real-time mobility data exchange 1.0 MVP only

Real-time Mobility Data Exchange

IRS 90918-11

Version 1.0: First version final 8th January 2024

real-time mobility data exchange 1.0 MVP only

Content

real-time mobility data exchange 1.0 MVP only

1 Summary and reading guide

Part of the European ambition of seamless travel in Europe is the realization of a service providing accurate real-time data on trains and replacement buses in Europe. This document holds a description of a service providing accurate real-time travel information for passengers. The purpose of the real-time information is to provide the passenger with information about disruptions and delays and, in the future, perhaps additional real-time platform services. The elements of the document are described below. Using the link, one can jump directly to the chapter of interest.

Glossary clarifies the terminology necessary to understand the document.

Introduction talks about the objective of seamless train travel in Europe and the current situation of real-time data exchange. The current situation of real-time passenger information is that information must be collected from multiple sources. The consumer of data must unify the data and piece together full train services before they can present the information to passengers. When writing this document, standard European regulations like the Rail Passengers' Rights and Obligations, and TAP-TSI were taken as a guideline.

In Business Capabilities the main capabilities that must be provided by the solution are described.

Requirements listed in this document cover the functional and the non-functional requirements for a real-time platform. The aim of a real-time platform is one platform and one standard across Europe, so one-stop shopping is possible regarding real-time on-train services. A standard that is open to all Data Providers and Data Consumers contributes to that. One of the goals is the availability of real-time information for passengers to any party that wants to serve the passenger in seamless travel. Consolidated output reduces the complexity to process the real-time information for all Data Consumers.

In Actors – setting the scene, the most important actors are described. That are Data Provider, real-time Data Consumer and two administrator roles. The Platform Admin is giving controlled access to the services of the platform and the Platform Data Manager is responsible for timely updates of the reference timetable and for checking the system quality.

In Use Cases the use cases to be covered by the system are described. Important use cases are:

- Load a (new) timetable reference plan
- Use cases with the goal to combine data to a full train service with consolidated real-time.
- Use cases on registration of sender and receiver of data.
- Use cases on defining/filtering what data to send to consumers.

The main service of the platform will be a push service providing real-time data on train services. Where swift processing and delivery of received real-time is key. But there is also a service to collect information on a single train service or a selection of train services. This is needed for consumers to build a reliable system.

Merging information on train services operated by more than one Data Provider (often cross-border trains) and filling gaps by estimating the missing values will solve matching problems centrally for all Data Consumers. Business rules for merging and estimating are described in paragraph Match and

real-time mobility data exchange 1.0 MVP only

merge ServiceRun data from multiple Carriers The real-time service takes into account that not all consumers want estimated values. Estimated values are, therefore, recognisable as such.

APIs describe the required API-specification covering all information needed to present train services in travel information. This document refers to the exact content and structure of the messages to API definition in the Bibliography

Guidelines for architecture and service level requirements are in described in Architectural design and Required Service Levels. With 160.000 Service Runs per day in Europe, it is expected that the platform will send 96.000.000 messages per day when all Data Providers in Europe are connected. The platform needs to have an availability of 99,9%. There must be 24x7 operational support for this service.

real-time mobility data exchange 1.0 MVP only

2 Glossary

In this glossary terminology is explained, that is necessary to understand the scope, the requirements and the solution described in this document. The list is sorted in alphabetical order.

Border Crossing	Crossing the border between the areas of two Infrastructure managers. Border crossings between IMs are not related to borders between countries.
Carrier	Railway Undertaking (RU) concluding the transport contract with the passenger and is responsible for the transport towards the passenger. In this document, a Carrier is addressed as a Data Provider when the emphasis is on that fact that data is provided to the platform. (see also Actors – setting the scene)
Consolidated Estimated Times	The estimated time by the platform for upcoming departures and arrivals is derived from applying consolidation business rules on the estimated times provided by the Data Provider. (See Figure 1 for the relation of the different time values)
Data Consumer	In this document, the Data Consumer is the party that collects real-time information to process this into travel information for the passenger. The words Data Consumers and the user are used interchangeably in this document.
Data Provider	A system that provides data (Service Runs) to the platform. In most cases, this will be a RU or a Carrier.
Distributor	Company combining transport contracts for a journey to a customer for passenger(s). The distributor might provide a travel companion and sales app to the passenger and/or customer.
Estimated Times	Estimations for upcoming departures and arrivals. There might be multiple estimates due to the involved parties (IMs, Carriers, Distributors, …) and the used algorithms. (See Figure 1 for the relation of the different time values)
Infrastructure Manager (IM)	The party that is responsible for maintaining an operational rail infrastructure. The IM is subjected to TAP-TSI to provide real-time information on trains to Railway Undertakings (RUs). (see also Actors – setting the scene)
Master	Responsible RU for stops of a service run.
NAP	National access points provide free available data
OAUTH 2	Standard for authorization. https://oauth.net/2/
Actual Time	Actual time of arrival or departure at a place. The Actual time from an IM perspective differs from the actual time from the passenger's perspective as defined in the Passenger Rights Regulation (PRR), which is the opening and closing of the access to the train service (e.g. opening of train doors).
OMG	Object Management Group (www.omg.org)
OSDM	Open Sales and Distribution Model (IRS 90918-10, https://osdm.io) (Ref see Bibliography) OSDM includes Real-time Data in the trip data and provides mechanisms to push real-time data related to a booking.

real-time mobility data exchange 1.0 MVP only

Planned Time	The planned arrival and departure time in a Service Run provided by the Data Provider. (See Figure 1 for the relation of the different time values)
Platform Admin	Administrator with access and maintenance rights for the platform (see also Actors – setting the scene)
Platform Data Manager	Person to validate the quality of the platform service and data accuracy. Send request to Data Provider when inconsistency need to be solved.
Railway Undertaking (RU)	The term Railway Undertaking (RU) is used in TAP-TSI for undertakings managing train services. The RUs might have the role of Carrier, Distributor or Retailer. In the light of this document the role as Carrier is of importance. In this document the RU is in most cases addressed as Data Provider or Carrier. (see also Actors – setting the scene)
Retailer	Company selling transport offers from distributors to the customer. For a real-time platform this a consumer of real-time data.
Service of the platform	A platform service is a task the platform can do like the service to send real-time messages
TAP-TSI	EU regulation on Telematics Application Passenger Traffic – Technical Specification for Interoperability.
Timetabled Times	Planned departure and arrival times according to a timetable. As there are multiple timetables the term planned time does not refer to a unique time. IMs usually reference a daily timetable for operations. Carriers and Distributors may refer to a defined timetable they compiled for their sales systems. (See Figure 1 for the relation of the different time values)
TIS	Train Information System provided by RNE (Rail Net Europe) implementing the TAP-TSI RU/IM data exchange.
Train Service / Service Run	A train service or Service Run is a description of a train with amongst other origin, destination, stops and arrival and departure times.
UML	Unified Modelling Language. A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems. OMG Unifier Modelling Language Superstructure v 2.1.2
UUID	Universally Unique Identifier. Standard to create a unique id. The specification is published as ISO/IEC 9834-8:2005.

real-time mobility data exchange 1.0 MVP only

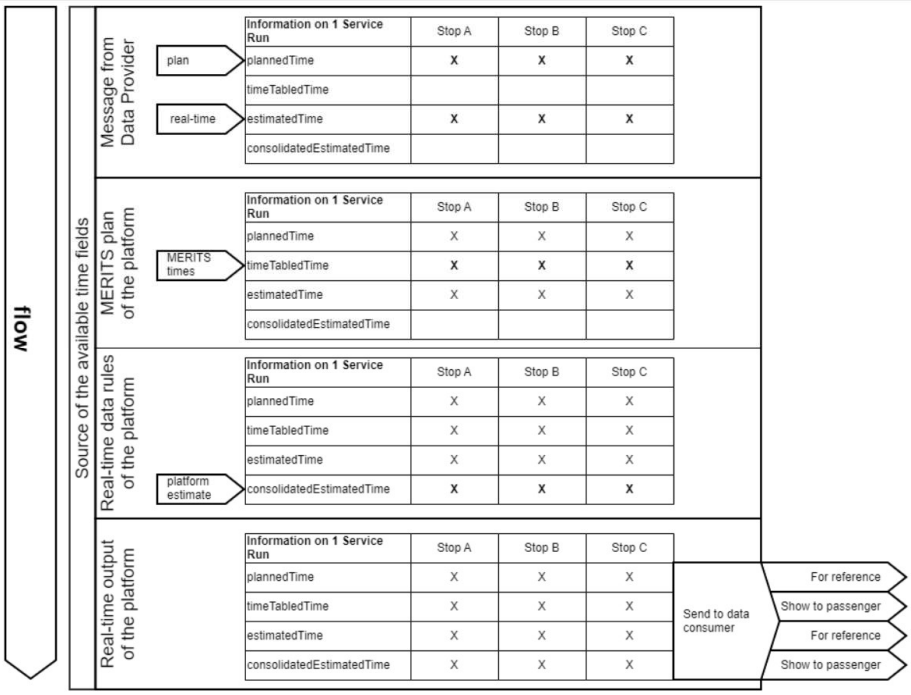


Figure 1 Source of the available time fields.

real-time mobility data exchange 1.0 MVP only

3 Introduction

3.1 Vision

This document is intended to support the vision of a more seamless passenger experience. A central platform according to this specification is:

- providing high-quality real-time data relevant for passengers
- on all services in one place
- including multi-RU services
- including complex services (merged, split trains)
- to be provided to all parties serving the passenger
- for an entire journey

This document contributes to a solution that will provide high-quality quality integrated real-time passenger information for all European trains. The described platform provides one-stop-shopping for all interested consumers using an open API standard for communication. A Data Consumer can collect all real-time data from one platform. The platform needs to be adaptable to future requirements. And the aim is to have the platform in successful operation, providing valuable passenger information to Data Consumers for more than a decade.

3.2 Current Situation of real-time data exchange

IMs change a train or detect change in a train run. IMs provide this logistic data to RUs which turn this information in passenger information. Consumer of this data that already provide a European wide service collects this information from multiple sources, mostly directly from the Carrier, but also from platform services which already combine information on a few Carriers.

There is legislation to improve the availability of real-time information. There is TAP-TSI data, but this is logistic information and not passenger information. And more recent channels are the National Access Points (NAP) every country needs to have. National Access Points are free in the format they use and an integrator functionality is not a required part of their platform. NAP's can also be only the central location to direct consumers directly or indirectly to the actual sources of data.

To sum up: The current situation is that information must be collected from multiple sources. The consumer of data must unify the data and piece together full train services before they can present it to the passenger.

3.3 Methodology

This document follows the UML specification to define the solution. In this document, you will find:

- Actors
- Sequence diagrams
- Use Case diagrams
- Deployment diagrams

UML data models have been replaced by the graphical data description used for JSON data structures.

real-time mobility data exchange 1.0 MVP only

3.4 Regulations

The following regulations are taken as guideline to design the functionality of the platform.

- **Rail PRR:** Regulation (EC) 1371/2007 on Rail Passengers' Rights and Obligations (Ref see Bibliography)
- **TAP-TSI** Revision (Ref see Bibliography)

  Hereafter how the obligations of PRR article 10 are transposed in the TAP recommendation of January 2022 chapter 4.2.18.1 and following:

  "The information about the train running forecast shall be delivered to the station manager, ticket vendor and tour operator in due time by the railway undertakings and/or infrastructure managers according to the conditions in article 10 of Regulation (EU) 2021/782."

  The important word is RUs "and/or" IMs. That means the IM should agree with the RUs on its network who should deliver the real-time information. During the TAP Revision WP, the RUs expressed the wish to deliver the information instead of the IMs.

real-time mobility data exchange 1.0 MVP only

4 Business objects and business capabilities

4.1 Business objects

The real-time data platform deals with the following business objects [1]only:

- The Service Run, the entity of information on a train. The data components that are part of the Service Run can be found in the API description. (See Service Run)
- The reference timetable of the platform. The reference timetable will be used by the platform to send out consistent planned times at stops. The timetable is used to merge real-time data from different RUs on the same Service Run.

The scope is all Service Runs in Europe operated by train or train-replacing transport.

4.2 Business capabilities

The basic objective of the real-time exchange here is to enable the following business capabilities:

- Allow a Carrier responsible for a train service to provide real-time information on a Service Run so that it will be part of integrated passenger information.
- Provide a solution for combining data for multiple Carrier services and cover all rail-related specifics
    - Combine real-time data from multiple Data Providers for the same service in a transparent way
    - Delay information, platform information, connections
    - Manage complex split and / or merged Service Runs
- Provide a specified open data format for real-time exchange for travel information purposes, so that:
    - Data Providers and Data Consumers do not make unnecessary costs on many convertors or on fee to standardization organisations
    - It enables the community to be compliant with competition law
- Provide integrated and consolidated output data in the specified format
    - Consistent delay information on an entire Service Run
    - One output for a service covering data provided by multiple providers
    - Transparency on the consolidation done by the platform

[1] A business object is an entity within a multi-tiered software application that works in conjunction with the data access and business logic layers to transport data. An example would be a concept like "Process" having "Identifier", "Name", "Start date", "End date" and "Kind" attributes and holding an association with the "Employee" (the responsible) that started it.

real-time mobility data exchange 1.0 MVP only

- Provide a centralized access point fit for collecting all real-time of all RUs in Europe.
- Provide a lean platform service so that the platform is attractive as source for consumers preparing passenger information.
    - only exchange valuable data
    - provide filtered data
    - provide services focused on the use case
- Provide an extensible specification open for future new requirements, to adapt to new or changing requirements when the API and platform are in production.

4.3 Basic principles

The following basic principles are behind the solution in this document:

- The provided real-time data has aways priority over the reference timetable.
- Data changes or additions by the platform must be made visible to the Data Consumers.

real-time mobility data exchange 1.0 MVP only

5 Requirements

The aim of a real-time platform is one platform and one standard across Europe, so one-stop shopping is possible regarding real-time on-train services. In this chapter, functional and non-functional requirements are listed. The requirements describe **what** the platform needs to do. **How** the platform could be built is not part of these requirements. The requirements need to provide freedom for experts and developers to technically specify and build the platform. In Requirements not in the scope for MVP there are requirements that are out of scope of the real-time platform, for a better understanding of what the platform will and will not do.

5.1.1 Functional requirements

- API to allow Data Providers to provide real-time information for passengers on trains.
- API to allow a Data Provider to send train replacement transport (e.g. busses).
- API to allow Data Consumers to receive push messages with real-time updates.
- API to allow Data Consumers to pull the status of individual trains or a set of trains.
- API for Data Consumers with filter functionality to allow the definition of filter conditions on the data to receive. A complete list of filters can be found in the API definition [see Bibliography]. For example, there must be filters on Country and Service brands and transport modes. More examples in paragraph Filter criteria:
- API must provide fields to exchange real-time information on trains. The message contains a complete state of a Service Run, so that Data Consumers do not need to combine information from multiple messages. A message includes:
    - Delays
        - Estimated times of arrival and departure
        - Time-tabled times of arrival and departure of a referenced timetable
        - Reason for Delay as a result of the reconciliation between RUs and IMs provided by RUs (PRR requirement)
    - Change of train composition (optional for Data Providers)
        - Train composition with track/platforms / coaches/platform sections
    - Replacement trains (PRR requirement)
    - Additional trains
    - Train replacing busses
    - Platform changes (PRR requirement)
    - Connections to other train services (optional for Data Providers)

real-time mobility data exchange 1.0 MVP only

- Cancelled Trains (PRR requirement)
- Cancelled Stops
- Additional Stops
- Geo-positions of a train (e.g. on passing CRD locations) [2]

- The API needs to send the following information to Data Consumers so that Data Consumers can match the real-time information for passengers to their reference plan[3] when their reference is different than the reference on the platform:
  - unique identification for a train provided by the platform
  - commercial train service number
  - operational train service numbers (optional, for future use)
  - line number
  - RUs of the train service
  - operational RUs of the train service if known
  - transport mode of the train service
  - vehicle IDs (European Vehicle Number) (optional)
  - planned arrival and departure times (as known in the real-time platform)
- The platform will only store one plan (integrated plan with all train Service Runs in Europe) that will be used for referencing and merging the real-time messages to keep the amount of data in the platform limited.
- Real-time platform service must pass on every real-time message from the Data Provider to the consumer, when it holds a change in relation to the last message sent for the same Service Run:
  - The first real-time message on a Service Run, regardless the time before departure or the information in the reference plan of the platform.
  - An update of previous received real-time information
- The platform will contain an integrator function that will combine partial information on trains into a real-time message with planned times and real-time on all stops. When part of the data is not yet available, then the platform will give an estimation for the missing information. (See for details Match and merge ServiceRun data from multiple RUs)
- An integrated timetable should be used by the integrator function of the platform to solve data issues e.g., when the planned times of two real-time messages for a train service delivered by two different Data Providers cannot be connected.
- Possible to grant access to any consumers (not only RUs that provide input)

[2] The MVP provides only the data format to communicate geo-positions. A message to do vehicle tracking by Data Consumers is not part of the MVP.
[3] A reference plan is for example used in travel planners. The plan contains the planned times for the coming weeks. The plan is updated with real-time data.

real-time mobility data exchange 1.0 MVP only

- Possible to process data on multimodal services when sent to the platform in the generic API standard of the platform.[4]

- It should be visible to the Data Consumer whether the platform has consolidated data or simply passed them through.

- The platform must provide quality control measures. A minimal outline for monitoring of data quality:

  Quality of the input. Per involved Data Provider:

  - Number of cases where a consolidation of delays was needed
  - Number of unmatched station codes

  Quality of output. Per registered consumer:

  - Number of retries necessary
  - Number of published notifications vs. number of retrieved Service Runs

- The platform must provide monitoring of platform performance and load

  - Number of updates on Service Runs received (per provider)
  - Number of Service Runs published
  - Number of Service Run notifications published (per consumer)
  - Number of Service Runs retrieved (per consumer)

5.1.2 Non-functional requirements

- Open Interfaces as API, API specification is open.
- Easily extendable interface, so an increase in information items or adding of functionality can be done without a major redesign. The platform must be future-proof.
- Extendable to hold all train services of all Data Providers in Europe. Resources should be scalable and flexible to accommodate for increase and decrease of the amount of data / number of integrations / number of requests / etc.
- Date time values must be encoded according to RFC 3339, section 5.6.
- Information exchange must support a state-of- the-art technology and architecture so that the system can handle the required number of messages and can be used for over a decade.
  - REST OPEN-API 3.1 (see Bibliography)
- A real-time message should be processed in 10 seconds between receiving a message from a Data Provider to the sending of the notification.

[4] The possibility is there, but at first the MVP will only hold trains and train replacement services..

real-time mobility data exchange 1.0 MVP only

- The platform supports versioning by content negotiation. The client provides the version in the accept header. The objective is to support two versions, to be able to move from one version to a new upgraded version.

5.1.3 Requirements not in the scope of MVP

The scope of the platform is real-time information for passenger information, where all real-time information is sent in the same format. To limit the complexity of the platform and to limit the size of the messages, the following is out of scope:

- Support for other import and export formats:
  - GTFS RT (Optional, only if it can work without GTFS offline)
  - TAP-TSI RU/IM (format TSI xsd taf-jsg.info/?page_id=172)
  - SIRI (optional, only if it can work without offline NeTEX data)
  - VDV (optional, only if it can work without offline data from VDV)
- Information on stations (e.g. accessibility and status of elevators)
- Train tracking messages and geographical positions

real-time mobility data exchange 1.0 MVP only

6 Actors – setting the scene

The following diagram shows the actors [5] and their relation to the platform. Note that the IM is not expected to have a direct relation to the platform. IMs provide real-time information to RUs. RUs are the Data Providers of the platform.

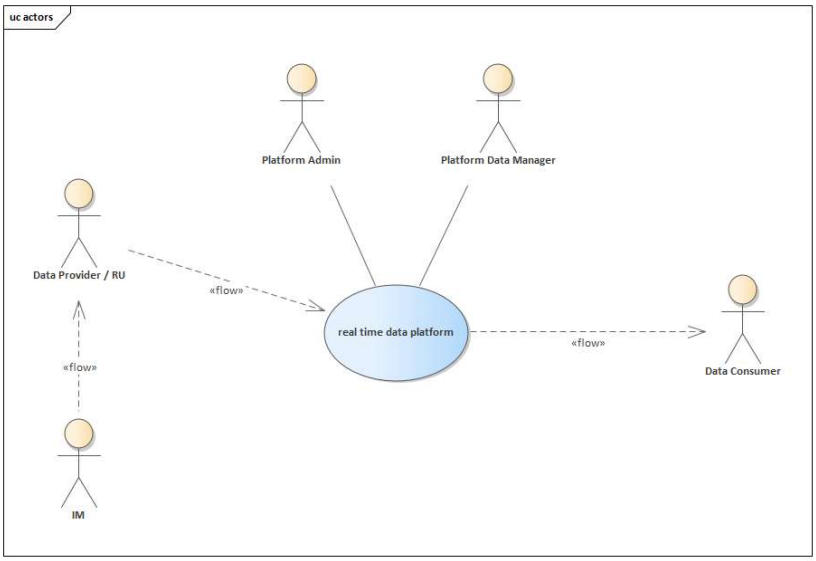


Figure 2 Actor model

[5] An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject.

Actors may represent roles played by human users, external hardware, or other subjects**. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity** that is relevant to the specification of its associated use cases. Thus, a single physical instance may play the role of several different actors and, conversely, a given actor may be played by multiple different instances.

Source: OMG Unifier Modelling Language Superstructure v 2.1.2

real-time mobility data exchange 1.0 MVP only

Role Name **Infrastructure Manager / IM**	
Definition	Infrastructure Manager according to TAP-TSI.
Motivation / Distinction to other roles	Delivers delay information and changes in train runs, that occur on his infrastructure to the RU. The IM is the original source of the data that is offered by the RU to the platform. The IM will not be an actor in the MVP.

Role Name **Data Provider**	
Definition	The Data Provider is the RU concluding the transport contract with the passenger and responsible for the transport towards the passenger.
Motivation / Distinction to other roles	The actor delivers real-time information to the platform.

Role Name **Data Consumer**	
Definition	Data Consumer is a party with a verified and active license to collect Real-time information from the platform.
Motivation / Distinction to other roles	The user of the real-time data can have multiple commercial roles. It can be a distributor or retailer involved in customer information or a provider of schedule information only. All have a role in bringing passenger information to their customers.

Role Name **Platform Admin**	
Definition	An entity that has access to the platform and has the rights to make changes to the platform.

real-time mobility data exchange 1.0 MVP only

Motivation / Distinction to other roles	The Platform Administrator has a role in managing the access to the real-time data access points. The Platform Administrator will verify that an order in the UIC webshop is finalised before granting access to the real-time platform.

Role Name **Platform Data Manager**	
Definition	An entity that has access to the platform and has the rights to make changes to the platform configuration data.
Motivation / Distinction to other roles	The platform data manager has a role in updating the timetable in the platform and checking the system quality. And is responsible for addressing problems with Data Provider data to the Data Provider.

real-time mobility data exchange 1.0 MVP only

7 Use Cases

7.1 Overview of use cases implemented in the platform

This chapter describes the use cases that will be covered by the Real-time platform for passenger information . In Appendix B The structure of a Use case the used structure is described.

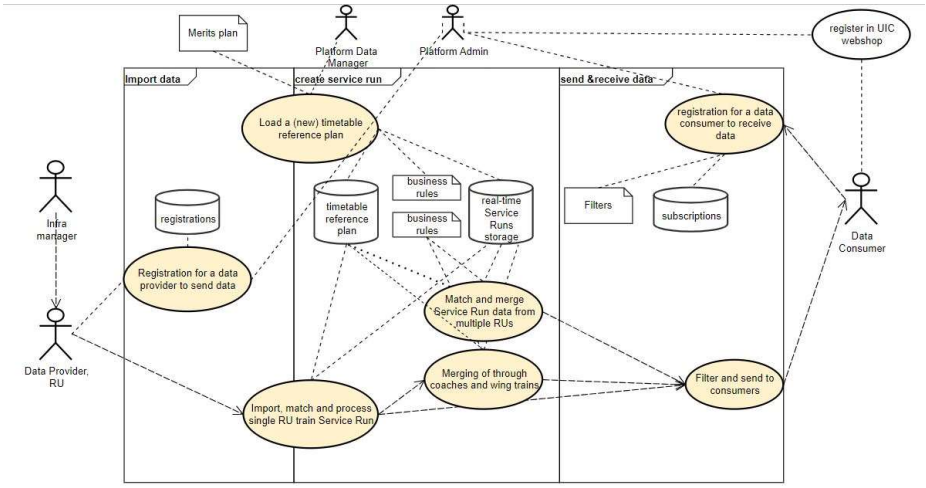


Figure 3 relation between the use cases covered by the platform.

7.2 Load a (new) timetable reference plan

Description: This use case covers the loading of timetable data in the real-time data platform. The timetable is used to make all planned times in the platform consistent and it is used as a consistent reference to connect data from border crossing trains. Messages send to consumers will have planned times from the reference time-table.

Trigger: A update of the plan is available for the platform

Included functionality:

- Automatic import process
- Monitoring on the import process
- Check syntax, content
- Replace older plan
- Add unique id (uuid) for service runs. (if known in previous plan, then continue using same Id)

Involved actor: The platform data manager is responsible for monitoring the process.

Business rules:

- The basic rule is that the plan is replaced without updating the planned data in all present real-time.

real-time mobility data exchange 1.0 MVP only

- The loading of the new plan must not slow down the processing of real-time messages. Real-time messages can be processed against the previous plan as long as the new plan is not ready.

Additional information:

- Data Consumers must update their reference plan themselves.

7.3 Registration for a Data Provider to send data

Description: To send real-time data, a Data Provider is required to register for the data delivery. This requires a Data Provider agreement. The process is outside of the real-time platform. As result a Data Provider will get credentials to access the platform for sending data.

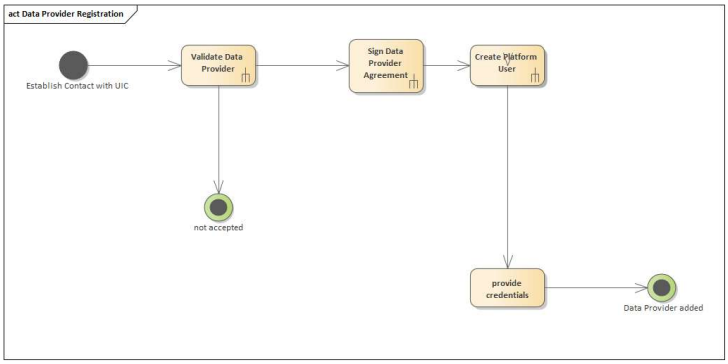


Figure 4 Registration Process

Trigger: A Data Provider contacts the Platform Administrator.

Included functionality: The …

- Functionality to create platform user. (end-point to place a registration via the API)
- Functionality to create and store credentials to access the acceptance platform end-point.
- Functionality to create and store credentials to access the production platform end-point
- Functionality to provide credentials to Data Providers (can be via e-mail)
- Functionality to view a list of providers and their rights of use

Involved actors and their role:

- The Data Provider requests for access and provides credentials
- The Platform Admin will check the credentials and provides access to the platform

Out of scope: interactions with the UIC webshop

Additional information:

Validation of syntax is part of usecase Import, match and process single Carrier train Service Run

real-time mobility data exchange 1.0 MVP only

7.4 Registration for a Data Consumer

Description: To receive real-time data, a Data Consumer is required to register and purchase a license in the UIC web-shop. The Platform Administrator will create the credentials and configure (set filters etc.).

After registration the consumer will receive credentials to access the platform services and to create subscriptions for data deliveries.

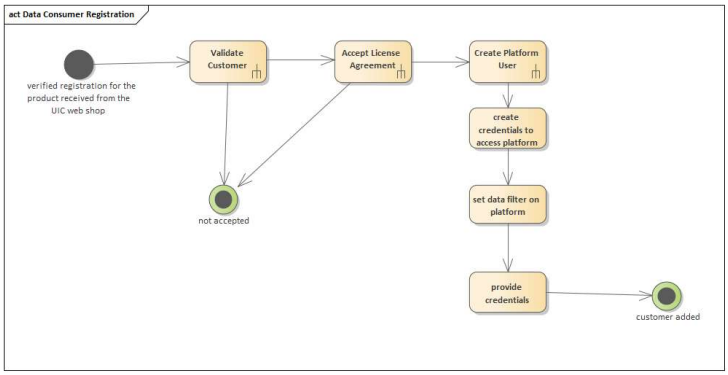


Figure 5 Registration Process

Triggers:

- Platform Administrator receives a new license from the webshop.

Included functionality:

- Functionality to create platform user (end-point to place a registration via the API).
- Functionality to create and store credentials to access the acceptance platform end-point.
- Functionality to create and store credentials to access the production platform end-point.
- Functionality to set filter by Platform Administrator (set what is available for Data Consumer).
- Functionality to provide by the Platform Administrator credentials to Data Consumers (can be via e-mail).
- Functionality to view a list of consumers and their rights of use.
- Functionality to manage (update/delete) platform user

Involved actors and their role:

- Platform Administrator provides the credentials and set the filter.
- Data Consumer receives the credentials.

Out of scope : interactions with the UIC webshop

Additional information:

Details on filter of the license (options in the webshop)

real-time mobility data exchange 1.0 MVP only

- Geographic (whole Europe or per country)
- Duration (one year, half year etc.)
- Number of subscriptions (according to APIs)

Details on filter options for the consumer

- how many webhooks to receive data (data feed one only cross border, other feed all data)

7.5 Manage a subscription

Description:

The Data Consumer can define the filters for the data delivery via the subscriptions.

Included functionality:

- Add a subscription
- Change a subscription
- Delete a subscription
- Get a list of all subscriptions
- Manage subscriptions

Involved actors and their role:

Data Consumer defines his subscriptions.

Platform Data Manager manages all subscriptions.

Business rules:

- The consumer is not allowed to exceed the subscriptions number according to the license
- The consumer has access only to their subscriptions

7.6 Import, match and process single RU train Service Run

Description:

This story covers the import of Service Runs provided by RUs in real-time platform data format. Train services that are owned by a single Carrier are stored in the database and send out to consumers without changing the real-time content. Only a unique number is added to the train service by which the train service is known in the platform's database. Train services that span more than one RU are passed on to the use case Merging Service Run data from multiple RUs.

Trigger:

- Train Service received by API of the real-time data platform

Included functionality:

The import and processing must provide:

- Validation of the message
- Find service run on the platform in the reference plan and add time-tabled time

real-time mobility data exchange 1.0 MVP only

- Recognise that it is a multiple RU train service, and pass on to the merging process.
- In case no service run found add platform specific unique train servicerunId.
- Update service run in the database.
- Trigger publication

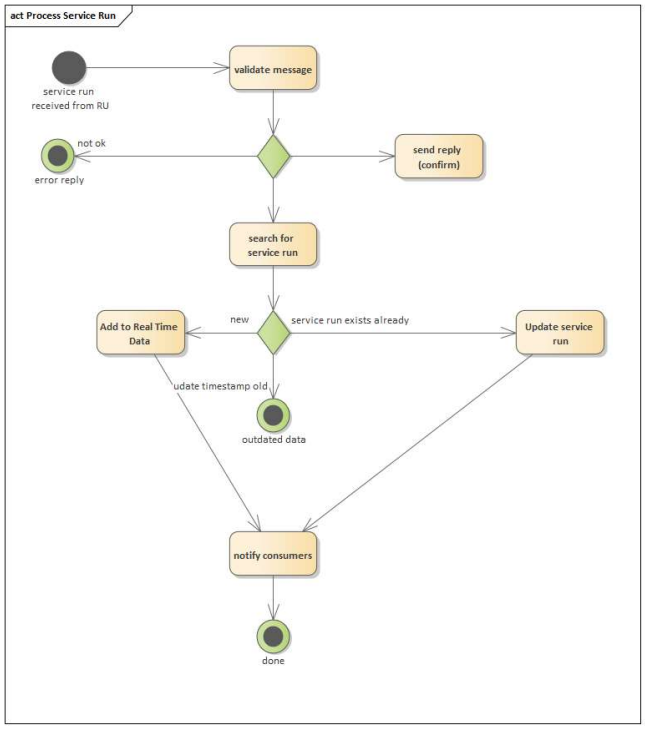


Figure 6 process new Service Run data

Involved actors and their role:

Data Provider sends a service run.

real-time mobility data exchange 1.0 MVP only

Business rules
The business rules related to making the message consistent in the platform are:

- Matching of the real-time message to service runs can be done by using the available information in the platform format only. For example, one or more of the following data: service number, service brand, carrier, stops, planned stop times on stops.

- Matching rules should be adopted to maximise the number of found and securely identified services. Matching rules must be made transparent.

- Stop times delivered might slightly differ between different providers. A configurable off-set in the comparison of planned times must be used to increase the matching rate. (e.g. if the planned times do not differ more than 3 minutes from the plan, then it is the same train.)

- Matching of train replacement services: Data provided for train replacement Service Runs must reference the train service in the plan it replaces when a reference to the original train service is given in the real-time message.

- Add the time-tabled time from the reference time table

- Solve discrepancies in number of stops between the information from the Data Provider and the reference time table. See table below.

- Check with real-time database if new the message contains information to communicate.

- Only store in real-time data base of the platform when a real-time message is sent to the Data Consumer, so that for a next real-time message the threshold for a significant change in delay is checked against the last send message. So, store data when:
    - The planned times are different. See table below for details.
    - There was no real-time send by the platform for this service yet.
    - There is a track change and platform section change that was not communicated yet.
    - There is a change in delay that is above at least one minute.

- A train service that is not in the reference time table plan is added as a new train service.

- A stop (departure and/or arrival) that is not in the reference time table plan is added to the train service as extra stop (departure and/or arrival).

- Actual times provided by the responsible RU, must never be changed back. Not even by a newer message from that same RU.

- Send real time data only in case there is a change in real time data or in case there were no real time data send before.

real-time mobility data exchange 1.0 MVP only

The following rules apply to send out real time data:

	In received real-time message	**Already in real-time database**	**Action related to planned times**
1	yes	yes	Store and send new real-time message when the information differs from the information already in real-time database.
2	yes	no	Store and send it out

The following rules apply to process cancelled stops. The system must be robust for different ways of communicating cancelled stops. The platform will always send information on cancelled stops, cancelled stops are not removed from the train service.

On stops where the provider of the message is responsible:

	In received real-time message	**In time table plan**	**Action related to planned times**
1	Stop d not in real-time: Stops a, b, c	Stop d in plan: Stops a, b, c, d	Add **cancelled** stop to real-time message, store in r-t database and send message
2	Stop d in real-time: Stops a, b, c, d	Stop d not in plan: Stops a, b, c	Mark as **extra** stop in real-time message, store in r-t database and send message
3	Stop d is cancelled: Stops a, b, c, ~~d~~	Stop d is not cancelled: Stops a, b, c, d	Cancelled stop d in real-time message, store in r-t database and send message

In the table above stop ~~d~~ means that stop d is marked as cancelled.

real-time mobility data exchange 1.0 MVP only

7.7 Match and merge ServiceRun data from multiple RUs

Description:

This use case handles the creation of a full train service from individual parts received by more than one Data Provider.

Multiple RUs might provide real-time data for the same train service. These will be identified and combined by the real-time data platform to provide the consumer a unique view on the real-time data of that entire train service.

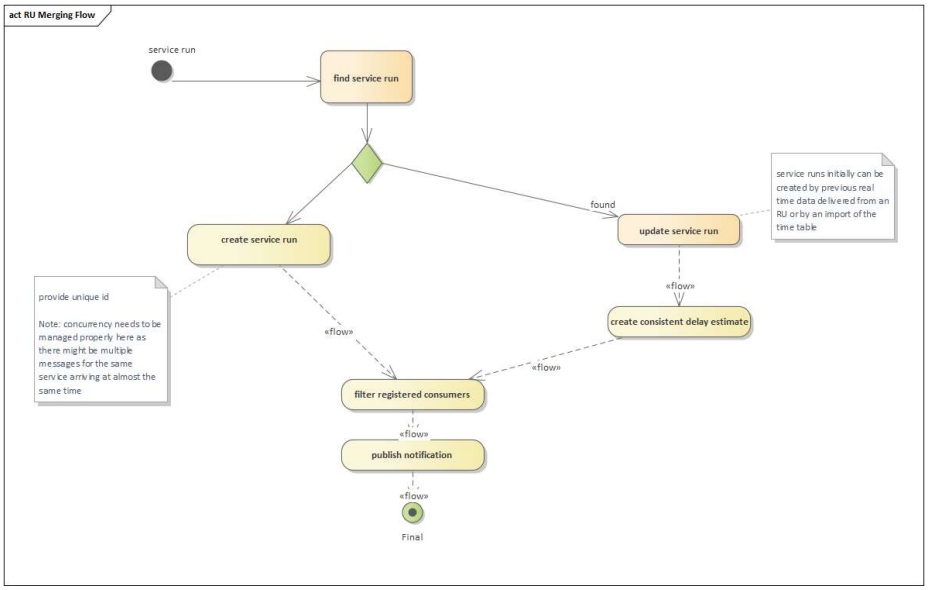


Figure 7 Process train Service Run

Trigger:

- Train service identified as multiple RUs received from use case “import, match and process single Carrier train service”. In the use case “import, match and process single Carrier train service” the planned times from the time table were already applied to the message

Included functionality:

The merging of real-time must include:

- Use information of Data Providers for those stops they are “master” for.
- Fill gaps (arrival and departure times) in the data by applying business rules.
- Store information in real-time database when the message contains new information.
- Add to (new Service Run) or replace (existing Service Run) in database .
- Timestamp of the previous update. This allows consumers to detect missed updates.

Involved actors and their role:

real-time mobility data exchange 1.0 MVP only

The platform is executing all the processes.

Business rules:

The same rules apply as for the single RU service run described in paragraph 7.6.

Additional rules related to this usecase:

- Real-time information on a stop is filled in the following order:
    1. Real-time from the “master” (indication master is sent by Data Provider)
    2. Real-time from another Data Supplier (indication forwarded is sent by Data Provider)
    3. Consolidated real-time (set by the Platform)
- When there is a conflict in the data, because two Data Providers claim to be “master” for the same station then the status “master” is derived from the station code by applying the territorial principle. The territorial principal defines a default master to be used for stops of a country. Th country can be derived from station codes. The default masters need to be configurable.
- When there is a conflict in the data, because no Data Provider claims to be “master” for a station then the status “master” is derived from the station code by applying the territorial principle. This applies especially if a stop is provided by the time-table data only and the platform adds this stop with the indication as cancelled because it is not provided by the “master”.
- When a station is only in Merits and not in the real-time (see Import, match and process single RU train Service Run), then the platform sets the indication for consolidated real-time.
- When data of multiple RUs is inconsistent estimated real-time is made consistent by using consolidated real-time.
- Calculation of the consolidated real-time estimates:
    - The consolidated delay estimates must obey the following consistency rules. Make consistent by moving the impossible real-time on stops so far in time that the following rules apply.
        - Delay times must be consecutive along the route. (No time travel back in time)
        - Delay times must not indicate a decrease of travel time between two stops of more than 10% (value to be evaluated and configurable) compared to the planned travel-time between two stations.
        - Delay times on departure at one stop must not be later than the delay estimate at the arrival on the same stop.
        - The stop at station can be reduced to configurable minimal number of minutes. It must be allowed to set the value to no adaption of the stop time. (e.g. any stop longer than 10 minutes is reduced to 10 minutes)

Kommentiert [KvPP1]: Important change

On stops where the provider of the message is **not** responsible:

	In received real-time message	**In reference time table**	**Action related to planned times**

real-time mobility data exchange 1.0 MVP only

1	Stop d not in real-time: Stops a, b, c	Stop d in plan: Stops a, b, c, d	The timetabled time is taken from the reference time table. The stop is taken from the time table when there is no information from the “master”.
2	Stop d in real-time: Stops a, b, c, d	Stop d not in plan: Stops a, b, c	Mark as **extra** stop in real-time message in case there is no real time data from the “master”, store in r-t database and send message
3	Stop d is cancelled: Stops a, b, c, ~~d~~	Stop d is not cancelled: Stops a, b, c, d	“Forwarded” cancelled stop d in real-time message in case there is no real time data from the “master”, store in r-t database and send message.

In the table above stop ~~d~~ means that stop d is marked as cancelled.

Out of scope:

Indication of the type of changes compared to the previous version to be included in the notification for consumers.

Additional information:

In the following figure there is the process of transferring information between the actors related to a Service Run offered by multiple RUs. All RUs need to be registered as supplier and Data Consumer of real-time data. The RU responsible for the first part of the Service Run will send real-time messages to the platform. The platform will apply the business rules of the integrator functionality and will send a Service Run with planned a real-time information on all stops. All Data Consumers will receive this information. The RU responsible for the second part of the service can already make an estimate on his part of the service. When also this information is processed by the platform also the third RU has sufficient input to do an estimate. If at any point in this process information is missing in the platform, then the business rules of the platform will be applied and always a complete Service Run is sent to Data Consumers.

real-time mobility data exchange 1.0 MVP only

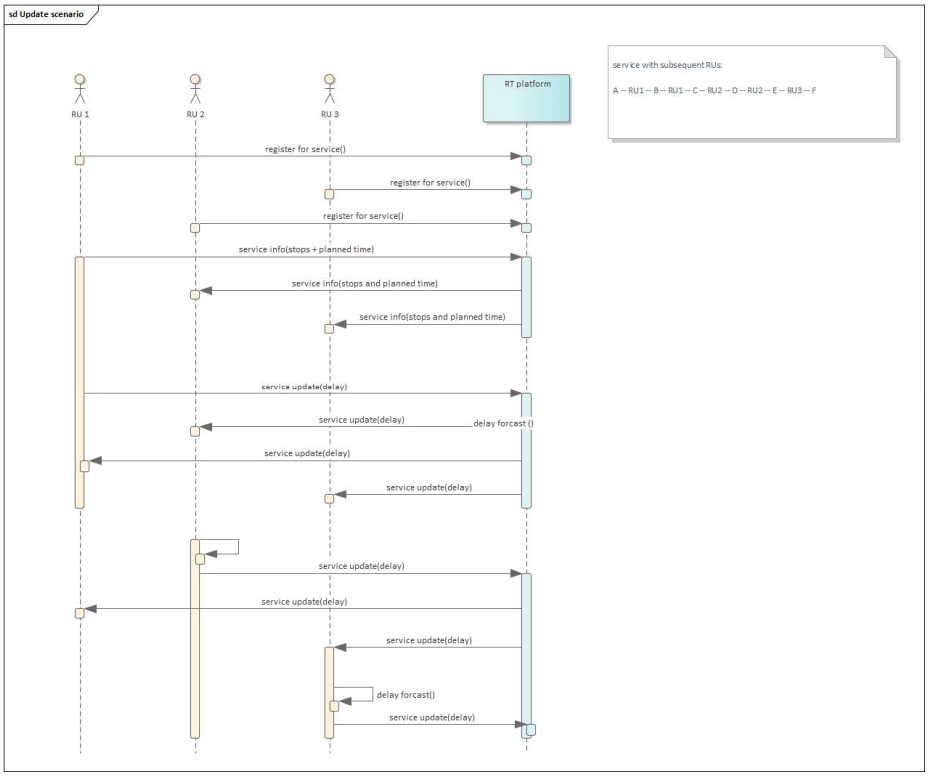


Figure 8 Real-time Flow between multiple RUs

7.8 Merging of through coaches and wing trains, split and join

Description:

More complex cases arise when rollingstock is combined and separated. The traveller in the coach perceives one direct trip without interchanges. The platform will need to send out real-time information that Data Consumers can process in such a way that changes are applied to the entire trip the traveller perceives. Only the RU knows which connecting service runs must be communicated as one, and which must be communicated as two service runs to the traveller. The RUs will provide this information to the platform and the platform will send out these cases in a consistent way that is equal for all relevant service runs, regardless of the RU.

real-time mobility data exchange 1.0 MVP only

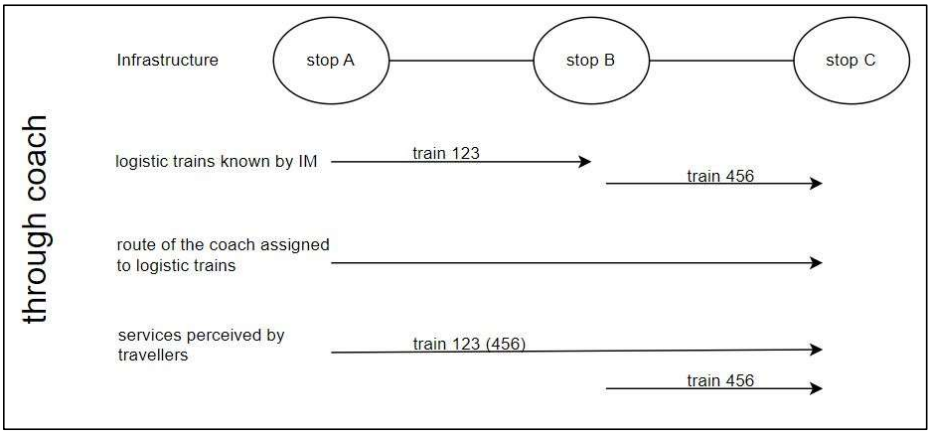


Figure 9 Explenation of a through coach.

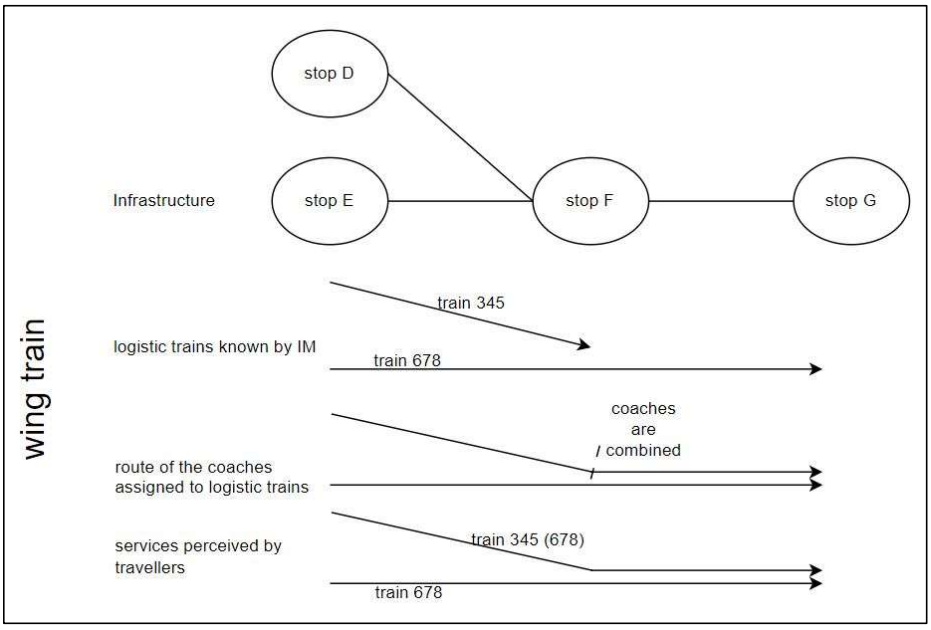


Figure 10 Explanation of a wing train.

In the platform the connection between these related train services is done by creating vehicle groups. There can be different train numbers per departure or arrival.

Appendix E complex example of through coaches and wing trains provides an example of a use case where wing trains and through coaches are combined. This example can be used to check if the created solution cover complex cases too.

real-time mobility data exchange 1.0 MVP only

The input by Data Providers before conversion to the platform API can follow different approaches as can be seen in the appendix.

Trigger

The triggers for the platform to recognise that the received real-time is related to a coach group are:

- Vehicle groups with different train numbers in the received real-time message.
- A received real-time message for which in the reference plan a coach group is defined.

Included functionality

This use case is about updating all services in the vehicle group where relevant.

Situations to cover:

- Consolidated delay at an arrival or departure must be the same for all combined services
- Platform at an arrival or departure must be the same for all combined services

Involved actors and their role:

The platform is executing all processes.

Business rules:

- For a service run with vehicle groups the change in the real-time must be applied on all other service runs containing these vehicle groups.
- The main service run Id must be the one known by the customer, on the customer ticket.
- When Master data between two vehicle groups is conflicting, the assumption is that the vehicle groups are no longer connected.

Additional information:

A suggestion: When developing this use case it might be wise to use the method of specification by example. Real examples are a valuable help to understand and to interpret the API.

Below there are three explanatory figures on split and join of wing trains and through connections. Every figure is built in the following way: The bottom row shows the infrastructure view on the service runs. In the top row there are ways the input is send by the RU to the platform with a customer view on the data. In the middle row there is the consistent platform output with a customer view on the data. In the Appendix F Simple example of through coaches and wing trains there are example messages to indicate the relationship between the figures and the platform API.

real-time mobility data exchange 1.0 MVP only

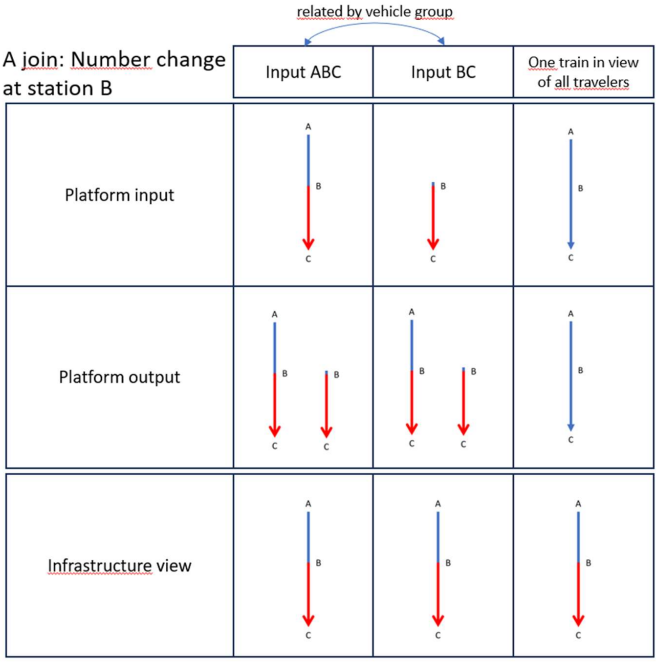


Figure 11 one piece of rolling stock running two consecutive runs. Perceived by user as one service run.

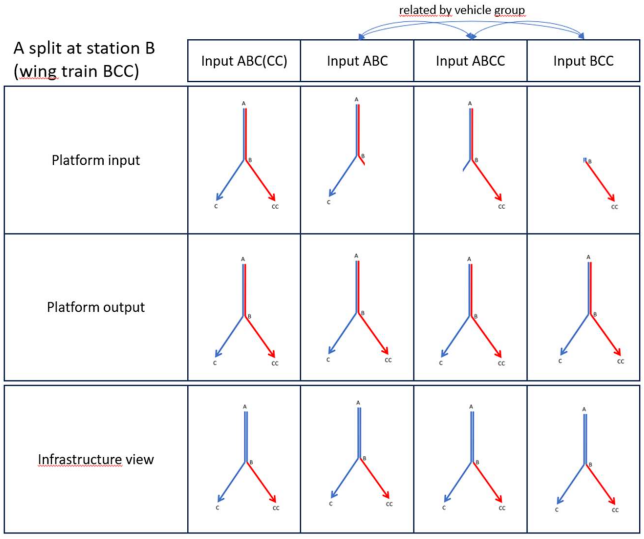


Figure 12 two pieces of rollingstock, partly combined as one service and then splitted.

real-time mobility data exchange 1.0 MVP only

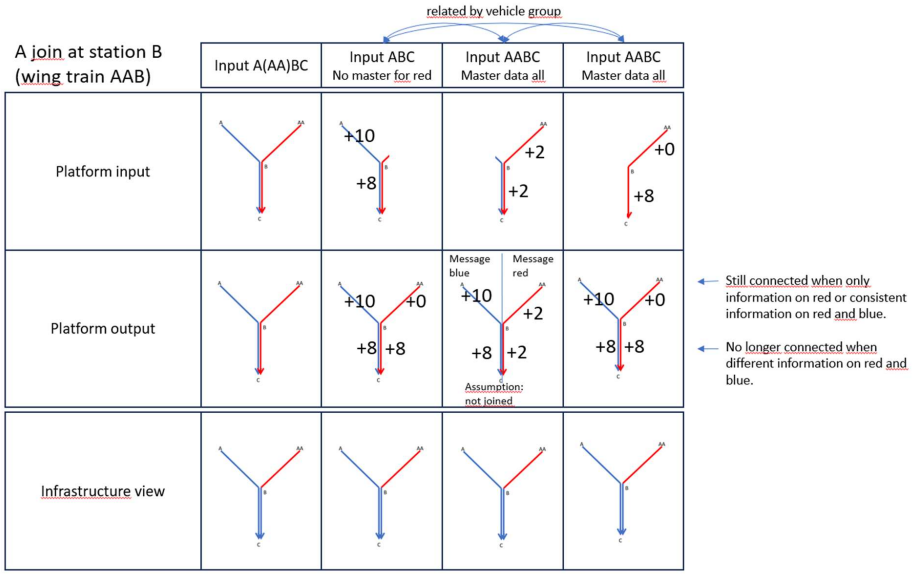


Figure 13 two pieces of rollingstock, two services joined into one service.

7.9 Filter and send to consumers

Description

This use case describes the required steps for filtering and sending consistent real-time service run messages to consumers. Messages may only be sent when registration and filter settings allow it.

In the following cases data is sent, based on the assumption that a combined push and pull mechanism is the best way to realize this data stream:

- Push:
  - Processed train service run information will result in a push message to the entitled Data Consumers providing them with the train service ids to get the data. The Data Consumers must be entitled according to the filter of the purchased product and the additional filters provided by the consumers in their subscription.

- Pull:
  - The provider will request train service runs using the train service run ids received from the push mechanism.
  - A Data Consumer can request for the latest version of a specific train service run of a specific RU on a specific date.

Triggers:

- Use case Import, match and process single Carrier train Service Run provides a consistent train service run to send.

real-time mobility data exchange 1.0 MVP only

- Use case Match and merge ServiceRun data from multiple Carriers provides a consistent train service run to send.
- Use case Merging of through coaches and wing trains provides a consistent train service run to send.
- The Data Consumer sends an API request to collect information on a Service Run or a set of Service Runs.

Included functionality:

The filtering and sending must provide:

- A guaranteed delivery (with a maximum of retries or retry period, extending the retry interval with each attempt (e.g. by a factor of 2) within the next 12 hours.
- Logging:
  - all updated service runs (with a maximum period of days or size)
  - to be kept for 7 days
- Applying the registration limitations and subscription filter settings
- A train service run under shared responsibility of more RUs must be send when at least one of the RUs is requested.

The filter parameter include:

- See paragraph Requirements for the filter options that need to be implemented.

Involved actors and their role:

The Data Consumer will send a confirmation for every received message. And it will send requests to collect data on service runs.

Additional information:

A Data Consumer will get a message with the following times:

objectType	- DEPARTURE - ARRIVAL - PASS_THROUGH
location	Location id and optional the name (for stations but extensible to other types of locations).
plannedTime	Planned time for the event according to the responsible Data Provider
timeTabledTime	Planned time according to the referenced timetable e.g. MERITS 11.11.2024
estimatedTime	Estimated time according to the responsible Data Provider
consolidatedEstimatedTime	Consolidated estimated time calculated by the exchange platform to avoid inconsistencies
actualTime	Time when the event actually happened
status	Status of the event (CANCELLED, DELAYED,...)
reasonForDelay	Reasons for a delay or cancellation of the event

Table taken from paragraph 0

real-time mobility data exchange 1.0 MVP only

The intended use of the output of the platform is to use the timeTabledTime for the planned time in the Data Consumer systems, and to use the consolidatedEstimatedTime for the real-time in the Data Consumer systems. (See also Figure 1 for the relation of the different time values). It is up to the Data Consumer to follow this suggestion or to deviate from that and use the planned time and/or the estimated time provided by the Data Provider. That information is also in the message that is sent by the platform. Additional trains don't have a timeTabledTime.

real-time mobility data exchange 1.0 MVP only

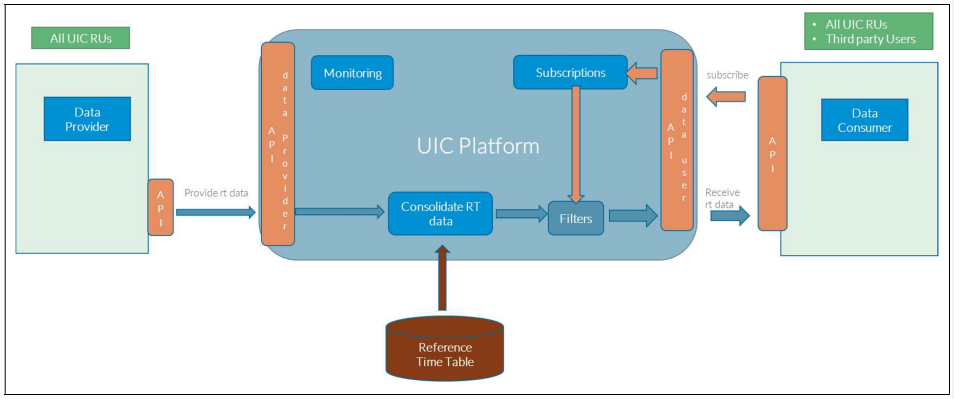
8 Architectural design

8.1 Overview of architecture

On high level the architecture is simple. A Data Provider will deliver real-time according to the standard API format of the platform [see Bibliography] . The API is developed with the information in mind to cover most of the use cases related to international trains. It is expected that an RU as Data Provider can deliver all essential information. Other Data Providers are allowed to send data as long as they fill the essential information in the API.

The platform will make consolidated Service Runs from the input of the Data Providers. The platform provides a push and a pull service to give Data Consumers the possibility to retrieve data.

The timetable reference plan serves as input for the platform to create Service Runs with consistent planned times.



It is key for the platform to be high available, which makes that essential information like already consolidated data may not get lost. Another essential part is that the system must be able to handle to load of all real-time messages from all trains in Europe.

real-time mobility data exchange 1.0 MVP only

8.2 Deployment architecture

The deployment architecture covers the adapter to receive real-time data from the RUs, the adapter to load a MERITS plan, A database to store the last status of the real-time, Log files for analysis and billing used platform services.

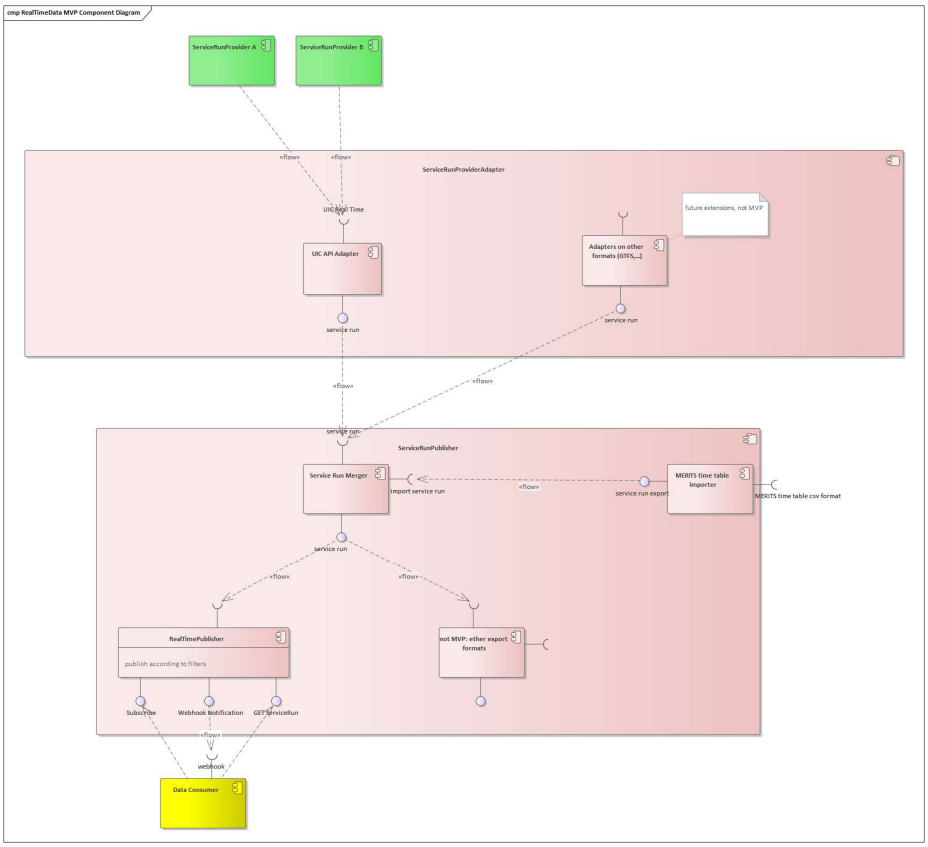


Figure 14 deployment architecture

real-time mobility data exchange 1.0 MVP only

8.3 APIs

The APIs for Data Provider and Data Consumer share the same object definition. This design was chosen as the differences between the data are small. A reference to the API definition can be found in Bibliography.

	Provider API	Consumer API
ServiceId	Yes	Yes
Stops / Delays	Yes	Yes
Train formation	Yes	Yes
Connections	Yes	Yes
Geo-Locations	Yes	Yes
Consolidated delay estimates	No	Yes
Results from the comparison to a timetable	No	Yes
Alternative ServiceRun Ids of other RUs	No	Yes

Figure 15 differences in Data Provider and Data Consumer API

8.3.1 Data Provider API

REST OPEN API

The technical specification is provided as open api specification version 3.1 (yml format).

One endpoint to send real-time data on a serviceRun is provided: POST /service/run/

- The platform service is idempotent.
- Authentication is implemented according to OAUTH 2.0.
- Communication errors are handled by the Data Provider. It is up to the Data Provider to ensure that in case of communication errors retries are implemented.

8.3.2 Data Consumer UIC RU API

REST OPEN API

The technical specification is provided as open API specification version 3.1 (yml format).

Data export is implemented as a webhook. The consumer needs to register an endpoint in a subscription where he can receive notifications on new or updated Service Runs. The provider will send notifications to the registered endpoints for new or updated Service Runs. The platform subscription service allows to define the filter options for receiving data.

The platform will provide endpoints to add, modify and delete subscriptions to consume data and to receive a list of the registered end points of a company.

- PUT /subscriptions/ …

real-time mobility data exchange 1.0 MVP only

- PATCH / subscriptions/{subscriptionId} …
- DELETE / subscriptions/{subscriptionId}
- GET / subscriptions/company={companyId}

The platform must ensure to keep the services running. To do so the platform is allowed to degrade the services for users not compliant with agreements or expected behaviour as:

- The platform might reject subscriptions if the number of registrations goes beyond the agreed limit.
- The platform might suspend a registered end point in case it is not responding according to the required service level.
- The platform will implement a retry mechanism in case of communication errors when providing notifications. The platform might abandon the delivery in case the communication errors on the end point provided by the Data Consumer goes beyond the usual limits.

The platform will provide an endpoint to retrieve the Service Run using the id provided in the notification. The handling of communication errors is up to the Data Consumer.

- GET /serviceRuns {serviceRunId} - retrieve one service
- POST /serviceRuns/get {serviceRunIds} - retrieve multiple services

The platform will provide an endpoint to search for services.

- POST /serviceRuns/search {search parameters}

The platform will provide a service to create or update platform service settings

- POST /serviceRun {serviceRun}

8.4 Monitoring

Monitoring must provide insight to the status of the system and must allow to check the agreed SLAs. It must include:

- Key figures of the SLA (uptime, performance)
- Number of Service Runs received per providers
- Number of Service Runs delivered per consumer
- Number of communication issues with web hooks provided by consumers (per consumer)

8.5 Data Objects

A complete list of all described data objects. The data structures defined in OPEN API specification [see Bibliography] are used to exchange ticket and annotation data. The master of the data structure definition is always the JSON schema! In other words: Any information in this parapraph and the related Appendix is indicative. The OPEN API specification [see Bibliography] is leading when this document and API are not equal. In Appendix C examples of Data Objects, graphical examples of the Data Objects in the API are given to help the reader understand the scope of the API.

- Service Run
- ServiceRun.DataDelivery

real-time mobility data exchange 1.0 MVP only

- ServiceRunEvent
- ServiceRunEvent.Departure
- ServiceRunEvent.Arrival
- ServiceRunEvent.PassThrough
- ServiceRunEvent.GeoTracking
- VehicleGroup
- Connection
- ServiceRunReference
- Subscription
- Filter criteria:

real-time mobility data exchange 1.0 MVP only

8.6 Code lists

8.6.1 URNs for code lists

Code List	Name Space and domain	Code List	Description	example	base path for relative references
stations	urn:uic	stn	UIC station codes (TAP-TSI retail station codes)	urn:uic:stn:8512345	urn:uic:stn:
stations	urn:rne	stn	CRD location codes (TAP-TSI infrastructure codes) Infrastructure codes must not be used in case there exists a corresponding retail code.	urn:rne:stn:8512345	urn:rne:stn:
service brands	urn:uic	sbc	UIC service brand code (TAP-TSI B.4.7009 / https://uic.org/passenger/passenger-services-group/article/service-brand-code-list)	urn:uic:sbc:17	urn:uic:sbc:
companies	urn:uic	rics:ac	company code (TAP-TSI https://www.era.europa.eu/registers/ocr_en / https://uic.org/support-activities/it/rics) and optional administration code (AC)	urn:uic:rics:1080:000011	urn:uic:rics:
countries	urn:iso	std:iso:3166	ISO Country Codes	urn:iso:std:iso:3166:CH	urn:iso:std:iso:3166:
currencies	urn:iso	std:iso:4217	ISO Currency Codes	urn:iso:std:iso:4217:CFR	urn:iso:std:iso:4217:

real-time mobility data exchange 1.0 MVP only

Code List	Name Space and domain	Code List	Description	example	base path for relative references
TAF reason for delay codes	urn:x-taf	urn:x-taf:25	TAF-TSI reason for delay codes defined by ERA (Codelist defined in UIC Leaflet 450-2)		
UIC Passenger related codes to display the reason for delay or cancellation	urn:uic	prc	See: Delay Reason Codes 8.6.2		urn:uic:prc
UIC vehicle id	urn:uic	veh			urn:uic:veh

Table 1 URNs for code lists

8.6.2 Delay Reason Codes

The proposed delay codes are aligned with the codes in the SIRI standard and can be found in _Appendix_D_.

8.6.3 Proprietary values

Be it to represent specific places, or any other type of reference data, some providers may need to extend a code set with proprietary values in order to support their use cases, while these values are unlikely to be relevant to the rest of the community. To do so, an provider should then use the following format for the proprietary values it wishes to extend the code set with:

urn: X_<3 letters code for the provider>::

real-time mobility data exchange 1.0 MVP only

For example: urn: x_zoo:paxtype:OLIPHANT

Code lists for the content of the data are references at the data structure definitions. The following general rules apply:

- Company codes

  Companies (Carriers, allocators, ...) are encoded using the UIC RICS code or the upcoming compatible EUR company code.

- Service brands
  Service brands are encoded using the UIC service brand code.

- Countries are encoded as 2A ISO country code unless it is stated differently

- Dates and time is encoded as UTC unless it is stated differently

8.7 Schema versions

The versioning is using semantic versioning:

Mayor version:

1

Minor version:

1.1

Patch version:

1.1.1

Mayor version difference indicate a breaking change.

Minor version differences indicate additional tags added or tags removed.

Patch versions differences indicate bug fixes.

8.8 Api versioning

The Api implements content negotiation for versioning:

A request specifies the accepted versions in the http Accept header parameter:

Accept: application/json; version=1, application/json; version=2

A Reply specifies the provided version in the content type:

Content-Type: application/json; version=1.0.0

or answers with http error 406.

real-time mobility data exchange 1.0 MVP only

8.9 Authentication and Authorization

The following three design principles are binding for each implementor:

1. We are using **OAuth2** (as used in ETCD and OSDM)**.**
2. The JWTs in use for the authentication should be **short-lived** (think of timeout duration single-digit multiples)
3. The JWTs sent by the consumer, regardless of where they are generated, must be **digitally signed** using a private key for which the provider is able to find the matching public key

In multi-environments (DTAP) consumers might register for each environment separately.

8.10 Services to be provided

See OPEN API YML file. [see Bibliography]

8.11 Error Handling

In order to communicate errors to a consumer the services support RFC7807.

This RFC defines a "problem detail" as a way to carry machine- readable details of errors in a HTTP response to avoid the need to define new error response formats for HTTP APIs.

A problem details object can have the following members:

- type: A URI reference RFC3986 that identifies the problem type. This specification encourages that, when dereferenced, it provide human-readable documentation for the problem type (e.g., using HTML [W3C.REC-html5-20141028]). When this member is not present, its value is assumed to be "about:blank".
- title: A short, human-readable summary of the problem type. It SHOULD NOT change from occurrence to occurrence of the problem, except for purposes of localization (e.g., using proactive content negotiation; see RFC7231, Section 3.4).
- status: The HTTP status code (RFC7231, Section 6) generated by the origin server for this occurrence of the problem.
- detail: A human-readable explanation specific to this occurrence of the problem.
- instance: A URI reference that identifies the specific occurrence of the problem. It may or may not yield further information if dereferenced.

Consumers MUST use the type string as the primary identifier for the problem type; the title string is advisory and included only for users who are not aware of the semantics of the URI and do not have the ability to discover them (e.g., offline log analysis). Consumers SHOULD NOT automatically dereference the type URI.

The following standard HTTP error codes are used in the specification:

Error Code	
400	Bad Request

real-time mobility data exchange 1.0 MVP only

401	Unauthorized
403	Forbidden
404	Not found
409	Conflict
500	Internal server error
501	Not implemented
503	Service unavailable

The API makes use of the JSON Problem structure to return information about functional errors in the handling of a request.

The problem structure is based on the Problem Details for HTTP APIs RFC RfC-7807 which defines a "problem detail" as a way to carry machine-readable details of errors in a HTTP response to avoid the need to define new error response formats for HTTP APIs.

The title property should begin with the code of the error type. This code should be the unique identifier for the functional situation in the absolute URI in the type property as well.

In order that implementations also behave consistently in error situations, the following error codes must be supported in case of functional errors by all implementations:

Functional Area	Code	Description
Subscriptions	TOO_MANY_SUBSCRIPTIONS	The number of subscriptions exceeds the agreed limits.

Table 2 functional error codes

real-time mobility data exchange 1.0 MVP only

9 Required Service Levels

Data volumes for pure geo-tracking of trains are not considered in the MVP. The requirements on service levels and scalability take into account only:

- Initial delivery of a Service Run
- Delay information and updates
- Platform changes and updates
- Updates on missed connections in case of delays

Number of services in Europe:

160.000 Service Runs per day (numbers from MERITS)

Number of messages per Service Run:

50 messages per Service Run[6]

Number of messages per consumer:

- Max. 8.000.000 messages per consumer and day

  (Production stream and smaller test stream)

- 96.000.000 messages per day sent by the platform[7]

Example lead from PKP as Data Provider:

- 1000req/5min; total: 160k/day for all RUs,
- last 24h no-0-delay: 82k ; 0-delay: 77k,
- PKP IC trains only: 25k/day (0 and no-0 delays)

9.1 Scalability

The architecture should be scalable up to 160.000 Service Runs and 40 consumers in parallel.

9.2 Service levels of the platform

The required service levels hereafter are minimal obligations in order not to discourage implementations. More ambitious agreements should be made where possible.

The web services must be available 24x7.

The required availability is 99.9% unless higher levels are agreed bilaterally.

Response times of web services (unless shorter times are agreed bilaterally):

[6] Estimation based on an analysis of data from one Data Provider on one specific day.
[7] Estimated that there will be 20 Data Consumers all filtering to receive only for them relevant messages.

real-time mobility data exchange 1.0 MVP only

1 sec for 95% of transactions

A real-time message must be processed within 10 seconds by the platform, also when a new reference plan is being processed by the platform.

9.3 Required service level of registered endpoints of data consumers

The web services must be available 24x7.

The required availability is 99.9% unless higher levels are agreed bilaterally.

Response times of web services[8] (unless shorter times are agreed bilaterally):

1 sec for 95% of transactions

2 sec for 98% of transactions

5 sec for 99.8% of transactions

Availability of real-time data on the platform is guaranteed by sharing the database over two locations.

[8] Response time is the time between the request from the Data Consumer and the sending of the requested information by the platform.

real-time mobility data exchange 1.0 MVP only

10 Bibliography

The document contains references to the following documents.

- REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)
- Reference to API specification on GitHub, GitHub - UnionInternationalCheminsdeFer/UIC-Real-Time-Mobility-Data-Exchange: api specification for the exchange of real time mobility data
- Rail PRR: Regulation (EC) 1371/2007 on Rail Passengers' Rights and Obligations
- TAP-TSI Revision
- Open Sales and Distribution Model (IRS 90918-10, osdm.io)
- In order to communicate errors to a consumer the services support RFC7807.

real-time mobility data exchange 1.0 MVP only

Appendix A Split into estimation packages

In this appendix a suggestion to divide the development in packages. Packages can be used for estimation of the work to be done to realise the platform.

Infrastructure

- Provider API
- Consumer API
- System architecture (must support scalability)
- Storage for Service Runs incl. housekeeping

Monitoring

- Monitoring of data quality
- Monitoring of service quality
- Monitoring whether providers send messages

Import of timetable

- Import of timetable
- Import of timetable - split into Service Runs
- Import of timetable - Service Runs with service number changes at the border
- Import of timetable - split into Service Runs for through coaches
- Import of timetable - split into Service Runs for multiple service numbers at the same stop

Authorization and subscription

- Registration and authentication of a Data Provider
- Registration and authentication of a Data Consumer
- UI for initial filters on Data Consumers
- Subscription service (Add, Change, Delete, List subscription)
- UI for UIC to see all subscriptions

Receiving RT message from a Data Provider

- Receiving RT message - validation
- Receiving RT message - identification of Service Run
- Receiving RT message - identification of Service Runs with different service numbers at border crossing
- Receiving RT message - identification of Service Runs for through coaches
- Receiving RT message - identification of Service Runs for multiple service numbers at the same stop
- Update of the Service Run(s) or creation of a new service
- Calculation of consolidated delay estimates

Sending RT message to Data Consumers

- Find all relevant subscriptions
- Check Filters from subscriptions and initial filters
- Sending data to the provider
- Retry in case of connection problems

real-time mobility data exchange 1.0 MVP only

- Service to get one or more Service Runs

Search for Service Run (request from Data Consumer on the API)

- Find the relevant Service Runs

real-time mobility data exchange 1.0 MVP only

Appendix B The structure of a Use case

In this document the following structure is used for Use cases.

- Title
- Description
- Use case diagram (UML) (optional)
- Trigger (optional)
- Included functionality (optional)
- Involved actors and their role (optional)
- Business rules (optional)
- Out of scope (optional)
- Additional information

real-time mobility data exchange 1.0 MVP only

Appendix C examples of Data Objects

Data Structures

The data structures defined are used to exchange ticket and annotation data. The technical data structures are defined in the OPEN API specification file according to OPEN API specification.

The data structure description given here uses a graphical notation for easy reading. The master of the data structure definition is always the JSON schema! In other words: Any information in this Appendix is indicative. The OPEN API specification [see Bibliography] is leading when this document and API are not equal.

Service Run

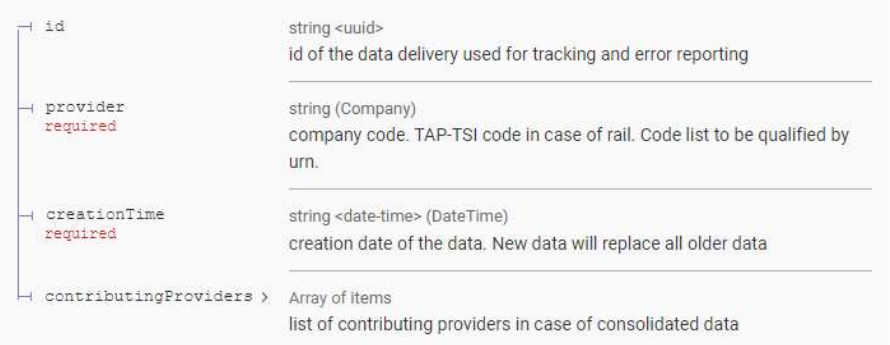
The Service Run is the main object for which real-time data are provided. It represents a single continuous run of a train service.

dataDelivery › required	object (DataDelivery) main service refence for this service run. The service reference might change along the route, the changed service refences are then included in the departiure event for the next part of the route
serviceRunRef › required	object (ServiceRunReference)
replacedServiceRuns ›	Array of objects (ServiceRunReference) list of services replaced by this service
replacingServices ›	Array of objects (ServiceRunReference) list of services replacing this service
timeTableReference ›	object (TimeTableReference) reference to a time table
events › required	Array of objects (ServiceRunEvent) list of events along the service run ordered by time
cancelled	boolean Default: `false` default false
cancellationType	string <x-extensible-enum - MISSING_FROM_CARRIER - CANCELLED_BY_CARRIER - MISSING_FROM_TIMETABLE> (CancellationType) details on the cause of cancellation. Either cancelled by the Carrier by sending the cancel information (CANCELLED_BY_CARRIER) or cancelled as the Carrier did not send the stop compared to the reference time table (MISSING_FROM_CARRIER) or the service is not any more included in a new time table update (MISSING_FROM_TIMETABLE) .
facilities	Array of strings facilities available on the whole routes of the service run. Additional facilities on specific sections are part of the VehicleStop object of the departure of the section

ServiceRun.DataDelivery

Metadata on the real-time data delivery.

real-time mobility data exchange 1.0 MVP only



ServiceRunEvent

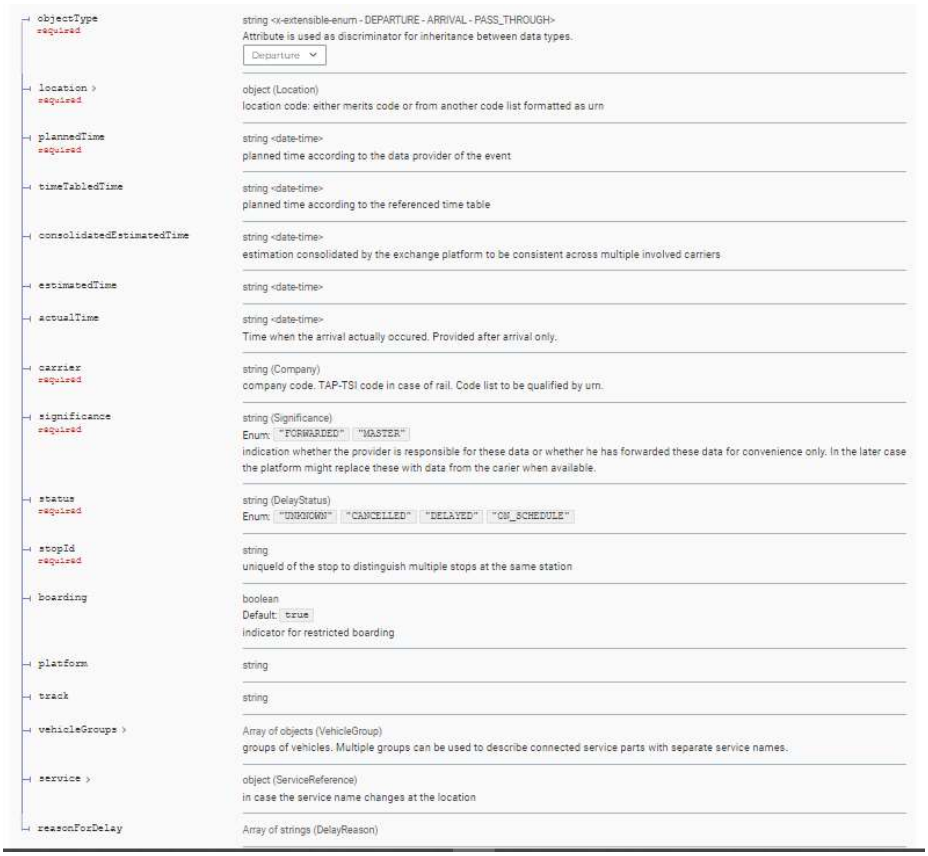
Event on a Service Run. This could be a departure, arrival or pass through.

Each event contains:

objectType	- DEPARTURE - ARRIVAL - PASS_THROUGH
location	Location id and optional the name (for stations but extensible to other types of locations).
plannedTime	Planned time for the event according to the responsible Data Provider
timeTabledTime	Planned time according to the referenced timetable e.g. MERITS 11.11.2024. For additional stops or stops in additional trains this is not provided.
estimatedTime	Estimated time according to the responsible Data Provider
consolidatedEstimatedTime	Consolidated estimated time calculated by the exchange platform to avoid inconsistencies
actualTime	Time when the event actually happened
status	Status of the event (CANCELLED, DELAYED,...)
reasonForDelay	Reasons for a delay or cancellation of the event

real-time mobility data exchange 1.0 MVP only

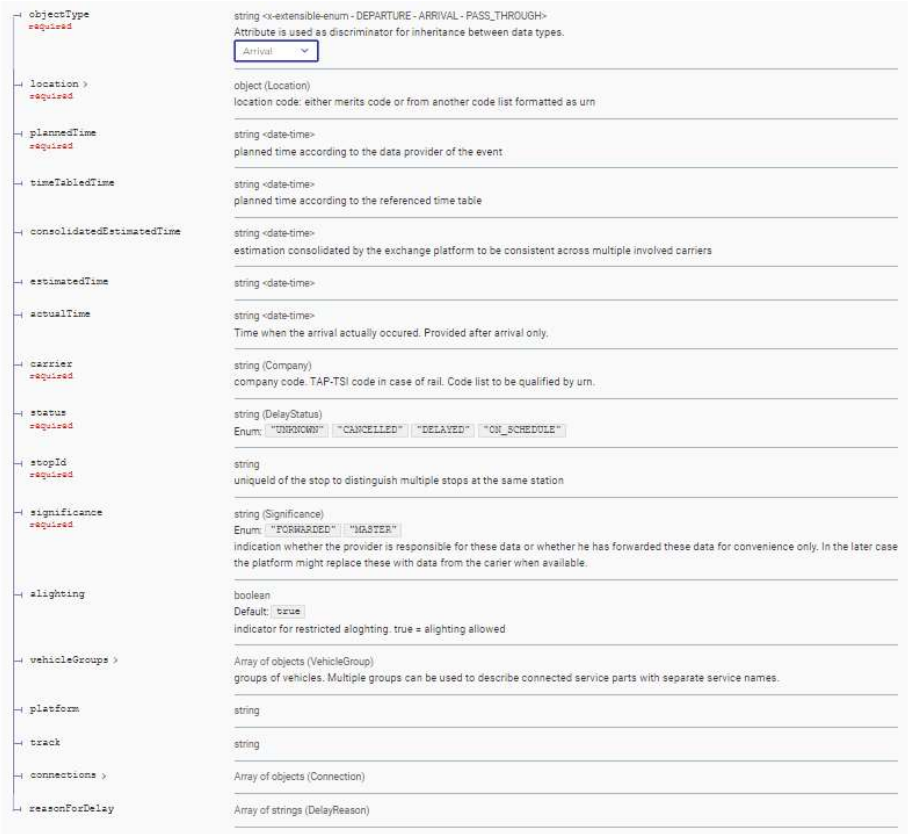
ServiceRunEvent.Departure



objectType required	string <x-extensible-enum - DEPARTURE - ARRIVAL - PASS_THROUGH> Attribute is used as discriminator for inheritance between data types. Departure
location > required	object (Location) location code: either merits code or from another code list formatted as urn
plannedTime required	string <date-time> planned time according to the data provider of the event
timeTabledTime	string <date-time> planned time according to the referenced time table
consolidatedEstimatedTime	string <date-time> estimation consolidated by the exchange platform to be consistent across multiple involved carriers
estimatedTime	string <date-time>
actualTime	string <date-time> Time when the arrival actually occured. Provided after arrival only.
carrier required	string (Company) company code. TAP-TSI code in case of rail. Code list to be qualified by urn.
significance required	string (Significance) Enum: "FORWARDED" "MASTER" indication whether the provider is responsible for these data or whether he has forwarded these data for convenience only. In the later case the platform might replace these with data from the carier when available.
status required	string (DelayStatus) Enum: "UNKNOWN" "CANCELLED" "DELAYED" "ON_SCHEDULE"
stopId required	string uniqueId of the stop to distinguish multiple stops at the same station
boarding	boolean Default: true indicator for restricted boarding
platform	string
track	string
vehicleGroups >	Array of objects (VehicleGroup) groups of vehicles. Multiple groups can be used to describe connected service parts with separate service names.
service >	object (ServiceReference) in case the service name changes at the location
reasonForDelay	Array of strings (DelayReason)

real-time mobility data exchange 1.0 MVP only

ServiceRunEvent.Arrival



objectType required	string <x-extensible-enum - DEPARTURE - ARRIVAL - PASS_THROUGH> Attribute is used as discriminator for inheritance between data types. Arrival
location > required	object (Location) location code: either merits code or from another code list formatted as urn
plannedTime required	string <date-time> planned time according to the data provider of the event
timeTabledTime	string <date-time> planned time according to the referenced time table
consolidatedEstimatedTime	string <date-time> estimation consolidated by the exchange platform to be consistent across multiple involved carriers
estimatedTime	string <date-time>
actualTime	string <date-time> Time when the arrival actually occured. Provided after arrival only.
carrier required	string (Company) company code. TAP-TSI code in case of rail. Code list to be qualified by urn.
status required	string (DelayStatus) Enum: "UNKNOWN" "CANCELLED" "DELAYED" "ON_SCHEDULE"
stopId required	string uniqueId of the stop to distinguish multiple stops at the same station
significance required	string (Significance) Enum: "FORWARDED" "MASTER" indication whether the provider is responsible for these data or whether he has forwarded these data for convenience only. In the later case the platform might replace these with data from the carier when available.
alighting	boolean Default: true indicator for restricted alighting. true = alighting allowed
vehicleGroups >	Array of objects (VehicleGroup) groups of vehicles. Multiple groups can be used to describe connected service parts with separate service names.
platform	string
track	string
connections >	Array of objects (Connection)
reasonForDelay	Array of strings (DelayReason)

real-time mobility data exchange 1.0 MVP only

ServiceRunEvent.PassThrough

objectType required	string <x-extensible-enum - DEPARTURE - ARRIVAL - PASS_THROUGH> Attribute is used as discriminator for inheritance between data types. PassThrough
location > required	object (Location) location code: either merits code or from another code list formatted as urn
plannedTime required	string <date-time> planned time according to the data provider of the event
timeTabledTime	string <date-time> planned time according to the referenced time table
consolidatedEstimatedTime	string <date-time> estimation consolidated by the exchange platform to be consistent across multiple involved carriers
estimatedTime	string <date-time>
actualTime	string <date-time> Time when the arrival actually occured. Provided after arrival only.
status	string (DelayStatus) Enum: "UNKNOWN" "CANCELLED" "DELAYED" "ON_SCHEDULE"
type required	string <x-extensible-enum - STATION - IM_HAND_OVER - IM_POINT>
fromIM	string (Company) company code. TAP-TSI code in case of rail. Code list to be qualified by urn.
toIM	string (Company) handover from this infratstucture manager
reasonForDelay	Array of strings (DelayReason) handover to this infrastructure manager

ServiceRunEvent.GeoTracking

For future use it is foreseen that a Carrier can provide geo locations of the train service. In case of through coaches or wing trains the geo locations would not form a line but a fork.

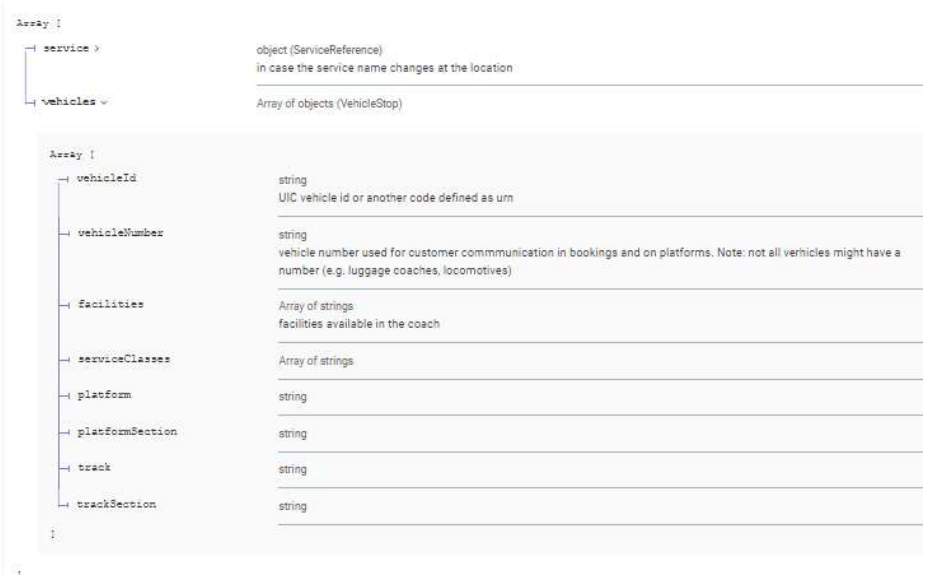
objectType required	string Attribute is used as discriminator for inheritance between data types. GeoTracking
time required	string <date-time> time when the service was at the provided position
position > required	object (GeoPosition) WGS84 coordinates position. Provided by OJP.
vehicleGroups >	Array of objects (VehicleGroup) groups of vehicles where this geo-position applies

real-time mobility data exchange 1.0 MVP only

VehicleGroup

Vehicle Groups can be used to provide data on vehicles included and their position at the track and platform

Vehicle groups can be used to provide data on coupled train parts with separate train service names and on through coaches that continue their run under another train service.



Connection

Connections describe connections to other train services that a traveller can get on arrival. Connection information can be provided and will be forwarded by the platform. Connections will not be validated or created by the platform.

ServiceRunReference

Train service attributes used to identify a Service Run.

The serviceId is provided by the platform in case the train service is already known in the platform. This might not always be the case as is is a normal case that services referenced in a connection or for a through coach reference a service that has not jet been provided to the platform.

real-time mobility data exchange 1.0 MVP only

serviceRunId	string unique id of the service run in the real time platform
serviceName required	string commercial servide name (in case of trains the train number) used for communication with the customer. Eigther service or line number must be provided
lineName	string
lineserviceRunId	string internal service number in case only the line number is used in customer communication
serviceBrands >	Array of objects (ServiceBrand) service brands in case the service is maketed under different brands
timeTableReferences >	Array of objects (TimeTableserviceRunId) references of this service to a technical service id used in a time table
providerServiceRunIds >	Array of objects service ids of the data providers for this service

serviceId	string unique id of the service run in the real time platform
serviceName required	string commercial servide name (in case of trains the train number) used for communication with the customer. Eigther service or line number must be provided
lineName	string
lineServiceID	string internal service number in case only the line number is used in customer communication
serviceBrand	string main service brand code of urn providing a code from another code list.
serviceBrandAbreviation	string
additionalServiceBrands >	Array of objects additional service brands in case the service is maketed under different brands
timeTableReferences >	Array of objects (TimeTableServiceId) references of this service to a technical service id used in a time table
providerServiceIds >	Array of objects service ids of the data providers for this service

real-time mobility data exchange 1.0 MVP only

Subscription

The subscription provides the end-point specification for receiving data. Additionally, the subscription provides the filter criteria to select relevant data.

id required	string <uuid>
company required	string (Company) company code. TAP-TSI code in case of rail. Code list to be qualified by urn.
hook required	string <uri>
user required	string
accessToken required	string
acceptedDelayTime	int <int32> accepted ealy in seconds to allow Notifications on a collection of servides to reduce the number of messages.
filters >	Array of objects (Filter) listed filters apply separately (OR)

real-time mobility data exchange 1.0 MVP only

Filter criteria:

includedCarriers	Array of strings (Company) applies in case one of these companies is the carrier on some part of the service
excludedCarriers	Array of strings (Company) applies in case the sevice is run by these carriers only
includedCountries	Array of strings (Country) applies in case one of these countries is part of the route of the service
excludedCountries	Array of strings (Country) applies in case the service is run in these countries only
countryBorderCrossings	Array of strings (CountryBorderCrossing) [items = 2 items] applies in case the service is crossing the border between the specified countries in one of the border crossings
multiCarrierServicesOnly	boolean Default: `false` data will be provided for multi carrier services only
countryCrossBorderOnly	boolean Default: `false` data will be provided for services crossing country borders only

]

real-time mobility data exchange 1.0 MVP only

Appendix D Delay Reason Codes

The proposed delay codes are aligned with the codes in the SIRI standard.

reason for delay or cancellation	description
ACCIDENT	Due to an accident
ACCIDENT_PERSON_HIT_BY_TRAIN	
ACCIDENT_PERSON_ILL_ON_VEHICLE	
ACCIDENT_PERSON_UNDER_TRAIN	
BORDER_ISSUE	Due to an issue at border crossing
BORDER_CONTROL	Due to a border control (not in SIRI)
BORDER_CUSTOM_ISSUE	Due to an issue at custom control
EQUIPMENT_BREAK_DOWN	
EQUIPMENT_BROKEN_RAIL	
EQUIPMENT_CLOSED_FOR_MAINTENANCE	
EQUIPMENT_DEFECTIVE_FIRE_ALARM_EQUIPMENT	
EQUIPMENT_DEFECTIVE_TRAIN	
EQUIPMENT_DEICEING_WORK	
EQUIPMENT_ENGINE_FAILURE	
EQUIPMENT_FAILURE	
EQUIPMENT_FUEL_PROBLEM	
EQUIPMENT_FUEL_SHORTAGE	
EQUIPMENT_GANGWAY_PROBLEM	
EQUIPMENT_LACK_OF_OPERATIONAL_STOCK	
EQUIPMENT_LIGHTING_FAILURE	
EQUIPMENT_LUGAGE_CAROUSEL_PROBLEM	
EQUIPMENT_MAINTENANCE_WORK	
EQUIPMENT_POWER_PROBLEM	
EQUIPMENT_REPAIR_WORK	
EQUIPMENT_SWING_BRIDGE_FAILURE	
EQUIPMENT_TECHNICAL_PROBLEM	
EQUIPMENT_TRACTION_FAILURE	
EQUIPMENT_TRAIN_DOOR_INCIDENT	
EQUIPMENT_TRAIN_WARNING_SYSTEM_PROBLEM	
EQUIPMENT_WHEEL_PROBLEM	
EQUIPMENT_OVERRUN	Due to too many passengers on a train
FATALITY_COLLISSION	
FATALITY_EMERGENCY_SERVICES	
FATALITY_LINE_SIDE_FIRE	
FATALITY_TRAIN_STRUCK_ANIMAL	
FATALITY_TRAIN_STRUCK_OBJECT	
INFRASTRUCTURE	
INFRASTRUCTURE_BRIDGE_STRIKE	
INFRASTRUCTURE_CONSTRUCTION_WORK	

real-time mobility data exchange 1.0 MVP only

INFRASTRUCTURE_DEFECTIVE_PLATFORM_EDGE_DOORS	
INFRASTRUCTURE_DEFECTIVE_ANNOUNCEMENT_SYSTEM	
INFRASTRUCTURE_DERAILMENT	
INFRASTRUCTURE_EMERGENCY_ENGINEERING_WORK	
INFRASTRUCTURE_LANDSLIDE	
INFRASTRUCTURE_LATE_FINISH_TO_ENGINEERING_WORK	
INFRASTRUCTURE_LEVEL_CROSSING_FAILURE	
INFRASTRUCTURE_NEXT_TRACK_SECTION_BLOCKED	(not in SIRI)
INFRASTRUCTURE_NEXT_STATION_ENTRY_BLOCKED	(not in SIRI)
INFRASTRUCTURE_OVERHEAD_OBSTRUCTION	
INFRASTRUCTURE_OVERHEAD_WIRE_FAILURE	
INFRASTRUCTURE_POOR_RAIL_CONDITIONS	
INFRASTRUCTURE_ROAD_CLOSED	
INFRASTRUCTURE_ROAD_WORKS	
INFRASTRUCTURE_ROADSIDE_ISSUE	
INFRASTRUCTURE_ROUTE_DIVERSION	
INFRASTRUCTURE_ROUTE_BLOCKAGE	
INFRASTRUCTURE_SERVICE_INDICATOR_FAILURE	
INFRASTRUCTURE_SIGNAL_AND_SWITCH_FAILURE	
INFRASTRUCTURE_SIGNAL_FAILURE	
INFRASTRUCTURE_SIGNAL_PASSED_AT_DANGER	
INFRASTRUCTURE_SIGNAL_PROBLEM	
INFRASTRUCTURE_SLIPPERY_TRACK	
INFRASTRUCTURE_TRACK_CIRCUIT_PROBLEM	
INFRASTRUCTURE_TRACK_WORKS	(not in SIRI)
INFRASTRUCTURE_TRACKSIDE_ISSUE	(not in SIRI)
INFRASTRUCTURE_TRACKSIDE_WORKS	(not in SIRI)
INFRASTRUCTURE_TRAFFIC_MANAGEMENT_SYSTEM_FAILURE	
OBSTRUCTION	
OBSTRUCTION_ANIMAL_ON_THE_LINE	
OBSTRUCTION_DEMONSTRATION	
OBSTRUCTION_FALLEN_TREE	
OBSTRUCTION_LEVEL_CROSSING_INCIDENT	
OBSTRUCTION_MARCH	Due to a public event „march“
OBSTRUCTION_OBJECT_ON_THE_LINE	
OBSTRUCTION_OVERCROWDED	
OBSTRUCTION_PERSON_ON_THE_LINE	
OBSTRUCTION_PROCESSION	
OBSTRUCTION_PUBLIC_DISTURBANCE	
OBSTRUCTION_ROAD_WORKS	
OBSTRUCTION_SIGHT_SEERS	
OBSTRUCTION_SPECIAL_EVENT	
OBSTRUCTION_STATION_OVERRUN	
OBSTRUCTION_VEGETATION	Vegetation obstructing the line
OBSTRUCTION_VEHICLE_ON_THE_LINE	Vehicle blocking the line

real-time mobility data exchange 1.0 MVP only

OPERATOR_CEASED_TRADING	
OPERATOR_INSUFFICIENT_DEMAND	Due to insufficient demand (on demand services)
OPERATOR_SUSPENDED	
PERSONELL_ISSUE	
PERSONELL_STAFF_ABSENCE	
PERSONELL_STAFF_IN_WRONG_PLACE	
PERSONELL_STAFF_SHORTAGE	
PERSONELL_STAFF_SICKNESS	
PERSONELL_STRIKE	
PREVIOUS_DISTURBANCES	
PREVIOUS_TRAIN_DELAY	(not in SIRI)
SECURITY	
SECURITY_AIR_RAID	
SECURITY_ATTACK	
SECURITY_BMB_ALERT	
SECURITY_BOMB_EXPLOSION	
SECURITY_CIVIL_EMERGENCY	
SECURITY_EMERGENCY_SERVICE_CALL	
SECURITY_EVACUATION	
SECURITY_EXPLOSION	
SECURITY_EXPLOTION_HAZARD	
SECURITY_FIRE	
SECURITY_FIRE_BRIGADE_ORDER	
SECURITY_FIRE_BRIGADE_SAFETY_CHECKS	
SECURITY_FIRE_SAFETY_CHECKS	(not in SIRI)
SECURITY_GUNFIRE_ON_ROADWAY	
SECURITY_POLICE_RECQUEST	
SECURITY_SABOTAGE	
SECURITY_SAFETY_VIOLATION	
SECURITY_SECURITY_ALERT	
SECURITY_SUSPECT_VEHICLE	
SECURITY_TELEPHONED_THREAT	
SECURITY_TERRORIST_INCIDENT	
SECURITY_UNATTENDED_BAG	
SERVICE_FAILURE	
UNDEFINED	The delay reason is known but does not fit to any code.
UNKNOWN	The delay reason is unknown.
VANDALISM	
VANDALISM_ALTERCATION	
VANDALISM_ASSAULT	
VANDALISM_PASSENGER_ACTION	
VANDALISM_RAILWAY_CRIME	
VANDALISM_STAFF_ASSAULT	
VANDALISM_THEFT	

real-time mobility data exchange 1.0 MVP only

WAITING_FOR_DELAYED_PASSENGERS	(not in SIRI)
WEATHER	
WEATHER_FALLEN_LEAVES	
WEATHER_FALLEN_TREE	
WEATHER_FLOODING	
WEATHER_FOG	
WEATHER_FROZEN	
WEATHER_HAIL	
WEATHER_HEAVY_RAIN	
WEATHER_HEAVY_SNOWFALL	
WEATHER_HIGH_TEMPERATURES	
WEATHER_HIGH_TIDE	
WEATHER_HIGH_WATER_LEVEL	
WEATHER_ICE	
WEATHER_LOW_TIDE	
WEATHER_LOW_WATER_LEVEL	
WEATHER_ROUGH_SEA	
WEATHER_STRONG_WINDS	
WEATHER_TIDAL_RESTRICTIONS	
WEATHER_WATER_LOGGED	

real-time mobility data exchange 1.0 MVP only

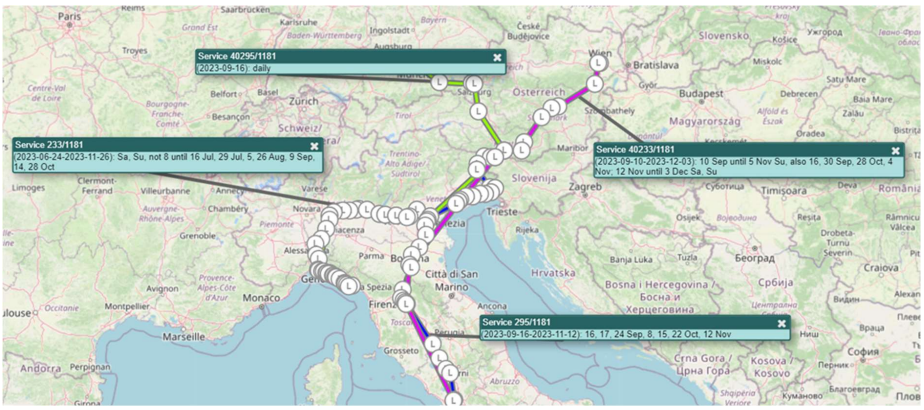
Appendix E complex example of through coaches and wing trains

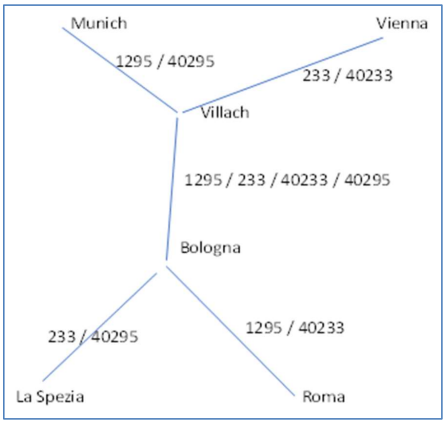
The basic concept of through coaches and wing trains is described in Merging of through coaches and wing trains. In this appendix a complex example of a combination of service runs that offer trips with multiple origins and multiple destinations. The ultimate test for checking if the platform solution for through coaches and wing trains covers all relevant real life manifestations of these trains, is this use case. It is also a test if different ways of communicating through coaches and wing trains by different RUs can be unified in the defines platform API.

In the example below travellers can make the following trips:

- Munich-Rome
- Munich-La Spezia
- Vienna-Rome
- Vienna-La Spezia

In the OBB data format before conversion to API definition of the platform there are 4 service runs: 40295, 40233, 233 and 295. So for every connection there is a service run. (In the figure the number 1181 is the identification of a specific Data Provider.)

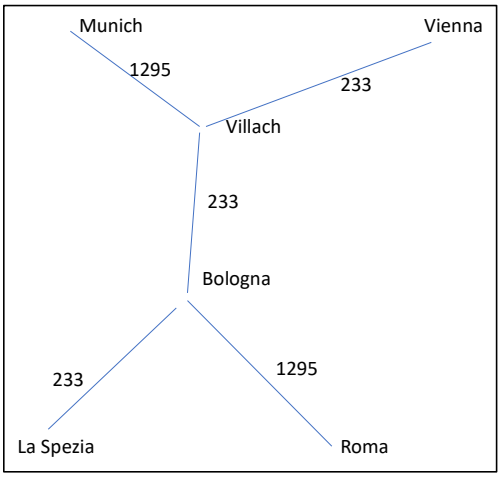




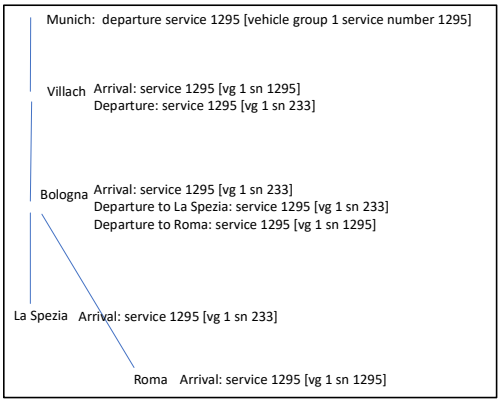
In the Tren Italia data before conversion to the platform API format there are two trains:

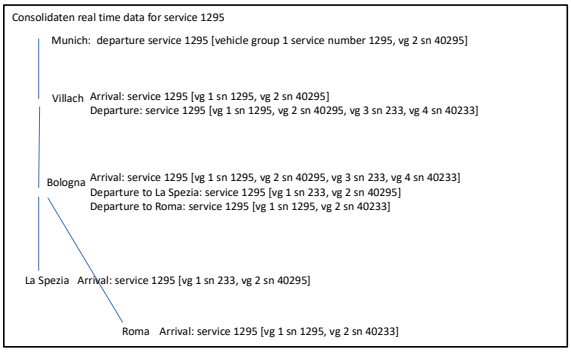
- Munich – Rome
- Vienna – La Spezia

In the data there is specified that there is a connection between these trains in for example Villach. There is no information on coaches in a service run for these trains.



real-time mobility data exchange 1.0 MVP only





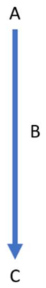
real-time mobility data exchange 1.0 MVP only

Appendix F Simple example of through coaches and wing trains

The following messages give a simplified insight in how joining and splitting will be handled by the platform.

Number change, customer view

Real-time data represents the customers'point of view. For each bookable service, a real-time service is created. The main serviceRef is the number known by customers (booking number), while serviceRunRef is used to connect trains together. Only the number used for booking is used, here Blue. No reference to the red train, since it does not exist.

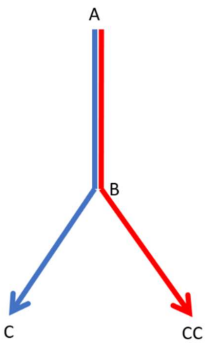



```
{
  "serviceRef": {
    "serviceName": "blue"
  },
  "events": [
    {
      "objectType": "Departure",
      "location": {
        "name": "A"
      }
    },
    {
      "objectType": "Arrival",
      "location": {
        "name": "B"
      }
    },
    {
      "objectType": "Departure",
      "location": {
        "name": "B"
      }
    },
    {
      "objectType": "Arrival",
      "location": {
```

```
        "name": "C"
      }
    }
  ]
}
```

Splitting customer view

Real-time data represents the customers' point of view. For each bookable service (in the example from A to C and from A to CC), a real-time service is created. The main serviceRef is the number known by customers (booking number), while serviceRunRef is used to connect trains together.




```
{
  "serviceRef": {
    "serviceName": "blue"
  },
  "events": [
    {
      "objectType": "Departure",
      "location": {
        "name": "A"
      },
      "vehicleGroups": [
        {
          "serviceRunRef": {
            "serviceName": "blue"
          }
        },
        {
```

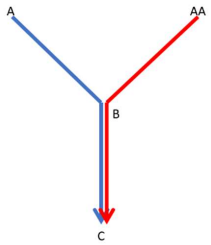
```
          "serviceRunRef": {
            "serviceName": "red"
          }
        }
      ]
    },
    {
      "objectType": "Arrival",
      "location": {
        "name": "B"
      },
      "vehicleGroups": [
        {
          "serviceRunRef": {
            "serviceName": "blue"
          }
        },
        {
          "serviceRunRef": {
            "serviceName": "red"
          }
        }
      ]
    },
    {
      "objectType": "Departure",
      "location": {
        "name": "B"
      },
    },
    {
      "objectType": "Arrival",
      "location": {
        "name": "C"
      },
    }
  ]
}


{
  "serviceRef": {
    "serviceName": "red"
  },
  "events": [
    {
      "objectType": "Departure",
      "location": {
```

```
      "name": "A"
    },
    "vehicleGroups": [
      {
        "serviceRunRef": {
          "serviceName": "blue"
        }
      },
      {
        "serviceRunRef": {
          "serviceName": "red"
        }
      }
    ]
  },
  {
    "objectType": "Arrival",
    "location": {
      "name": "B"
    },
    "vehicleGroups": [
      {
        "serviceRunRef": {
          "serviceName": "blue"
        }
      },
      {
        "serviceRunRef": {
          "serviceName": "red"
        }
      }
    ]
  },
  {
    "objectType": "Departure",
    "location": {
      "name": "B"
    },
  },
  {
    "objectType": "Arrival",
    "location": {
      "name": "CC"
    },
  }
]
```

}

Joining customer view

Real-time data represents the customers' point of view. For each bookable service (in the example for A to C and AA to C), a real-time service is created. The main serviceRef is the number known by customers (booking number), while serviceRunRef is used to connect trains together.




```
{
  "serviceRef": {
    "serviceName": "blue"
  },
  "events": [
    {
      "objectType": "Departure",
      "location": {
        "name": "A"
      }
    },
    {
      "objectType": "Arrival",
      "location": {
        "name": "B"
      }
    },
    {
      "objectType": "Departure",
      "location": {
        "name": "B"
      },
      "vehicleGroups": [
        {
          "serviceRunRef": {
            "serviceName": "blue"
          }
        },
        {
```

```
          "serviceRunRef": {
            "serviceName": "red"
          }
        }
      ]
    },
    {
      "objectType": "Arrival",
      "location": {
        "name": "C"
      },
      "vehicleGroups": [
        {
          "serviceRunRef": {
            "serviceName": "blue"
          }
        },
        {
          "serviceRunRef": {
            "serviceName": "red"
          }
        }
      ]
    }
  ]
}

{
  "serviceRef": {
    "serviceName": "red"
  },
  "events": [
    {
      "objectType": "Departure",
      "location": {
        "name": "A"
      }
    },
    {
      "objectType": "Arrival",
      "location": {
        "name": "B"
      }
    },
    {
      "objectType": "Departure",
      "location": {
        "name": "B"
      },
      "vehicleGroups": [
        {
          "serviceRunRef": {
            "serviceName": "blue"
          }
        },
        {
          "serviceRunRef": {
            "serviceName": "red"
          }
        }
      ]
    },
```

real-time mobility data exchange 1.0 MVP only


```
    {
      "objectType": "Arrival",
      "location": {
        "name": "CC"
      },
      "vehicleGroups": [
        {
          "serviceRunRef": {
            "serviceName": "blue"
          }
        },
        {
          "serviceRunRef": {
            "serviceName": "red"
          }
        }
      ]
    }
  ]
}
```