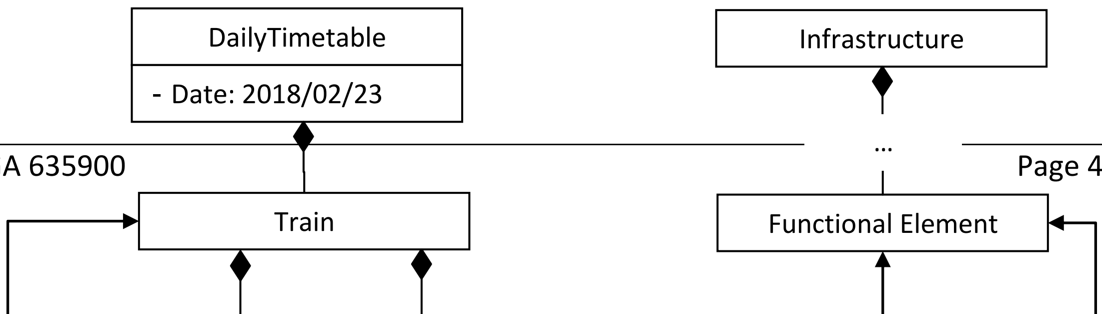


Figure 7.7: First approach for timetable class diagram

In the simplified diagram the separation between contractual conditions (e.g. times for passenger exchange), operational decisions and just forecasted times between them is removed.

In TMS the timetable shall represent two main concepts:

- what and when shall be done by involved staff and systems – operational tasks;
- forecast/plan for the future and past train position in time and space.

The class diagram on Figure 7.8 follows this idea: the train is represented by two sequences - OperationalTasks and TrainPositions. This class diagram has following advantages:

- planned/forecasted time and train positions are located at one place,
- there is a separation of concepts: the operator would modify OperationalTasks and the TrainPositions will be created by the simulation to a large extent,
- compact representation in the client – trains with few operational tasks (start – finish) would not require a null-reference in each TrainPosition (like in OcpTT);
- client applications can locate required data quickly: e.g. PIS is interested in OperationalTasks PassengerExchange and PassengerConnections would find them from the short list of tasks, without a need to search in all TrainPositions (or OcpTTs),
- the number of optional attributes is small – they can be replaced by default value, e.g. TrainPosition.offset = 0, TrainPosition.DwellTime=0.

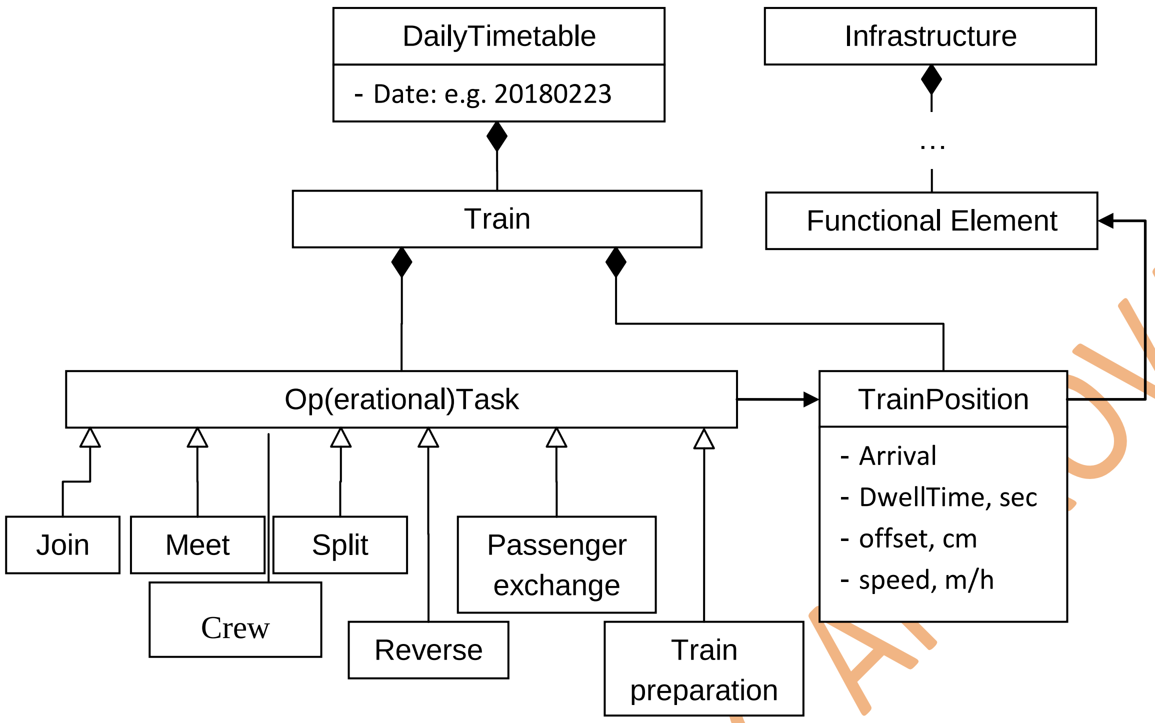


Figure 7.8: Proposed state of the timetable class diagram

The TrainPosition does not have an ID, therefore OperationalTasks would reference them by its index in the sequence. It is very efficient in most cases – access in O(1) time and no IDs has to be transferred. As a backside is the necessity to adjust all indexes in case of "re-routing" causing changes in the number of TrainPosition-Elements in the sequence.

The polymorphism concept is well supported by the Protobuf using oneof-keyword, as well as by JSON and XML serialisation. In the operational tasks additional references can be used, e.g. PassengerExchange can reference the Station and the planned platform.

A forecast algorithm could publish only the TrainPosition-part of the train adjusting the forecast curve each couple of seconds to the new train position reports.

7.2.3.1 Open points in further Shift2Rail projects

The ATO Trackside service would have to identify, which TrainPositions specify the "control values" for the TMS and which are the estimation of the future train behaviour:

- it is task of ATO TS to identify previous trains and send an interval between previous and the next train. If the Interval is bigger than some amount e.g. 5 minutes – skip it. So automatically intervals only on bottlenecks will be sent.

Further open points are representation of the blocking times and overlaps. There are several possibilities to model them:

- interpolation from available TrainPositions;
- additional fields with beginReserve to endReserve.

The decision depends on typical clients requiring blocking information: if the client requiring blocking time would have runtime calculation functionality (e.g. a Timetable Editor), it could interpolate the blocking times itself. A simple client requires explicit pre-calculated blocking times.

7.3 Traffic Control Topics

This chapter defines the processes of communication, and content of topics used for the messaging between:

- an external Traffic Control Application for ETCS L2 and L3 located inside a TMS and the Radio Block Centre (RBC)/Interlocking (IL) in the field and;
- an external Traffic Control Application for Object and Route control located inside a TMS and the ILS in the field.

The scope of functionalities for which the messaging data are defined, shall be limited to support core operational procedures. It is well known that there exist a lot of additional country or client specific operational requirements for additional messaging and or data to be processed. The messaging between TMS and RBC shall support ETCS L2, L3 and Moving Block functionality.

This chapter specifies the methodology of the communication from TMS to field infrastructure via the Integration Layer (IL) applying Subscribe and Publish (S&P) processes.

Subscribers involved in the communication process may have implemented only those functions necessary to secure the messaging necessary to execute their specific operation however they shall be able to respond to TMS following the specified Command, Command Response, Command Reject and Command Acknowledgement procedure.

7.3.1 Functional Architecture based on In Memory Data-Grid technology

The Integration Layer must provide high-availability and scalability by distributing data across multiple machines.

In-memory technology architectures take advantage of low-latency transaction processing. This is a consequence of the fact that the price of RAM is dropping significantly and rapidly and as a result, it has become economical to load the entire operational dataset into memory with performance improvements of over 1000x faster. In-Memory Compute and Data Grids provide the core capabilities of an in-memory architecture.

The figure below shows a possible architecture which includes a “CCS Framework Manager” managing the administration services (start, stop, connect, monitor, deploy and un-deploy, central logging….) of the Interfaces between Integration Layer and Signaling Field Infrastructure.

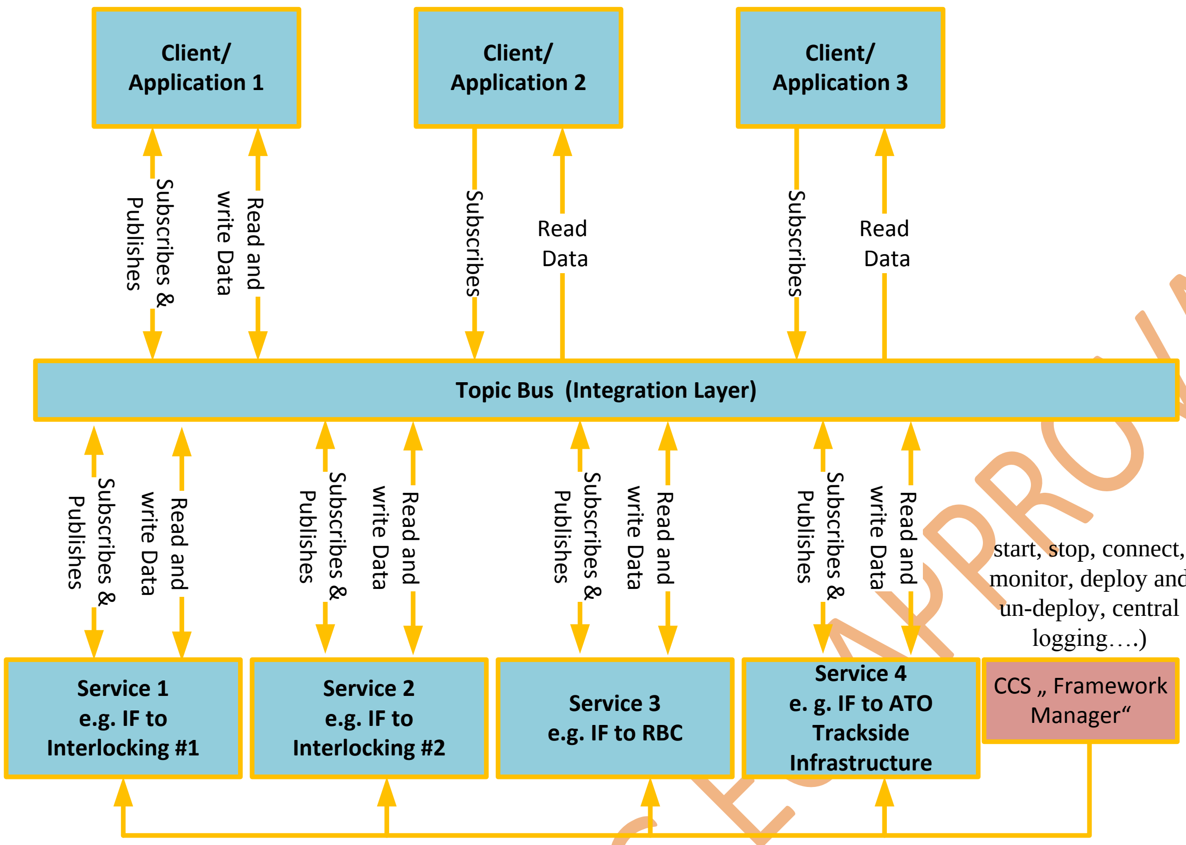

Client/ Application 1
Client/ Application 2
Client/ Application 3
Subscribes & Publishes
Read and write Data
Subscribes
Read Data
Topic Bus (Integration Layer)
start, stop, connect, monitor, deploy and un-deploy, central logging….)
Service 1 e.g. IF to Interlocking #1
Service 2 e.g. IF to Interlocking #2
Service 3 e.g. IF to RBC
Service 4 e. g. IF to ATO Trackside Infrastructure
CCS „ Framework Manager“

Figure 7.9: Architecture based on IMDG Technology

Classic IMDG operation is characterized using key sets where each key belongs to a different application. The external database component is optional. If present, then IMDGs will usually automatically read data from the database or write data to it.

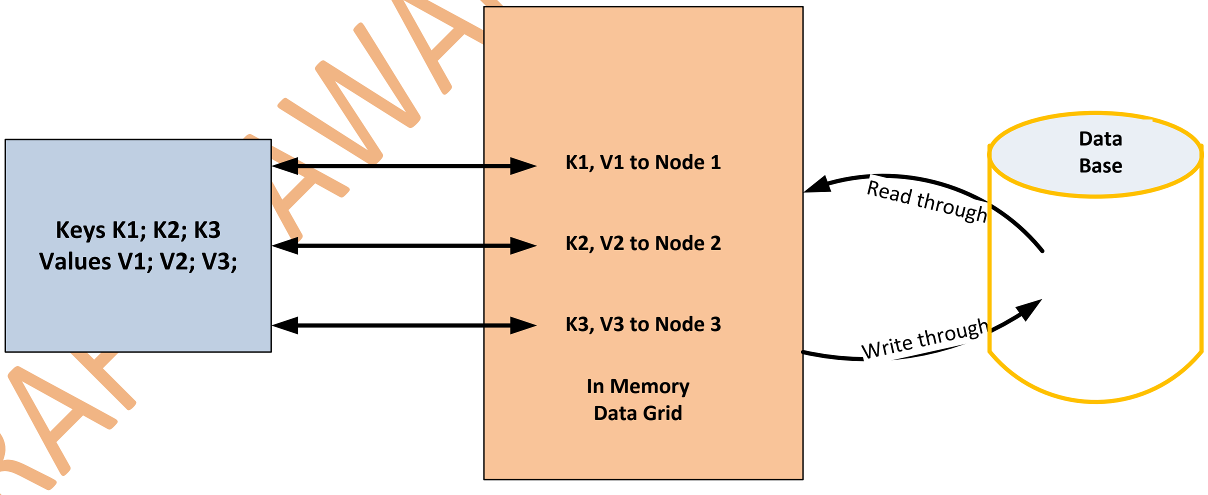


Figure 7.10: Classic IMDG operation with key sets and optional Data Base

7.3.2 Services Management

The generic model to manage the interaction between Integration Layer and Signaling Field Infrastructure is shown in Figure 4.1. The process is similar for Interlocking, RBC and ATO.

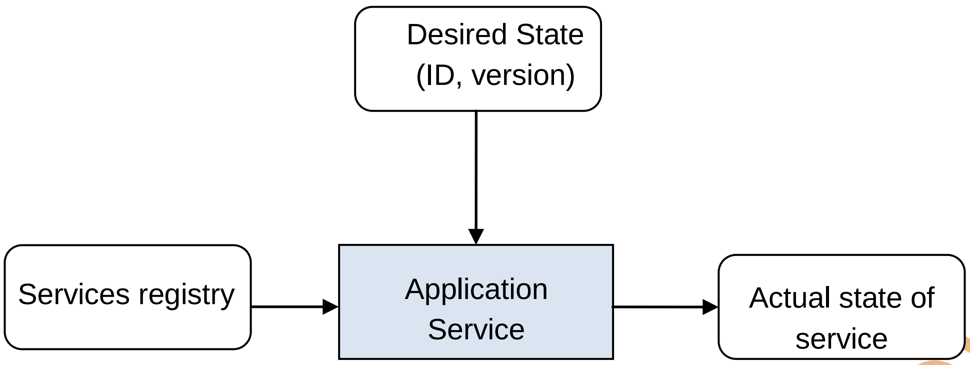
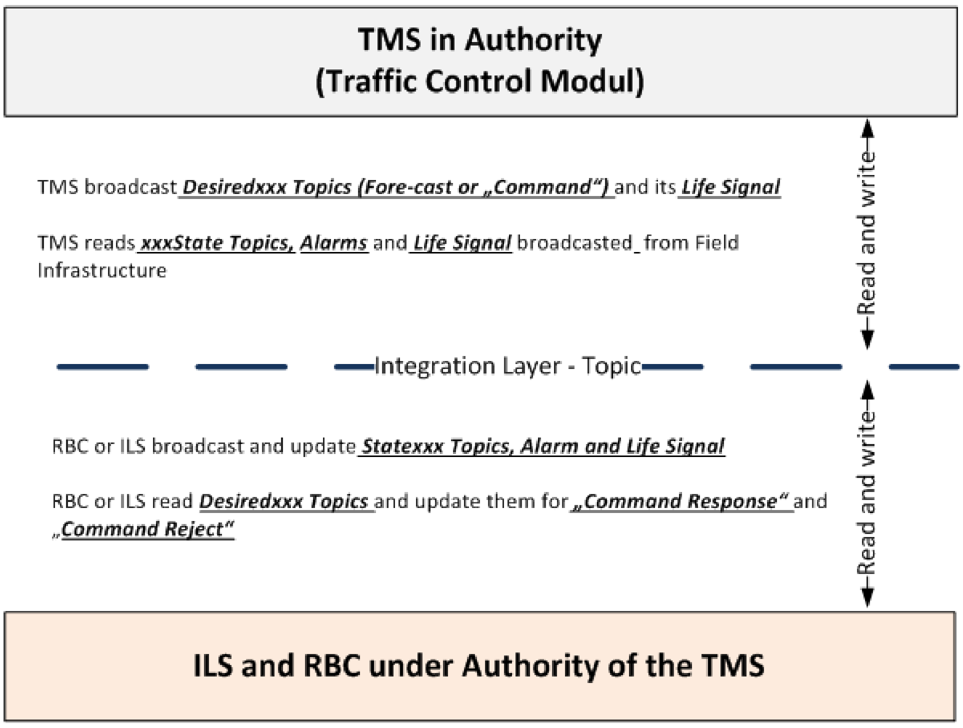


Figure 7.11: Management Process for CCS "Application Framework" for ILS, RBC, ATO

The services/functions needed at the level of the Interfaces to Interlocking, RBC or ATO are managed by an "CCS Framework Manager for start, stop, connect, monitor, deploy and un-deploy, central logging Operations (see also In2Rail WP8 [D8.6] and [D8.7]).

7.3.3 Communication Process TMS - Signalling Infrastructure


TMS in Authority (Traffic Control Modul)
Read and write
Integration Layer - Topic
Read and write
ILS and RBC under Authority of the TMS

TMS broadcast ***Desiredxxx Topics (Fore-cast or „Command")*** and its ***Life Signal***

TMS reads ***xxxState Topics, Alarms*** and ***Life Signal*** broadcasted from Field Infrastructure

RBC or ILS broadcast and update ***Statexxx Topics, Alarm and Life Signal***

RBC or ILS read ***Desiredxxx Topics*** and update them for ***„Command Response"*** and ***„Command Reject"***

Figure 7.12: Communication Processes between TMS and Field Infrastructure

7.3.4 Traffic Control related Topics

The following table lists Topics related to Traffic Control according the scope of In2Rail WP8 and which subscriber and publisher may be engaged.

Topic Name	Subscriber	Publisher

Topic Name	Subscriber	Publisher
Route State	Traffic Management System, Energy Management System, Asset Management System	Interlocking System
Route Desired	Energy Management System, Asset Management System, Interlocking Systems	Traffic Management System,
Object state	Traffic Management System, Energy Management System, Asset Management System	Interlocking System Asset Management System (In specific situations a manual over-write of the status can be done the Operator)
Object Desired	Interlocking Systems	Traffic Management System Asset Management System
Train State	Passenger Information System, Energy Management System, Freight Management System, Asset Management System, Traffic Management System	RBC
TSR State	Traffic Management System, RBC, Asset Management System	RBC
TSR Desired	Traffic Management System, RBC, Asset Management System	Traffic Management System, other sources which can request a TSR e.g. Maintenance Staff
Alarm	Traffic Management System, RBC, Interlocking System,	Traffic Management System, RBC, Interlocking System
Life Sign	Traffic Management System, RBC, Interlocking System,	Traffic Management System, RBC, Interlocking System,

Table 7.1: CCS related Topics and possible Subscribers and Publishers Topic Structure

7.3.5 Life Sign

The “Life Sign” Topic provides the following information which shall be updated periodically.

Source	Name	Description
TMS, RBC, ILS	Client ID	
	Time Stamp	Time of Update/Refresh after specified Time
	Site data version	• Loaded Site data version • Indication that the Client is in process to change

Source	Name	Description
		site data version
	State	• Client is subscribed and broadcast Data to • IL(Publish) • Client is subscribed and broadcast Data to a local installation (Local)
	Authority	• (System) Authority (incl. indication of ID of authorized system) or Local (Authority) • Clients which can request Authority • Client is in "Authority"- hand over process
	Client-Operator	• Operator ID, Name • Attributes representing his responsibility profile e.g. License to manage L3 operation • Operator phone number • Flag representing the request to establish contact with indicated Operator
TMS	Systems under Authority	List of system which are under Authority (Client Ids)
ILS	Objects and Routes under control	ILS is controlling Objects and Routes in the area of Control (true/false)
RBC	Status of Communication RBC – OBU	Defines whether the communication RBC-Train for all Trains in Control is available (true/false)
	Trains under control	RBC is controlling Trains in the area of Control (true/false)

Table 7.2: Topic "Life Sign"

7.3.6Train Control and Train State

7.3.6.1 Train Control Command Topic

The following Train Control Commands shall be possible at minimum:

Name	Description
Unconditional Emergency Stop	Request to stop a train under emergency stop conditions
Revoke Emergency Stop	Request to revoke an emergency stop
Stop Train	Request to stop a train without emergency condition

Continue Trip	Request to continue the service
Position Request	Request to send the position of a train
Set a new Train Number	Request to give a new Train Number to a train e.g. after splitting a train
Split a Train	Request to split a train
Couple a Train	Request to couple two trains train
De-register a Train	Request to de-register a train e.g. after coupling manoeuvre
Confirm Configuration	Request to confirm the configuration of a train
Plain Text Message	Request to send a text message to the train driver.
Emergency Alert	Indication of an emergency alert to the train

7.3.6.2 Train Indication Topic

The following Elements shall be applied to indicate status data of a Train.

Element Name	**Sub-structure**	**Description**
Train Identity	Engine Identity	Onboard ETCS identity
	Train Number	Operational train number
Train Static Data	Train Category	Train categories for which the command applies
	Train Configuration	Description of train consist
	Train Length	Length of train
	Train Max Speed	Indicates the train max speed
	Loading Gauge	Indication of the maximum height and width for railway vehicles and their loads to ensure safe passage through bridges, tunnels and other structures
	Axle Load	The axle load defines the total weight for all wheels connected to a given axle
	Air Tight	Defines whether cars or waggons are air tight to allow for higher speed in Tunnels
	Traction Power	Defines the traction power that can be used by a train
	STM Identity	Identifies which National Class B ATP systems are available on the train
Train Dynamic Data	Train Speed	Actual speed of the train
	Train Position	Actual position of the train
	Doubt Over	Difference between upper bound of the confidence interval and the estimated train position
	Doubt Under	Difference between lower bound of the confidence interval and the estimated train position.
	Train Mode	Identification of Operational Mode of the train e.g. Train in full Supervision, On Sight…… ATO, other modes

Element Name	Sub-structure	Description
	ETCS Level	Indicates current ETCS level for a train
	Radio Communication	Indicates state of communication between RBC and the train e. g. normal, terminated
	TIMS	Status of Train Integrity • No train integrity information available • Train integrity confirmed by integrity monitoring device
	Action Required	Train requests Central System operator action e. g. approximate position required
	Request Route	Indicates whether the train requests route or not
	End of MA	Indicates current end of Movement Authority of the train
	Authorisation	Indicates if train has been given a Movement Authority e.g. No authority; Movement Authority; SR Authority; Shunting Authority; Reversing Authority ……
	MA Over RBC Border	MA which exceeds border of RBC
	Train Transfer State	Indicates whether the train is changing RBC area
	Train State	Indicates a special status of the train in the RBC when applicable. Omitted when train is in normal operational state e. g. Removed meaning Train has left RBC area and is deleted
	Emergency Stop	Indicates whether the train is emergency stopped or not
	SiB Front Offset	Distance from the SiB ahead to the front of the train.
	SiB in Path	Identity of a Signal Board which is included in the current path for the train
	Balise ID	Balise ID
Text Message from TMS to Train via RBC.	Message Text	Message to send to the train driver
	Message Acknowledgement	Indicates whether the Driver/OBU has acknowledged the text message or not.

7.3.7TSR State and Desired TSR

7.3.7.1 TSR Management

A TSR is activated and deactivated and therefore all TSRs are stored in a TSR register, both at the RBC and in the TMS.

The following types of TSRs are part of In2Rail scope:

- Route Map TSR TSR is based on specific conditions of a route or route section. A Route Map TSR may not be de-activated from the Operator by simple procedure as it represents a permanent condition for a section of the track

- Maintenance TSR — TSR is established to execute Maintenance work
- Dispatcher TSR — Conditional TSR set by the dispatcher

The Operator designs a TSR in a sandbox and execute all mandatory steps and communication to obtain the agreement and approval for the TSR. The TSR is then send to the RBC to be stored in the TSR Register. All TSRs send from the TMS to the RBC have the status “deactivated”. The RBC updates Topic indicating registered TSRs.

All TSRs are broadcasted in the TSR State Topic.

7.3.7.2 TSR State Topic

The TSR State Topics carries the information of all registered and approved TSRs. The state information is updated from the RBC. The Indication for a TSR shall have the following information as the minimum.

Name	Description
TSR Identity	Identity of the TSR
TSR Type	TSR Type: Route Map TSR Maintenance TSR Dispatcher TSR
TSR activated/de-activated	State of the TSR (True/False)
TSR Speed	Requested speed value in a TSR or TSA area
TSR Line	Definition of the geographical position where the TSR starts, geographical position where the TSR ends and direction how to drive through the start and the end position of the TSR. (instead of geographical positions Objects can represent the start and the end of a TSR)
Direction	Direction of travel, Nominal/Reverse is defined for all tracks. This parameter defines in which direction the TSR is valid
Train Category	Variable which defines for which Train categories the TSR is valid
TSR Status	Indicates whether TSR is active or has been deactivated
Deactivation Possible	Indicates that a route map defined TSR can be deactivated from an Operator
Approval	Indicates approvals provided for a specific TSR
Comment	Text illustrating specific characteristics of the TSR

Table 7.3: Data embedded in TSR State Topic

7.3.7.3 Desired TSR Topic

The Desired TSR Topic carries the information of all TSRs for which state change is requested. TSRs under negotiation are not visible on this topic (sandbox).

The information is broadcasted from the TMS or other sources which can request a state change of a TSR.

In addition to the described information to be embedded in the TSR State Topic the Desired TSR Topic shall allow for the following "Desired States" (Commands) at minimum:

Name	Description
Store TSR	TMS sends request to RBC to store a TSR into the register
Delete TSR	TMS sends request to RBC to delete a TSR from the register
Activate TSR (see TSR State)	TMS sends request to RBC to activate a TSR
De-activate TSR (see TSR State)	TMS sends request to RBC to de-activate a TSR
TSR Request	A Client sends a request to TMS/Operator to define a TSR. The request must contain a minimum set of data see 8.4

Table 7.4: Additional Data (Commands) embedded in Desired TSR Topic

7.3.7.4 TSR Request Indication

The Client sends a request to define a temporary speed restriction. If the indication "Bi-directional" is set to False then the validity direction of the TSR is the same as defined by a directed start position, a directed end position and all tracks included in the extent (including the tracks the start and end position belongs to).

For a TSR request the following table shows the data which are required at minimum.

Name	Description
Start Position	Position where TSR starts
End Position	Position where TSR ends
Track List	List of all tracks included
Speed	Required Speed limit]
Train Length Delay	Delay between deactivation or activation for a long train to enter or to leave the TSR area
Bi-directional TSR	Indicator whether the TSR is for both directions (True/False)
Time Frame	Requested period for the TSR
Reason	Reason why the TSR is requested

Table 7.5: Additional Data required for TSR Request

7.3.8Object State and Desired Object

The scope of In2Rail comprises the following Objects:

- Point (Machine);
- Signals;
- Level Crossing;
- Track Circuit;
- Axle Counter.

The data structure for other Objects will be developed under the S2R Program in X2RAIL-2 WP6 and Impact-2 WP7.

For the Objects the Data representing Object State and Desired Object (State) are defined. In addition, available data representing state of degradation of the Object are introduced.

A concept defining which data can be made available to indicate asset status is under development of In2Rail WP9 and S2R IP3 projects and the data model and structure will be updated when results are available. For the time being a first set of data is considered to be integrated in the data structure.

7.3.8.1 Object State Topic

The Object State Topic carries the information of all Objects.

The state information is updated from the Interlocking. For specific cases a manual over-write of the status issued from the Operator is possible.

7.3.8.1.1 Point (Machine)

The Indication for a point consist out of signalling related status information and data representing the current asset status to be used for Traffic forecasting and maintenance processes and shall have the following information as the minimum:

Name	Description
Object ID	Identity of the Object
Object Position	Geographical position of the Object
Object Type	Simple variation to left Simple variation to right Triple switch Crossing without change of direction Crossing with Double junction Special version
Max. Speed	Maximum allowed speed to pass the object
Max Axle Load	Maximum allowed axle load to pass the object
Normal	Object is in operation
Unknown	Object is out of control e.g. under manual operation
Left	Object is in left position
Right	Object is in right position
Moving	Object is moving
Clamped Left	Object is clamped in left position
Clamped Right	Object is clamped in right position
Availability prognosis	Indication on availability fore-cast for the Object
Maintenance	Object is under maintenance

Table 7.6: Indication for Point Machines

7.3.8.1.2 Signals

The indication for Signals shall have minimum the following data:

Name	Description
Object Identity	Identity of the Object
Proceed	Proceed state
Stop	Stop state
Speed Indication	Indication of Line speed
Availability prognosis	Indication on availability fore-cast for the Object
Maintenance	Signal is under maintenance (True/False)

Table 7.7: Indication for Signals

7.3.8.1.3 Level Crossing

The indication for Level Crossings shall have minimum the following data.

Name	Description
Object Identity	Identity of the Object
LX Blocked	Level Crossings is Blocked (True/False)
LX Local	Level Crossings is in "Local" Operation (True/False)
Communication to LX	Communication to LX available (True/False)
LX status unknown	Level Crossings Status is Unknown (True/False)
LX status open	Barriers are open (True/False)
LX status closed	Barriers are closed (True/False)
LX status moving	Barriers are moving (True/False)
LX status inhibited	Level Crossings Operation is Inhibited (True/False)
Availability prognosis	Indication on availability fore-cast for the Object
Maintenance	Object is under maintenance (True/False)

Table 7.8: Indication for Level Crossings

7.3.8.1.4 Track Circuits

The indication for Track Circuits shall have minimum the following data.

Name	Description
Object Identity	Identity of the Object
Unknown	Track Circuit is out of control.
Occupied	Track Circuit is occupied
Free	Track Circuit is free
Availability prognosis	Indication on availability fore-cast for the Object
Maintenance	Track Circuit is under maintenance (True/False)

Table 7.9: Indication for Track Circuits

7.3.8.1.5 Axle Counter

The indication for Axle Counters shall have minimum the following data.

Name	Description
Object Identity	Identity of the Object
Unknown	Axle Counter is out of control.
Active	Axle Counter is active
In-Active	Axle Counter is in-active
Availability prognosis	Indication on availability fore-cast for the Axle Counter
Maintenance	Axle Counter is under maintenance (True/False)

Table 7.10: Indication for Axle Counters

7.3.9Desired Object Topic

The Desired Object Topic carries the information of all Objects for which state change is requested.

The information is broadcasted from the TMS or other sources which can request a state change of an object.

In addition to the described information to be embedded in the Object State Topic the Desired Object Topic shall allow for the following "Desired States" (Commands) at minimum:

7.3.9.1 Point (Machine)

For Point Control the following "Desired States" (commands) shall be implemented at the minimum:

Name	Description
Left	Move to Left
Right	Move to Right
Block Point	Block the point
Unblock Point	Unblock the point
Revoke Trailed	Revoke Trailed status of point
Maintenance requested	Maintenance activities for this object are requested

Table 7.11: Desired States (Commands) for Point Machines

Note: Depending on the type of operation required there may be more components to be added.

7.3.9.1.1 Signals

For Signals the following "Desired States" (commands) shall be possible at the minimum:

Name	Description
Set Proceed state	Set Signal to Proceed State
Set Stop state	Set Signal to Stop State
Set Speed Indication	Set Indication for following Line speed (e. g. TSR)
Maintenance requested	Maintenance activities for this object are requested

Table 7.12: Desired States (Commands) for Signal

Note: Depending on the type of operation required there may be more components to be added.

7.3.9.2 Level Crossing

For Level Crossing the following "Desired States" (commands) shall be possible at the minimum:

Name	Description
Open	Open Barriers (Command from TMS)
Close	Close barriers (Command from TMS)
Local	Switch Level Crossing to Local Operation (Command from TMS)
Central	Switch Level Crossing to central /System) operation
Block	Block the Level Crossing (Command from TMS)
Unblock	Un-Block the Level Crossing (Command from TMS)

Inhibit	Command from Operator To ILS: Ignore status from LX object
No-Inhibit	Command from Operator to ILS : Consider status from LX object
Maintenance requested	Maintenance activities for this Level Crossing are requested

Table 7.13: Desired States (Commands) for Level Crossing

Note: Depending on the type of operation required there may be more components to be added.

7.3.9.3 Track Circuits

For Track Circuits the following "Desired States" (commands) shall be possible at the minimum:

Name	Description
On	Switch Track Circuit to active
Off	Switch Track Circuit to in-active
Inhibit	Command from Operator To ILS: Ignore status from Track Circuit
No-Inhibit	Command from Operator to ILS: Consider status from Track Circuit
Maintenance requested	Maintenance activities for this Track Circuit are requested

Table 7.14: Indication for Track Circuit

7.3.9.4 Axle Counter

For Axle Counter the following "Desired States" (commands) shall be possible at the minimum:

Name	Description
On	Switch Axle Counter to active
Off	Switch Axle Counter to in-active
Set to Zero	Command from Operator To ILS: Set count from Axle Counter (Over-write Command from TMS after count shows still axle is in the section but section has been controlled on site by staff and no Axle is in the section)
Inhibit	Command from Operator To ILS: Ignore status from Track Circuit
No-Inhibit	Command from Operator to ILS: Consider status from Track Circuit
Maintenance requested	Maintenance activities for this Track Circuit are requested

Table 7.15: Indication for Axle Counter

7.3.10 Route State and Route Desired

7.3.10.1 Complex "Object" (Route) Topic Definition

The design of Route State Topic and Desired Route Topic requires the specification different Elements needed to describe the current and desired state. The description of the Elements of a Route is structured in different individual sections described below.

It is further assumed that the Interlocking or an Automatic Route Setting Device (ARS) is evaluating the different possible routes for a train service and presenting the result to an operator for decision or sets the route according implemented principles.

For ATO purposes the granularity of the Element “Path” may be increased to meet the requirements of “virtual” multiple borders with individual ETA.

A Path representing a route is kept active in the Route State Topic until the train has reached its final destination.

Movement Authority allowing the train to proceed along the path is given according applied signaling rules and available track sections belonging to the set path (Route).

7.3.10.1.1 Route Identification

The Route Identification Element is characterized at least by the following attributes:

Name	Description
Route ID	Identifier for the selected the selected Route
Route Type	• Normal • Staff responsible (set sequential by the operator) • Reverse Route (indicates that path in the reverse direction will be different to normal route) Shunting Route (Route set to execute shunting operation under specific conditions)
Start Object and/or Start Position of Route	Defines the start of route (see 10.1.2)
End Object and/or End Position of Route	Defines the end of route(see 10.1.2)
Scheduled ETA at defined location	ETA for specific positions (stops). This Element can contain several Position/ETA pairs.

Table 7.16: Route Identification

7.3.10.1.2 Position

A Position is described by its Track Identity.

Name	Description
Position	Position is defined by Track Identity

Table 7.17: Position

7.3.10.1.3 Directed Position

A Directed Position is determined by a position and a direction in relation to the nominal direction of the track the position belongs to.

Name	Description
Position	Position is defined by Track Identity (see 10.2.1)
Direction	Direction is defined as directional parameter using nominal track direction as a reference with the following attributes: • Unknown, • Nominal Direction, • Reverse Direction, • Both Directions.

Table 7.18: Directed Position

7.3.10.1.4 Object

An Object is described by its Object Identity.

Name	Description
Object	Position is defined by Object Identity

Table 7.19: Object

7.3.10.1.5 Extent

Extent the sum of all track parts from Start Position to End Position and is defined by a directed start position, a directed end position and list of all tracks included in the extent (including the tracks the start and end position belongs to).

Name	Description
Start Position	= Directed Position (see 9.3.10.1.3)
End Position	= Directed Position (see 9.3.10.1.3)
Track List	All tracks between Start and End Position listed with their Track Identity

Table 7.20: Extent

7.3.10.1.6 Area

An area is a specified territory of the network characterized by extent(s) and borders. Borders are Directed Positions in the area.

Name	Description
Extent	= Extent (see 10.2.3)
Border	= Directed Position (see 10.2.2)

Table 7.21: Area

7.3.10.1.7 Path

Defines the path (sequence of Objects, Tacks) the train shall follow the time when it shall arrive and characterizes the current End of MA.

Name	Description
Extent	= Extent (see 10.2.3)
PathTarget	Directed Position (see 10.2.2)
Path Target Time	ETA calculated for Path Target
Path Target Type	The Path Target Type (position) shall have at least the following attributes: • Destination (Final); • Path Conflict Reached = First occurrence in the path where the path conflicts with another one; • Path Conflict Cleared = First occurrence in the path where the conflicting paths diverge again; • Route Locking Conflict = First occurrence in the path where the route locking is / will be stopped.

Table 7.22: Path

7.3.10.1.8 Location

For specific operations a “Location” can be defined and managed. The Route state topic includes an Element to describe a location and its state. In the Desire Route Topic, the commands to switch state of a location are listed.

Name	Description
Location (Object)Identity	Identity of the Location (Object)
Location Operation	Identifier which Operation is executed at the location
Priority of the Operation executed	Identifier for Piority of the executed Operation
Blocked	Location is blocked for Train Operation (True/False)
Blocked for Destination	Location is blocked for Train Operation in direction of a specific destination (True/False)
Under Control of ILS	Location is controlled from an Interlocking (True/False)
Maintenance Status	The following attributes shall be available at the minimum: • Location is in Operation; • Location is requested for Maintenance Operation; • Location is in Maintenance process.

Table 7.23: Location

7.3.10.2 Operation Variables

The Topic "Route Desired" includes the operational attributes (commands) to operate a section. The commands listed in the table below represent a minimum which may be increased depending on specific IM requirements.

7.3.10.2.1 Variables (Commands) for Section Operation

To operate a section a set of variables (commands) is needed.

Name	Description
Track Section	ID of Track Section (Object ID)
Set Block	Command to et Section to Blocked state
Release Block	Command to release Section Block
Set Restriction	Command to set a predefined Section Restriction
Remove Restriction	Command to release a Section Restriction
Reset Axle Counter	Command to reset Aixle Counter of the section
Reset Track Circuit	Command to reset Track Circuits of the section

Table 7.24: Variables for section operation

Note: Depending on the type of operation required there may be more components to be added to a border specification.

7.3.10.2.2 Variables (Commands) for Route Management

Name	Description
Route (Object) ID	ID of Route
Set Route	Command to set a conventional train route
Release Route	Command to release a conventional train route
Set Cooperative Train Route	Command to set a ETCS train route
Release Cooperative Route	Command to release an ETCS train route
Set Shunting Route	Command to set a shunting route

Release Shunting Route	Command to release a shunting route

Table 7.25: Variables for Route Management

7.3.10.2.3 Area Operation

In the context of this document an Area is an artificially created "region" for the execution of specific operations e.g. Joining Area, Splitting Area…. or just to bundle several individual activities (commands) into one process. Areas are predefined and are activated or deactivated. Areas are registered in the TMS and the Interlocking.

Name	Description
Activate Area	Command to activate an Area
De-activate Area	Command to de-activate an Area
Delete Area	Command to delete an Area from the register

Table 7.26: Area Operation

7.3.11 Alarm Topic

Alarms are generated from ILS and RBC and shall have as minimum following parameter:

Name	Description
Alarm Source ID	ID the source of an alarm sent from RBC or ILS
Alarm Acknowledged	Specifies the status of an alarm sent from RBC or ILS (True/False)
Alarm Group	This variable identifies a group of alarm codes. The values are platform dependent.
Alarm Code	Specifies the alarm code within an alarm group. The values are site and system dependent.
Alarm Parameters	Defines the Parameter and the values of the parameter of an Alarm
Alarm Acknowledged by	System/Operator´s ID who has acknowledged the Alarm

Table 7.27: Data on Alarm Topic

7.4 Energy Management Topics

This chapter gives a brief introduction into the Topics needed for Energy Management. For more detailed description please see [D10.4 – TMS/MMS Interface Specification]. The following Energy Management information shall be possible to communicate at minimum:

Name	Description
Vehicle power limits and expected power usage	Describing the impact of temporary limitations of power availability for electrical motor units. TMS Topic Online Production Plan with its data structure timetable, train parts and related schedule sections.
ETS Component Status	Providing status of Electric Traction power System components (e.g., power switches, catenary sections)
Temporary resource restriction "Feeding section is not energized"	Temporary resource restriction "Feeding section is not energized"
Component disruption	Disruption information "Outage of a major component <x> of

Name	Description
information	substation <y>”
Component status change information	Changed status information of components “status of component <x> is (secure on, secure off, secure in-between, unknown)”

Table 7.28: Information on Energy Topic

7.4.1 Other Topics involved

Electric Traction power Systems, especially the Multi Train Simulator modules require the following topics:

TMS:

- Network Infrastructure (especially Functional Assets);
- Vehicle Characteristics;
- Temporary Access Restriction;
- Online Production Plan;
- Forecast.

Maintenance Management:

- Operated Maintenance.

7.5 External Services (WEB-IF) Topics

In the following, the IL-Topics and involved data structures are described for a selection of interfaces communicating to external services or applications, typically using HTTP based protocols.

7.5.1 Weather forecast/report services

Today, there are a couple of open data initiatives around Web based supply of updated weather information as, e.g., *opendata.dwd.de* (provided by Deutscher Wetterdienst, Germany) or *https://www.yr.no/* (provided by Norwegian Meteorological Institute together with the Norwegian Broadcasting Corporation).

TMS Interfaces publishing relevant IL topics from these services have to support the different protocols according to the terms and conditions for accessing them as being available from the respective Web sites.

Weather reports about the current status and forecast of weather as related to geography or network infrastructure in the IL is available through the WeatherCondition data structure which is referencing the respective locations and time information.

Note that based on this information, TMS business logic generates and updates specific types of Temporary Access Restrictions impacting forecast calculation and conflict detection of the TMS. Examples are temporary speed restrictions or line/bridge closures due to critical wind speeds or snowfall. Other examples are reduced traction force or wheel/track adhesion due to solar irradiation, temperature, humidity.

7.5.1.1 Data Structure WeatherCondition

According to the Assets data structure of the Network Infrastructure, the *Location Reference* for weather condition information can be modelled either as AreaLocation, LinearLocation or SpotLocation depending on the characteristics of the measured information. Usually, the available open data services provide grid based information with certain resolution i.e., distance between grid points. However, the other location types might be of relevance for future data services or not yet assessed ones. Besides the location information, also the time is of interest, since one can have reported and forecasted information. Consequentially, the time as provided with the external weather service is used by the interface in order to feed the *Time Reference* information. This data structure carries different parameters included in sub-structures. It is a container and shall contain as a minimum following parameter:

Name	Description
WeatherConditionWind	Condition for wind (See 9.5.1.1.1)
WeatherConditionHumidity	Condition for humidity (See 9.5.1.1.2)
WeatherConditionTemperature	Condition for Temperature (See 9.5.1.1.3)
WeatherConditionAux	Condition for auxiliary (See 9.5.1.1.4)

Table 7.29: Weather Condition

The sub-structures of WeatherCondition and the information they carry are described in more detail in the next sections.

7.5.1.1.1 Data Structure WeatherConditionWind

Following wind related parameters shall be communicated at minimum:

Name	Description
Wind Direction	The direction of the wind
Max Wind Speed	Decimal (m/s)
Meridional wind speed	Decimal (m/s)
Zonal wind speed	Decimal (m/s)

Table 7.30: Information on Weather Condition Wind

7.5.1.1.2 Data Structure WeatherConditionHumidity

Following humidity related parameters shall be communicated at minimum:

Name	Description
Relative humidity	Relative humidity 2m above ground
Surface specific humidity	e.g., steel

Table 7.31: Information on Weather Condition Humidity

7.5.1.1.3 Data Structure WeatherConditionTemperature

Following temperature related parameters shall be communicated at minimum:

Name	Description
Air temperature	Decimal (degree centigrade)

Ground temperature	Decimal (degree centigrade)

Table 7.32: Information on Weather Condition Temperature

7.5.1.1.4 Data Structure WeatherConditionAux

Following Auxiliary Weather Condition related parameters shall be communicated at minimum:

Name	Description
Precipitation	Precipitation in mm Rain
Soil moisture	Decimal (percent)
Snow depth	Decimal (m)
Snow density	Decimal (g/cm^3)
Solar irradiation	Decimal (W/m^2)

Table 7.33: Information on Weather Condition Aux

7.5.2 Dynamic Demand services

The currently planned assignment of rolling stock units and staff/crew members to train runs is communicated using the Offline and Online Timetable topics. In order to enable communication of options regarding these assignments in the view of RU's resource planning systems, additional data structures have to be considered.

7.5.2.1 Data Structure DDAlternativeResourceLinkStaff

With this data structure, alternative resource links for staff as assigned to train runs within the production plan are communicated. The links are assigned at the locations (operational control points) where the respective on- or off-boarding operations are performed, i.e., at start, destination and crew/staff exchange stations or stop points. For the Alternative Resource Link Staff the following parameters should be communicated as a minimum:

Name	Description
Train ID	The links are assigned to a train run using the identification of the train *TrainID* and the related operational control point *TrainOCPRef* (see next line)
TrainOCPRef	Related operational control point (see above Train ID description)
StaffName	Name of the staff for identifying the staff resource
StaffPhone	Phone number of the staff for identifying the staff resource
StaffType	Type of the staff for identifying the staff resource
ResourceAvailabilityTimeFrom	Constraints on earliest and latest time of availability of a staff resource at the location
ResourceAvailabilityTimeTo	Constraints on earliest and latest time of availability of a staff resource at the location
MinHandlingTime	For addressing timing needs, minimum and maximum handling times are used
MaxHandlingTime	For addressing timing needs, minimum and maximum handling times are used
LinkType	The data element *LinkType* is of Boolean data type indicating on- or off-boarding

Table 7.34: Information on Alternative Resources Link Staff

7.5.2.2 Data Structure DDAlternativeResourceLinkRollingStock

With this data structure, alternative resource links for rolling stock as assigned to train runs within the production plan are communicated. The links are assigned at the locations (operational control points) where the respective attaching or detaching operations are performed, i.e., at start, destination and material exchange stations or stop points. For the Alternative Resource Link Rolling Stock the following parameters should be communicated as a minimum:

Name	Description
Train ID	The links are assigned to a train run using the identification of the train *TrainID* and the related operational control point *TrainOCPRef* (see next line)
TrainOCPRef	Related operational control point (see above Train ID description)
VehicleRef	For identifying the rolling stock resource and using vehicle details, the reference link to the vehicle information on the IL is required
ResourceAvailabilityTimeFrom	Constraints on earliest and latest time of availability of a rolling stock resource at the location
ResourceAvailabilityTimeTo	Constraints on earliest and latest time of availability of a rolling stock resource at the location
MinHandlingTime	For addressing timing needs, minimum and maximum handling times are used
MaxHandlingTime	For addressing timing needs, minimum and maximum handling times are used
LinkType	The data element *LinkType* is of Boolean data type indicating attachment or detachment of rolling stock

Table 7.35: Information on Alternative Resources Link Rolling Stock

7.5.2.3 Data Structure DDAlternativeTT

This data structure is used for communication of alternative RU/passenger requirements for trains regarding timetable locations and related times as currently negotiated and reflected by the Offline and Online Timetable structure of the IL. For the Alternative Timetable the following parameters should be communicated as a minimum:

Name	Description
Train ID	For identification of the train and related operational control point, data elements *TrainID* for and *TrainOCPRef* are available (see next line also)
TrainOCPRef	Related operational control point (see above Train ID description)
AltEarliestArrivalTime	In order to transfer alternative earliest/latest times of arrival at a departing/destination station *AltEarliestArrivalTime* and *AltLatestArrivalTime* are used in contrast to the currently negotiated arrival times. Please note that in case of optional changes in timing at departure station e.g., regarding loading processes before departure, there can be alternative earliest/latest arrival times (i.e., time making the train available at unloading

Name	Description
	track) for the train communicated with this structure.
AltLatestArrivalTime	See description *AltEarliestArrivalTime*
AltStopInfo	Used for communicating on-demand stop or cancellation of stop e.g., if there is no real interest of stopping anymore
AltOCPRef	An alternative departure, destination or intermediate handling location for freight and passenger can be communicated by making use of the *AltOCPRef* element. At the same time, earliest/latest times of arrival at that location may be used with *AltEarliestArrivalTime* and *AltLatestArrivalTime* as described above.

Table 7.36: Information on Alternative Timetable

7.5.2.4 Data Structure DDAvailabilityConstraintTrack

Availability constraints of track resources are communicated to RUs based on a topic *TemporaryAccessRestriction*. From this structure, maximum speed (0=blocking), maximum total load, maximum axle load, maximum gauge, energy and signalling system restrictions can be retrieved and forwarded to RUs. It is used primarily, to let RUs consider appropriate options for alternatives in geography and timing of train services as being communicated by previous data structures.

7.5.2.5 Data Structure DDAvailabilityConstraintRollingStock

In order to let RUs transfer information about availability constraints of rolling stock resources to IM, different data elements are available:

Name	Description
ReducedMaxSpeed	Indicates a reduced maximum speed for the vehicle which has been observed either by traction engine drivers or by available on-board systems
ReducedMaxTractionForce	Represents a temporarily reduced maximum traction force of a vehicle
RestrictionSigSystem	Enumeration type referring to situations where, due to some OBU component failure, specific type(s) of signalling systems can temporarily not be supported
RestrictionTCSystem	Enumeration type related to temporary train control system restrictions as indicated by the OBU of a vehicle
RestrictionBrakeSystem	Enumeration type related to temporary braking system restrictions as observed by staff or indicated by OBU equipment of a vehicle
VehicleRef	For identifying the vehicle itself, the Vehicle ID *VehicleRef* is used which also acts as reference link to the vehicle base information on the IL

Table 7.37: Information on Availability Constraint Rolling Stock

All values above act as an overlay to the vehicle’s technical base data as being available from the IL as well. The value “-1” is used for removing the overlay to revert back to the value of technical base data, i.e., re-setting to the technical default value. “0” is used to indicate that there is no temporary change.

7.5.2.6 Data Structure DDPassengerCount

Today's trip or journey planning mobile applications (Apps) are providing information about the individual booking details and may also supply TMS with changes in travel priorities in the future e.g., if train faces significant delay. Also central ticket booking or seat reservation systems are technically able to supply details about how many people are expected to be on which train heading for what destination. Knowing this, future TMS will be able to consider "costs" of alternative travel plans for passengers sitting in a train within disturbed situations in order to support object functions of optimizers or re-scheduling modules.

For communication of numbers of passengers within the train as per destination station, we have to be aware that these figures may change at any stop for passenger (dis-) embarking. As a consequence, this data structure contains a number of sub-structures of type *DDPassengerCountStation* reflecting the passenger numbers for each stop. Please be aware that passengers tend to apply individual decision making in order to re-plan their journey dynamically so that these figures might be sometimes representing estimations rather than water proof figures. The following parameters are communicated for Passenger Count:

Name	Description
DDPassengerCountStation	Reflecting the passenger numbers for each stop
Train ID	For identification of the train

Table 7.38: Information for Passenger Count

7.5.2.7 Data Structure DDPassengerCountStation

The following parameters are communicated for Passenger Count Station:

Name	Description
TrainOCPRef	Referring to a commercial stop for passenger (dis-) embarking
DDPassengerCountStation Dest	Number of sub-structures of type *DDPassengerCountStationDest* where passenger numbers (per destination station) sitting in the trains between this commercial stop and the next can be found

Table 7.39: Information for Passenger Count Station

7.5.2.8 Data Structure DDPassengerCountStationDest

The following parameters are communicated for Passenger Count Station Destination:

Name	**Description**
OCPRef	For identifying the destination station
PassengerCountFirstClass	Representing the number of passengers heading for this destination which itself is not necessarily part of the scheduled train run
PassengerCountSecondClass	Representing the number of passengers heading for this destination which itself is not necessarily part of the scheduled train run

Table 7.40: Information on Passenger Count Station Destination

7.5.3 Passenger information services

For information about numbers of passengers within a train with their assigned destination, please refer to the respective Dynamic Demand data structures.

The following parameters are communicated for Passenger information services as a minimum:

Name	Description
PassengerCountReservation	Described in section 9.5.3.1
PassengerCountDetected	Described in section 9.5.3.4
PassengerTimetableUpdate	Described in section 9.5.3.7
PassengerTrainPositions	Data structure should look like "forecast" data structure
PassengerTransferConnectionStatus	Status of passenger transfer connection, e.g. status "at risk" and "broken"

Table 7.41: Information on Passenger information services

7.5.3.1 Data Structure PassengerCountReservation

The number of passengers who reserved seats in specific coaches within 1st or 2nd class are available from this data structure. Please note that we decided to show the passenger numbers against their planned destinations within the Dynamic Demand data structures described in the previous sections. This structure represents the absolute figures given by seat reservation systems as per rolling stock unit (seat reservation systems). It contains one or more sub-structures of type *PassengerCountReservationStation* where the effective number of seat reservations can be found.

7.5.3.2 Data Structure PassengerCountReservationStation

This sub-structure contains a *TrainOCPRef* referring to a commercial stop for passenger (dis-) embarking and one or more sub-structures of type *PassengerCountReservationStationUnit*.

7.5.3.3 Data Structure PassengerCountReservationStationUnit

This sub-structure contains a *VehicleRef* referring to the rolling stock unit and data elements *PassengerCountReservationFirstClass* and *PassengerCountReservationSecondClass* representing the number of seat reservations within the two classes.

7.5.3.4 Data Structure PassengerCountDetected

Numbers of passengers within a rolling stock unit can also be detected or estimated by technical components as e.g., scales. Also GPS based localization capability for mobile phones may be available as a future source of similar information. This data structure holds the actual passenger numbers as detected live during train operation.

7.5.3.5 Data Structure PassengerCountDetectedStation

This sub-structure contains a *TrainOCPRef* referring to a commercial stop for passenger (dis-) embarking and one or more sub-structures of type *PassengerCountDetectedStationUnit*.

7.5.3.6 Data Structure PassengerCountDetectedUnitStation

This sub-structure contains a *VehicleRef* referring to the rolling stock unit and data element *PassengerCountDetected* representing the number of detected passengers within it.

7.5.3.7 Data Structure PassengerTimetableUpdate

This data structure is used for sending updates on passenger timetables as published by TMS to the respective Passenger Information Systems of the RUs or other parties.

It is the result of applying an appropriate filter for passenger services to the Offline or Online Timetable data structure of the IL.

8 Conclusions

Even if the Traffic Management System, the Signalling Systems, Energy Management and many other internal systems are in the hand of one Infrastructure Manager the systems are not fully integrated. Especially in the area of optimising the processes and used functions the systems does not deliver and exchange the necessary information today. The same is valid for external systems. Often they are not even connected to the Traffic Management System. So Passenger information or information about freight demands e.g. cannot be considered during runtime calculation or conflict resolution.

Another problem is the Interfaces that are always developed new for every new project and systems/software that are connected. So far no common standard communication model is developed in the area of operation and traffic management. For the planning process the RailML Version 2 represents a well-known (de-facto) standard for serialisation of timetables and infrastructures. Thus it makes sense to take this model as an initial point for the developments of the In2Rail-specific Canonical Data Model for Traffic Management purposes. To communicate this standard data model a standard communication platform is introduced, called Integration Layer to connect internal and external systems for the traffic management.

The aim of this report has been to describe the data structure and data that is exchanged for internal Interfaces of the Integration Layer as well as External Web Services and Dynamic Demand IFs.

This document describes the Information Topics interchanged by the Integration Layer for the different traffic management processes.

The scope of the In2Rail data modelling is limited to the data needed for the main processes of Traffic Management:

- Infrastructure;
- Timetable;
- Train Control;
- Energy;
- External Services.

This specification is the first version and contains only the information topics needed for the core processes of Traffic Management. With the feedback of WP7 it will be enhanced and further developed during several Shift2Rail projects before getting a real standard. After that, it will be used as a basis for development of new standardised communication platforms and interfaces. It is expected that the IL will be extensively used to reduce development efforts and provide integrated systems.

9 Glossary

Term	Definition
CDM	The Canonical Data Model as defined in [CDM] is representing the data model definition as used within the Integration Layer.
Forecast	Estimate information in the future, deviations from the plan estimated.
Integration Layer	Communication link between the different Business Services.
Node	Typically, it is a computer with some operating system running or a virtual machine where bundles/containers are deployed. With the existence of cloud based Container-Services, the term Node can mean a managed cluster as well. In this sense the Node is one unit of the execution platform.
Nowcast	Instant information in the present.
Publisher	Entity that publics messages to be consumed by one or more subscribers. Actor of a business process that publishes Topics (i.e. make available and updates).
Subscriber	Entity that consumes messages sent by the Publisher. Actor of a business process that receives updates about one or more different topics it has subscribed.
Topic	Information specific to a business process. A topic is made up of a structured collection of operational data.
xxxDesired Topic	Topics representing a "Fore-cast" or a requested state (Command) are characterized by the word "Desired" in the Topic name.
xxxState Topic	Topics representing the actual state show the expression "State" in the Topic Name.
Indication	An "Indication" is a set of data representing static, dynamic and process data embedded in a topic for objects, routes, sections or trains. These data are updated from a source (publisher) which controls or monitors the assets when changes of state occur, an update is required from the operator/TMS or the client indicates an executed action after a command or command acknowledgement. A self-closing tag or an empty tag in an "Indication" means that the status information on this attribute is removed.
Authority	The Traffic Management System has Authority on RBCs and Interlockings.
Control	The RBC controls trains and an Interlocking has control over Routes, Sections and Objects.
Command	In the context of Integration Layer, commands represent a "desired" state, which is published in a xxxDesired topic and continues to be active, until the "desired" state is achieved.
Simple Command	In case of a simple command procedure, the issuer updates the appropriate xxxDesired topic. The command receiver executes

Term	Definition
	the command and updates the xxxState topic when the action is executed. In case the Command is rejected, an Alarm is triggered.
Acknowledged Command	Acknowledged commands have an extra acceptance step where the Client confirms the that it is possible to execute the command updating the xxxDesired topic. Then the issuer updates the appropriate section of the topic with a Command Acknowledgement to request the start of the execution of the command. In case the Command is rejected, an Alarm is triggered.

10 References

ERA – ERTMS/ETCS Class 1 System Requirement Specification – July 2017, SUBSET-026 Version 3.6.0

[CDM]	Annex to D8.3 and D8.6. Description of the Canonical Data Model, In2Rail, 2017
[D8.3]	Description of Integration layer and Constituents, In2Rail, 2017
[D8.6]	Description of the Generic Application Framework and its constituents, In2Rail, 2017
[D8.7]	Interface Control Document ICD for Application specific Interfaces, In2Rail, 2017
[D10.4]	TMS/MMS Intercace Specification, In2Rail, 2017
[Protobuf 2017]	https://developers.google.com/protocol-buffers/
[Hazelcast]	http://docs.hazelcast.org/docs/3.8.1/manual/html-single
[Redis]	https://redis.io/documentation
[Shift2Rail]	https://shift2rail.org/
[OMG DDS]	https://www.omg.org/spec/DDS/About-DDS/
[BookKeeper2017]	http://bookkeeper.apache.org
[Zookeeper2017]	http://zookeeper.apache.org/
[Mesos2017]	http://mesos.apache.org