Contract No. H2020 – 636078

INFORMATION TECHNOLOGIES FOR SHIFT TO RAIL

D4.2 – Trip Tracker Specifications

Due date of deliverable: 31/10/2017

Actual submission date: 20/07/2018

Leader of this Deliverable: OLTIS Group

Reviewed: Y

Document status		
Revision	Date	Description
1	22.10.2015	First draft (v0)
1.1	11.12.2015	First issue (v1)
1.2	18.04.2016	Core Release version (v2)
1.3	27.10.2016	Additional Release version (v3)
1.4	31.07.2017	Final Release draft (v4)
1.5	03.07.2018	Final Release final version (v5)
2	20.07.2018	Final version after TMC approval and Quality check

Project funded from the European Union's Horizon 2020 research and innovation programme

Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/05/2015 Duration: 36 months

REPORT CONTRIBUTORS

Name	**Company**	**Details of Contribution**
Petr Buchníček Petra Juránková	OLTIS Group	Entire document
Gianpaolo Cugola Matteo Rossi	PoliMi	Complex Event Processing MatchPatterns and EvaluateConflicts
Leyre Merle Carrera	INDRA	BA interface
Cathy Minciotti Guido Mariotta	Leonardo	BA Cooperation, EvaluateConflicts
Elvino Monteiro	Thales Portugal	Events Listening
Achim von der Embse	HaCon	Manage Alternatives
Jens Unger Flavia Saplacan	HERE	Logs repository and logging functionality
Sebastian Pretzsch	Fraunhofer	UpdatePatterns
Marco Ferreira	Thales Portugal	Overall overview
Maria Laura Trifiletti	RINA C-BE	Quality check

EXECUTIVE SUMMARY

This document presents the current status of the specifications of the Trip Tracker module of the IT2Rail project. Through functional scenarios it describes the Trip Tracker's behaviour in several use cases, its capabilities, relations with external actors as well as interfaces between different components. It also includes the decomposition of the Trip Tracker's components into logical functions and functional exchanges. This document also shows how these functionalities interact, among them and also with other IT2Rail modules, to achieve the goals of Trip Tracker.

The terminology used in this document is based on the IT2Rail ontology.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The goal of Trip Tracker functional area lies in monitoring of irregularities in transport and in responding to such anomalies in on-line mode, including suggestions of alternative solutions. So far, journey planners have focused on searching the optimal connection in compliance with planned timetables. Life, however, brings a large number of events that invoke irregularities in transport. Ultimately, this means that a Traveller cannot meet the original travel plan, and has to react to the current situation. Then a lot of bad decisions are made due to insufficient amount of information, or due to their too late acquisition.

The main role of Trip Tracker, in the integration with the other IT2Rail modules, is to inform the Traveller about events (conflicts) which may affect the Traveller’s scheduled itinerary and for that reason the Trip Tracker will take the current traffic situation into account. The Trip Tracker should therefore be in perspective the natural link to other modules and components in order to be able to provide the user with real-time information about the traffic for the current and future legs of his/her journey. The Trip Tracker should also become the solution which is open to data interchange with Traffic Management System or maintenance systems.

After activation of tracking and obtaining all necessary information such as the Traveller’s preferences or itineraries, the Traveller will be informed on all known travel conflicts that may affect the planned journey. Evaluation of travel conflicts will run since its activation until the end of travel. The system notifies the Traveller of possible risks of deviation from original plans. Depending on the impact on the travel plan, different types of messages (Information, Warning or Alert) are sent. Moreover, the system also offers alternative solutions to the Traveller and thus enables him/her to continue travelling in case of disruption. All important data are stored in logs and available for further processing by the Business Analytics.

2. REFERENCED DOCUMENTS

This section lists the document reference number, title, revision, and date of all documents referenced in the specifications document.

Reference Number	Title	Revision	Date
	IT2RAIL-Proposal_second stage_SECTION 1-3_28082014_.pdf	1	28.08.2014

Table 2-1: Referenced documents

3. PERATIONAL ASPECTS

This chapter describes the position and the role of the Trip Tracker within of the other IT2Rail modules. After laying out some key principles followed in the design of Trip Tracker, it presents the key actors with which Trip Tracker interacts, thus the Traveller and the Travel Event Provider.

3.1 PRINCIPLES

The Trip Tracker aims to design and develop a modular solution, which will be able to expand monitoring of European traffic irregularities and transport systems failures within a simplified possible implementation in accordance with the IT2Rail use case. Such a solution must be maximally interoperable, and of course, it must be able to process a huge amount of real-time data too.

In evaluation of events and analysis of their impact to transport, the Trip Tracker will deal with all main transport modes (rail, road and air transport). The close cooperation of the Trip Tracker and other IT2Rail functional areas also represents an important part of the solution. It will help to sustain a qualitative growth of services provided in public transport and thus to make public transport more attractive in comparison to the individual transport.

3.2 SCOPE/PURPOSE

Trip Tracker monitors relevant events available on the ‘web of transportation things’ that could affect the Traveller’s journey. Matching those events with the preferences and door-to-door itineraries, stored in the Travel Companion, Trip Tracker aims to provide the user with a shield against travel disruptions by enabling seamless re-arrangement. To this end, Trip Tracker may automatically invoke Travel Shopper and Ticketing to create alternate itineraries, and, through the Travel Companion display alerts, offer alternative routings and trigger re-accommodation where applicable or desired.

The main objectives of the proposed solution are to:

- Listen to the on-line information on traffic irregularities (real-time data).
- Alert Travellers on all relevant events concerning their planned journey.
- Propose alternative solutions in reaction to such exceptions.

Trip Tracker, activated on user’s request, uses the Complex Event Processing (CEP) techniques to identify travel conflicts and their impact on individual transport connections, taking preferences of each user into account. Events will be obtained from many different sources. Events will be obtained from many different sources. On one hand, there are information provided by IMs (Infrastructure Managers), on the other hand, there are channels from central dispatching systems of individual RUs (Railway Undertakings) or transport companies of other transportation modes. It requires a large amount of analytical work to consolidate information coming from such different sources.

When analysing the travel conflict, the Trip Tracker cooperates with the Travel Shopper on getting alternative solutions for the Traveller. These suggestions are forwarded to the Travel Companion of the relevant Customer and related data are provided at the same time to enable re-accommodation when necessary.

3.3 DESIGN DRIVERS

The main principle that leads the design and implementation of the Trip Tracker module is interoperability. All IT2Rail modules are designed in order to build a self-consistent platform able to add ‘on-the-fly’ new components that enrich the abilities of the whole system. Designing a system with these features means to meet specific requirements in terms of interoperability, both syntactic and semantic. The former aims at allowing heterogeneous systems to communicate each other by exchanging information by means of specified data formats and communication protocols. The latter enables systems to automatically understand the information exchanged in terms of meaning in order to produce useful results for IT2Rail users.

3.4 ACTORS AND USE CASES

3.4.1 Actors

The external actors, which are in relation to the Trip Tracker capabilities, are the following:

- **Traveller:** the generic IT2Rail user who travels by using any transportation means. Traveller uses his/her smart device in order to plan all activities related to his/her travel.
- **Travel Event Provider:** an organization which is able to broadcast information on various irregularities in transport (events), e.g. infrastructure managers or railway undertakings in the railway ecosystem.

3.4.2 Context

Trip Tracker aims at satisfying Traveller’s needs during his/her travel, especially when any irregularity occurs. The goal of trip tracking lies in monitoring of these irregularities and in responding to such anomalies in on-line mode, including suggestions of alternative solutions.

Trip Tracker does not interact with a Traveller directly, but only through the Travel Companion Personal Application. When activated on user’s request, Trip Tracker retrieves all the necessary information from Travel Companion Cloud Wallet. Alternative solutions are proposed to the Traveller through the Travel Companion too.

To be able to invoke the listening to the on-line information on traffic irregularities, Trip Tracker cooperates with the Events Sources Resolver of the Interoperability Framework. When the validation of Traveller’s itinerary is necessary, or alternative solution needs to be offered, Trip Tracker relies on Travel Shopper capabilities. Alternative offers are always requested from the provider of the original travel plan first. If another Offer Provider needs to be requested, then Trip Tracker relies on the appropriate resolver of the Interoperability Framework as well.

Trip Tracker also provides data to the Business Analytics module in asynchronous mode.

3.4.3 Use cases

The main mission of the Trip Tracker is to track journeys scheduled by the Traveller and to keep him/her informed on all known travel conflicts that may affect the planned journeys. Therefore, the Trip Tracker must implement at least the following capabilities:

- **Activate Tracking** – includes the Resolve Events Sources capability of the Interoperability Framework to enable the events listening invocation upon Traveller's request.
- **Perform Event Processing** – the ability to respond to events broadcasted by the Travel Event Provider.
- **User Interface** – supports sending messages to the Travel Companion from any of Trip Tracker components.
- **Manage Alternatives** – extends the event processing by the service of suggestions of alternative solutions.
- **BA Cooperation** – supports the asynchronous provision of data to Business Analytics.

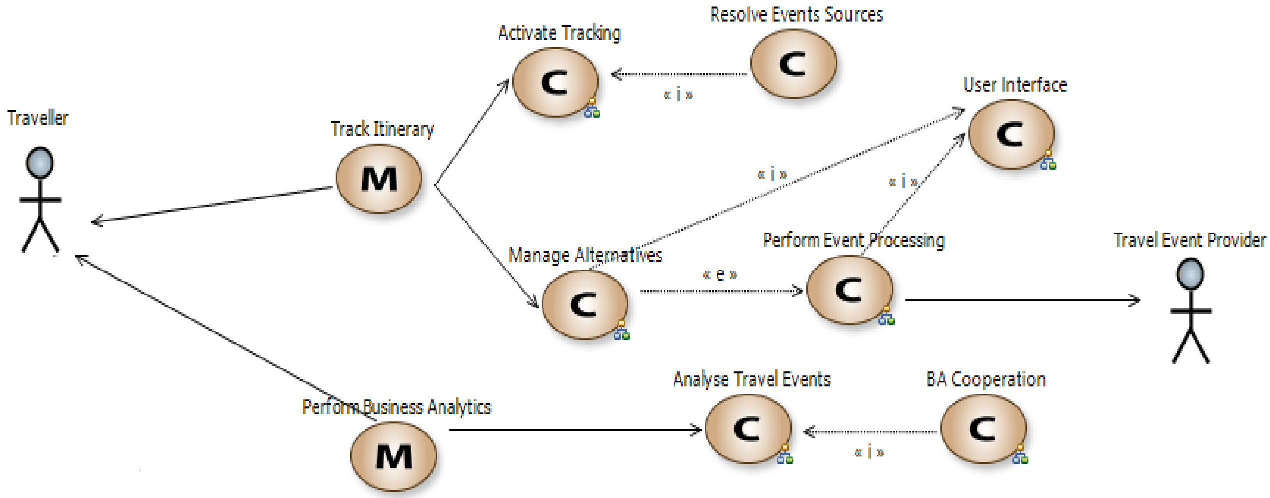

Figure 3-1: Trip Tracker capabilities

4. FUNCTIONAL ASPECTS

4.1 COMPONENTS

Trip Tracker module is composed of several standalone components, as shown in the following diagram. These components will be developed iteratively with a view to complete the Trip Tracker Proof of Concept.

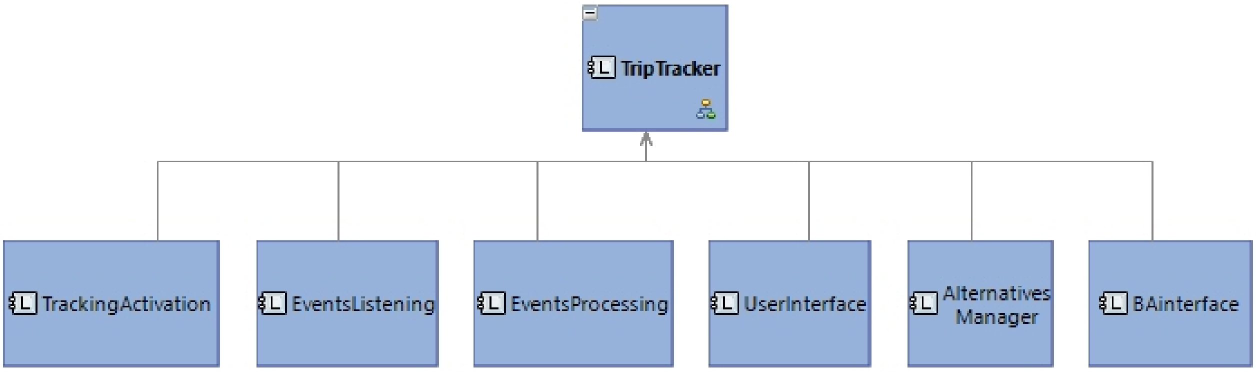

Figure 4-1: Trip Tracker components

The aim of **TrackingActivation** component is to support the implementation of the Activate Tracking capability. It implements an interface to Travel Companion, which is designed to retrieve all necessary information required to enable tracking activation. It also manages the validation of itinerary (interface to Travel Shopper). To fully achieve its goal, it may also use the functionality of EventsListening component (to invoke the listening to various events sources) and UserInterface component (to response the Traveller to the tracking request). The TrackingActivation component implements following Trip Tracker functions:

- OrchestrateActivation.
- GetPreferences.
- GetItinerary.
- ValidateItinerary.
- UpdatePatterns.

The **EventsListening** component aims at activation or deactivation of listening to the on-line information on traffic irregularities (events). To successfully achieve the goal, it also implements an interface to the Interoperability Framework. The EventsListening component implements following Trip Tracker functions:

- EventsListening.
- InvokeListening.

The **EventsProcessing** component implements the Perform Event Processing capability. To identify travel conflicts and their impact on individual travel plans, Trip Tracker uses the Complex Event

Processing techniques and technologies. The implementation comprises following Trip Tracker functions:

- OrchestrateCEP.
- Monitor.
- MatchPatterns.
- EvaluateConflicts.
- LogCEP.

The only purpose of the **UserInterface** component is to manage the sending of messages to users. However, it doesn't provide the direct communication to user's smart device. Through the exposed service, which may be used by any other Trip Tracker component, it represents the communication channel from Trip Tracker to Travel Companion. The UserInterface component implements following Trip Tracker functions:

- SendMessage.
- LogMessages.

The **AlternativesManager** component implements the Manage Alternatives capability. The aim is to enable the re-accommodation when necessary. To be able to satisfy the need of alternative offers, it implements an interface to Travel Shopper. Thanks to the interface to Interoperability Framework it also enables to change the provider for alternative offers. The AlternativeManager component implements following Trip Tracker functions:

- OrchestrateAlternatives.
- GetAlternatives.
- CheckAlternatives.
- ResolveAlternatives.
- SubmitAlternatives.
- LogAlternatives.

The **BAinterface** component serves the BusinessAnalytics with a set of functions intended for gathering the statistics on Trip Tracker's operation. It implements an interface to Business Analytics, providing following Trip Tracker functions:

- OrchestrateBA.
- ProvideBAdataCEP.
- ProvideBAdataMessages.
- ProvideBAdataAlternatives.

The purpose of the functions listed above is described in the chapter **Error! Reference source not found.**. They are used in functional scenarios, as described in the chapter **Error! Reference source not found.**.

4.2 FUNCTIONS

The Trip Tracker main functions are organized as follows.

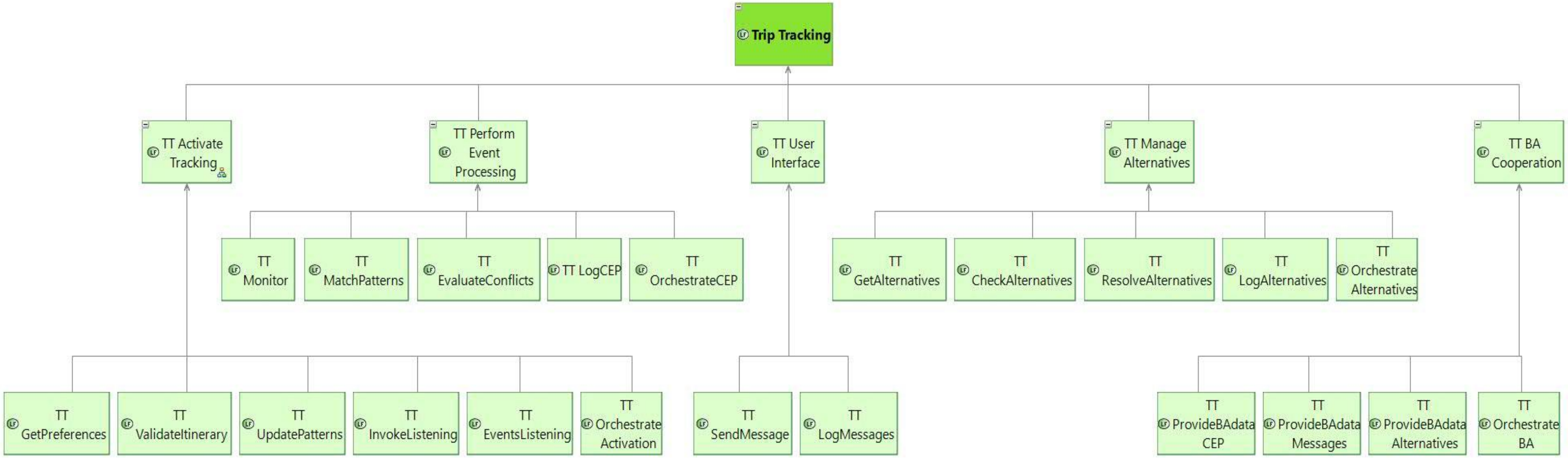

Figure 4-2: Trip Tracker functions

The operation of Trip Tracker is based on logical functions specified in the following chapters.

4.2.1 Activate Tracking

Operations executed under this activity are obvious from the related functional scenario, described in chapter 6.2.1. They are provided by the following functions:

- **OrchestrateActivation**
  The aim of this function is an activation of tracking of a trip. It starts with the traveller requesting the tracking of a specific journey through the Travel Companion Personal Application. This, in turns, sends the tracking request to the Trip Tracker. If the user needs to track an itinerary which is composed of multiple journeys, the Personal Application will split the request to several “journey tracking activation requests”, one for each part of the itinerary.
- **GetPreferences**
  This function retrieves from the Travel Companion all the preferences of a given user, which are relevant for trip tracking. Thus, it enables to customize the Trip Tracker’s behaviour according to Traveller’s preferences. A proposal of Trip Tracker’s preferences is presented in the following table. Certainly, this list is not complete and may be extended anytime. Its purpose is just to show that there are different groups of preferences, which may affect the behaviour of Trip Tracker in different ways.

Group	Preference	Eligible values	Default value
1. Trip Tracker behavior			
	1.1 Automatic tracking activation	Yes / No	No
	1.2 Offer alternatives	Yes / No	Yes
2. Messages selection			
	2.1 Message type	Information	Yes
		Warning	Yes
	2.2 Message content	*(examples below)*	All yes
		Cancellation messsage	
		Rerouting message	
		Platform change	
		No first class	
		No dining car	
		No refreshment	
		WC out of order	
		Air conditioning / heating out of order	
		Wi-Fi inaccessible	
		Newspapers and magazines not available	
		…	
3. Information on delays			
	3.1 Significant delay	value in minutes	5 min.
	3.2 Absolute connection time	value in minutes	10 min.
	3.3 Marginal connection time	value in minutes	10 min.
	3.4 Avoid message duplication	value in minutes	15 min.
	3.5 Minimum delay change	value in minutes	5 min.

Table 4-1: Trip Tracker’s Preferences

The first group of the Trip Tracker's preferences affects the Trip Tracker's behaviour in general. It contains two preferences: Automatic tracking activation and Offer alternatives. The first one enables the Traveller to request journey tracking automatically, whenever a new itinerary is stored in the Travel Companion Cloud Wallet (CW). The second preference enables the Traveller not to be bothered with alternatives offering.

The second group of the Trip Tracker preferences is called **Messages selection**. It enables the Traveller to disable receiving of certain type of messages. There is no choice for the third type of messages, as alerts must be sent always. Another possible approach is to choose exactly, which kind of information the Traveller wants to receive or not.

The last group of the Trip Tracker's preferences is dealing with **Information on delays**. If the required value is set up to 0, then the message is always displayed. The first preference of this group called Significant delay enables the Traveller to set up the minimum delay, when a message is sent. Other preferences from this group allow the Traveller to set up other options, when the delay message shall be delivered or not.

- **GetItinerary**
  In order to minimize the volume of data transferred from user's device, at the moment of tracking activation the Trip Tracker receives only the identification of an itinerary to be tracked. Therefore all the details of the itinerary in question have to be retrieved separately. That's the goal of this function.

- **ValidateItinerary**
  This function aims at validation of journeys which are to be tracked. The proposed concept assumes, that the validity of a journey will be verified through the Travel Shopper, which may provide either a special validation service returning the final decisions, or journey planner outputs, which could have been used by the Trip Tracker to make the decisions itself. This function is simulated in IT2Rail project and it is expected that the validation of itinerary will be solved within next projects of Shift2Rail. The simulation is carried out by random selection of successful or unsuccessful validations. This simulation affects following behaviour of Trip Tracker.

- **UpdatePatterns**
  This function is in charge of collecting information related to planned journeys. Patterns are generated for each Itinerary. These patterns are based on arrival, departure and connection times and reflect rules that match events. For example, such a rule can reflect that a connection is broken when an arrival exceeds a specific delay. Such patterns can be consequently used to create filters, which enable the Complex Event Processing engine to match incoming events properly.

- **EventsListening**
  The aim of this function is to initiate the listening to event sources relevant to the journey being tracked. This function creates a set of monitors responsible for listening to event sources related to the given journey. Because various event sources use different codelists for some data elements, it is necessary to deploy decoders, which are able to encode/decode such data. For the purpose of the IT2Rail project, it was decided to make use of the Navitia

platform[1] as a source of disruptive events. Therefore an appropriate "Navitia decoder" service is called from the Interoperability Framework.

- **InvokeListening**
  This function is in charge of activation and deactivation of listening to different event channels. Once the Trip Tracker is aware of the source of events, listening to this source may be started. When there is no need to listen to any of events source anymore, listening to that source may be stopped. This happens, when no Traveller in the particular network segment requires to be tracked at the moment.

4.2.2 Perform Event Processing

Operations executed under this activity are obvious from the related functional scenario, described in chapter 6.2.2. They are provided by the following functions:

- **Monitor**
  The aim of this function is to listen to event sources that may provide events which possibly affect Traveller's plans. Two ways of capturing events are commonly used. The event sources either broadcast information about events to subscribed users, who can subscribe only to a specific subset of event types of their interest, or all the information are available on the basis of request – response communication (in such case the Monitor must repeatedly request the news at specified intervals). The InvokeListening function instantiates the Monitors for each event source accordingly. The second task of the Monitor is to translate events from different event sources before they are submitted for further processing. The "Navitia decoder" service provided by the Interoperability Framework is used for that purpose. Only events matching the pre-set filter are released for further processing.

- **MatchPatterns**
  The role of this function is to recognize disruptions occurring in the infrastructure (e.g., unfeasible connections due to delays) from specific transport events, independent of the travellers moving through the transportation system. More precisely, the "Match Patterns" function combines together different - typically simple - events (possibly from different sources) to detect occurrences of more complex situations, for example disruptions. As an example, the function could take events concerning single vehicles moving through a transportation network - such as their time of arrival at stop places -, correlate them (by looking for certain "patterns" in the occurrences of events), and detect critical situations that involve multiple vehicles - e.g., the impossibility to make a connection. The events processed by this function to detect critical situations are usually received from monitors (i.e., the from the "Monitor" function of the Trip Tracker component), but the function can also correlate events that itself detects from other ones. The detection performed by this component is independent of actual travellers moving through the system, in the sense that it does not take into consideration user preferences, or even whether there are actually users, but it only depends on the infrastructure and on the vehicles moving through it.

---

[1] The Navitia platform (https://www.navitia.io) provides API for access to public transport data of approximately 1.600 networks in 25 countries who support an "open data" concept.

The events generated by this function are typically consumed by other functions of the system (in particular by the "Evaluate Conflicts" function). Hence, the "Match Patterns" function should publish a feed of the events it generates, to allow other functions to retrieve them.

A particular case of the mechanism described above is the one where the information received by the "Match Patterns" function from the "Monitor" function is already rich enough to be passed on to other functions. In this case, the "Match Pattern" function acts as an intermediary and a collector of information to be relayed to other functions of the system.

- **EvaluateConflicts**
  This function takes as input the events generated by the "Match Patterns" function and, for each user, knowing the journey that the user is making and depending on the user preferences, determines which disruptions are of interest for that user. To do this, the "Evaluate Conflicts" function receives the feed of events from the "Match Patterns" function and, for each event, evaluates its impact on the user journey. As an illustrative example, the "Match Patterns" function could detect a shrinking actual connection time between two vehicles along a journey (because of delays in the transport network, though the reason for the disruption is not necessarily of interest for the "Evaluate Conflicts" function), but not all users on the same impacted journey are affected in the same way: for some users a short connection time could still be enough to make the connection, whereas for others (for example for PRM, because they need more time to move around a stop place) it is not.

  If a disruption impacts the user journey in a way that the "Evaluate Conflicts" function deems relevant, the latter notifies the function that interfaces the Trip Tracker with the user that an alert should be sent to the Traveller. The alert message will be sent to the Trip Tracker User Interface.

- **LogCEP**
  For the logging of different data, it is necessary to find a way to centralize all the logs so they can be further analysed in an easy way. The performance of the Complex Event Processing engine is logged by this function in order to collect data for Business Analytics purposes. The following data are logged:

    - Arrival Delay Event based on User-defined Rules.
    - Departure Delay Event based on User-defined Rules.
    - User Activation of User-defined Arrival Delay Event Rules.
    - User Activation of User-defined Departure Delay Event Rules.
    - User Deactivation of the User-defined Delay Event Rules.

- **OrchestrateCEP (not implemented)**
  The orchestration of the Perform Event Processing component was scheduled because it was supposed that this component will appear in multiple instances. However, it was decided to simplify the IT2Rail project, therefore this function is not described.

4.2.3 User Interface

Operations executed under this activity are obvious from the related functional scenario, described in chapter 6.2.3. They are provided by the following functions:

- **SendMessage**
  This is the universal messaging feature intended for sending any message to the Traveller (through the interface to Travel Companion). Since one event may have different impacts to different customers, 3 different impact levels and appropriate message types are expected to be distinguished (see Table 4-2: Impact levels & Message types).

Impact	Description	Message
No	no direct impact on Traveller's journey	Information
Low	there is an impact on Traveller's journey, but neither re-routing nor provision of alternatives are required	Warning
High	the highest impact on Traveller's journey, invokes the recalculation of journey and provision of alternatives	Alert

Table 4-2: Impact levels & Message types

- **LogMessages**
  Every message sent by the Trip Tracker to the Traveller is logged by this function in order to collect data for Business Analytics purposes.

4.2.4 Manage Alternatives

Operations executed under this activity are obvious from the related functional scenario, described in chapter 6.2.4. They are provided by the following functions:

- **GetAlternatives**
  When a disruption is detected, the Traveller should be informed about possible solutions how to reach the destination either in time or at least with the lowest possible deviation. The list of possible alternatives will be calculated by calling the Travel Shopper.

  The GetAlternatives function needs to provide the Travel Shopper with the itinerary concerned. Only with this original itinerary the Travel Shopper is able to calculate the right routes taking into consideration preferences, possible stop overs, planned arrival time etc. First, GetAlternatives gains this information out of the cloud storage of the concerned Traveller. In order to do it, it needs the BookedOfferID. This is passed over by invoking it through the call of the Travel Companion. Then, the Travel Shopper is called and as a result the list of several alternative routes is passed over to the Alternative Manager.

- **CheckAlternatives**
  The goal of this function is to check the offer of alternatives retrieved from the Travel Shopper. Sometimes it may happen that the offer of alternatives, provided by a Travel Shopper, doesn't take into account the event, which originally caused the request for alternatives provision. Such offer is considered to be insufficient and the ResolveAlternatives function is called repeatedly, until the correct offer of alternatives is returned, or the timeout expires.

- **ResolveAlternatives**
  There are several concepts how to resolve the incorrect offer of alternatives. The entire offer of alternatives may be thrown away and a new attempt to get the offer from another provider (recommended by the Interoperability Framework) may be made (at least once). Or a new attempt to get the offer from the same provider may be made, but with the modified mobility request (for example a particular edge may be excluded). Certainly, the offer of alternatives may also be shortlisted by excluding only those invalid alternatives from the list and then such a fixed offer of alternatives may be proposed to the Traveller without looking for another solution.

- **SubmitAlternatives**
  In case that the previous functions have been able to determine at least one alternative route this thereafter hast to be forwarded to the user. This is done as the response of the previous request of alternatives from the Travel Companion. If no alternative route could be found an empty list is send to the Travel Companion.

- **LogAlternatives**
  The performance of the AlternativesManager component is logged by this function in order to collect data for Business Analytics purposes.

- **OrchestrateAlternatives (not implemented)**
  This function will not be implemented in IT2Rail. In the future it might be necessary to pass the calculation of alternative itineraries to different alternative managers. These local managers have to be orchestrated and organised. This would be necessary when neither the original travel experts nor alternative providers are able to find proper alternative routes for a traveller. In such case, the offer of alternatives is completed by this function from the particular results provided by local alternative managers.

4.2.5 BA Cooperation

Operations executed under this activity are obvious from the related functional scenario, described in chapter 6.2.5. They are provided by the following functions:

- **ProvideBAdataCEP**
  The aim of this function is to provide Business Analytics with information on events' evaluation from the Complex Event Processing engine.
  The CEP recognises two main types of events:
    - Arrival Delay Event based on User-defined Rules.
    - Departure Delay Event based on User-defined Rules.

CEP also manages three user-defined activities:

  - User Activation of User-defined Arrival Delay Event Rules.
  - User Activation of User-defined Departure Delay Event Rules.
  - User Deactivation of the User-defined Delay Event Rules.

The ProvideBAdataCEP function also provides information regarding these last three activities.

- **ProvideBAdataMessages**
  The aim of this function is to provide Business Analytics with information on all messages that were sent to Travel Companion to be displayed to the Traveller. This function retrieves the information logged by the LogMessages function, relevant to extract KPIs that can give a value insight of the messages send to the user in relation to the impact of the events on user´s journeys. More details of the results and KPI that can be obtained through this function are included in Business Analytics deliverables.

- **ProvideBAdataAlternatives**
  The aim of this function is to provide Business Analytics with information on alternatives offered upon Traveller’s request. It enables Business Analytics to compute statistics related to alternatives management. This function retrieves the information logged by the LogAlternatives function, relevant to extract KPIs that can give a value insight of the alternatives management processes that take place in case of disruption or event affecting user´s journeys. More details of the results and KPI that can be obtained through this function are included in Business Analytics deliverables.

- **OrchestrateBA (not implemented)**
  The orchestration of the BA Cooperation component was scheduled because it was supposed that this component will appear in multiple instances. However, it was decided to simplify the IT2Rail project, therefore this function is not described.

4.3 TRIP TRACKER FUNCTIONAL ARCHITECTURE

The Trip Tracker functional architecture is described in the following diagram:

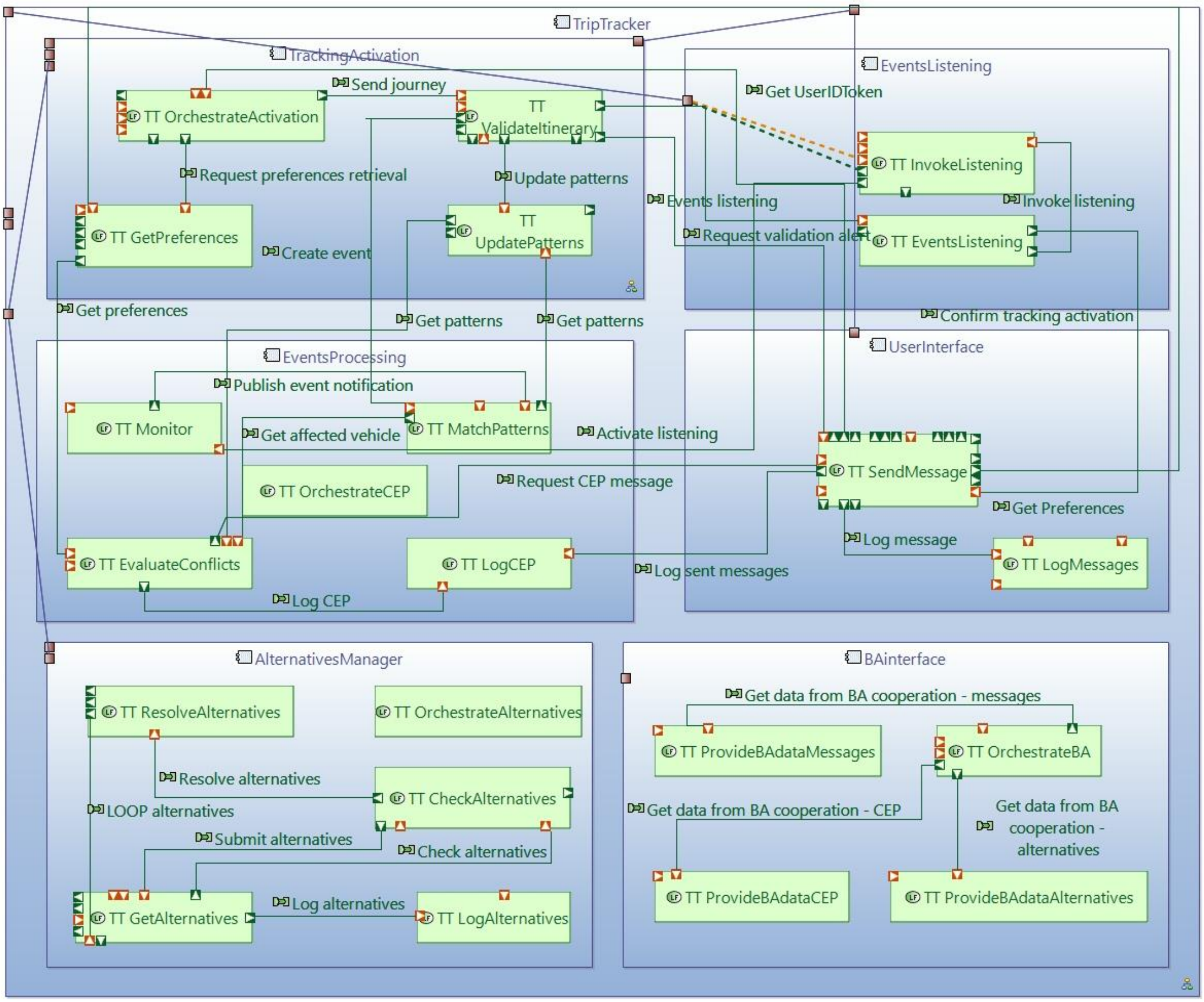

Figure 4-3: Trip Tracker functional architecture

4.4 FUNCTIONS OF INVOLVED ACTORS

As described in the chapter 3.4.1, two external actors are involved in the Trip Tracker capabilities:

- Traveller.
- Travel Event Provider.

In this chapter, the way of their involvement is described. The following functions are linked to these actors.

- **Trigger tracking**: This function is used by a Traveller to launch the activities described in the Activate Tracking functional scenario (see also 6.2.1). The tracking request can be either made explicitly by the Traveller or triggered automatically according to the settings of Traveller’s preferences.

- **Broadcast events**: This function belongs to the Travel Event Provider. It is used to publish the information on disruptions or other traffic irregularities in the particular network segment. When such information is captured by the Trip Tracker, it initiates activities described in the Perform Event Processing functional scenario (see also 6.2.2).

- **Request alternatives**: This function is used by the Traveller to confirm the willingness and readiness for receiving the alternative offer suggestions. It triggers the activities described in the Manage Alternatives functional scenario (see also 6.2.4).

The following figure illustrates the connection of actors’ functions in the Trip Tracker capabilities.

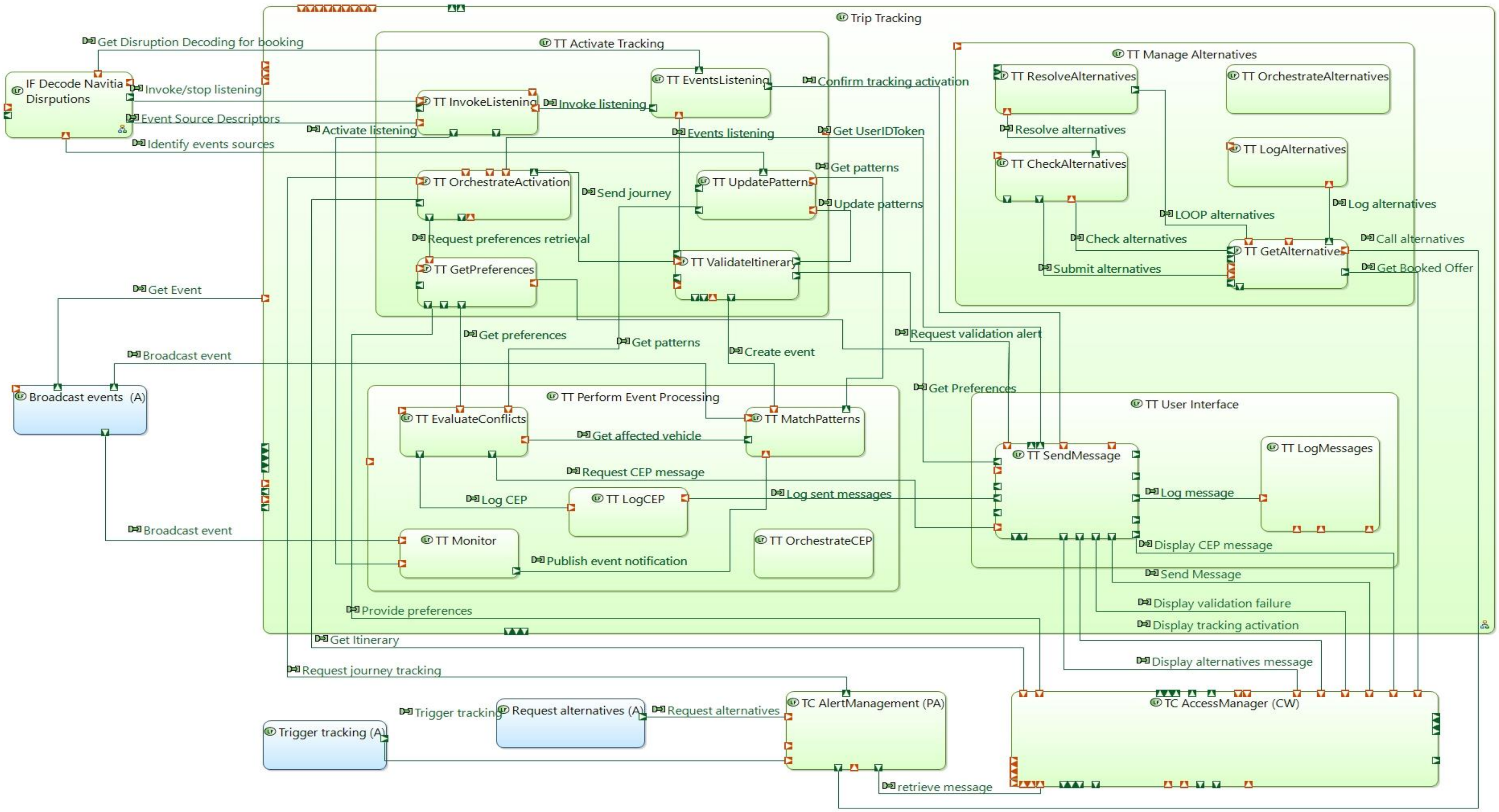

Figure 4-4: Actors' functions

5. DESIGN DECISIONS AND CONSTRAINTS

This chapter contains all the design decisions and constraints that have arisen from Trip Tracker and that may affect other functional areas of IT2Rail.

Design decisions related to the **Activate Tracking**:

- The user (userID) is identified by a unique identifier (UserIDToken).
- The journey is identified in relation to Travel Companion (PushMessageCW) as journey.
- The functionality of Trip Tracker is influenced by user's preferences and Trip Tracker's preferences which are retrieved from Travel Companion.
- To identify relevant event sources it is necessary to query the Interoperability Framework.
- In some cases it will be necessary to deploy a decoder which is able to translate some data elements (StopPlaces IDs, journey IDs, …) used by the event sources (the Navitia decoder service provided by the Interoperability Framework is used for the project purposes).
- When the activation of tracking is completed, a confirmation is sent to the user.

Design decisions related to the **Perform Event Processing**:

- The trackingEvent data structure retrieved from Event Source is modified in the Monitor.
- The result of the event's impact assessment is sent to user's device through the User Interface component.

Design decisions related to the **User Interface**:

- The availability of SendMessage function is ensured by the exchange item RequestMessages_EI.
- Messages are sent to the user by the exchange item PushMessageCW through Travel Companion (TC AccessManagement).
- The message is identified by a class SendMessage which consists of:
    - messageTypeS.
    - messageTitle.
    - messageShortText.
    - messageFullText.
    - messageObject.
    - messageAsk4Alt.

Design decisions related to the **Manage alternatives**:

- To gain maximum flexibility the Alternative Manager receives only the necessary IDs and the following additional information:
    - UserID token.
    - GeoCoordinates.
    - TravelExpert.
    - BookedOfferID.

  This information is enough to enable the Alternative Manager to calculate alternative solutions for the Traveller.

- A central framework (the core “Alternative Manager”) handles all requests and responses. Changes can easily be implemented if desired.
- Only where necessary responses are verified and acknowledged. E.g. no response is expected from the BA component because it provides data stored by other components.

Design decisions related to the **BA Cooperation**:

Activation of Event Processing rules managed by the BACooperation module will be achieved through the implementation of a Java application deployed on a Tomcat server exposing REST web services.

User updating the above-mentioned Event Processing rules makes use of two rule templates, one for Arrival Delay Events and the other for Departure Delay Events:

- select * from ArrivalDelayEvent where idVehicle=’’ and locationCode = '' and nominalArrivalTime=’’.
- select * from DepartureDelayEvent where idVehicle=’’ and locationCode = '' and nominalDepartureTime=’’.

where idVehicle, locationCode and nominalDepartureTime are defined by the user through the Tracking Activation user interface.

6. CAPABILITIES: SEQUENCE DIAGRAMS

This chapter contains UML sequence diagrams for each capability in which the Trip Tracker module is involved. For each of these capabilities there are two functional scenarios defined: at system level and at logical level.

6.1 SYSTEM LEVEL

At this level, Trip Tracker activities corresponding to respective use case are described mostly in terms of interactions between IT2Rail modules. More detailed description follows at logical level.

6.1.1 Activate Tracking

This scenario describes tracking activation at system level.

This scenario aims at reflecting Traveller's decision on journey tracking. When triggered by the Traveller (that means the Traveller explicitly agrees to be informed on events which affect the planned journey), Travel Companion sends a request to the Trip Tracker to activate tracking of selected journey. First of all, Trip Tracker retrieves the Traveller's preferences, in order to customize its behaviour. Then, having all the necessary data collected, Trip Tracker responds to the tracking request. Such response must be considered only as an acknowledgment of receipt, not the confirmation that the tracking of given journey has been already started.

Before the tracking of given journey may be activated, its validity must be verified. If an itinerary becomes invalid, the Trip Tracker not only sends an alert to the relevant Traveller, but also considers such information to be a new event and thus will be able to evaluate its impact not only to the related Traveller, but to other travellers as well.

When the validation of journey which is to be tracked ends successfully, relevant filters are updated to enable the Complex Event Processing engine to match incoming events properly. Possible sources of events are identified by the appropriate resolver of the Interoperability Framework. Once the Trip Tracker is aware of the source of events, listening to this source may be started. In the end, the confirmation of tracking activation is sent to the related Traveller (through the Travel Companion).

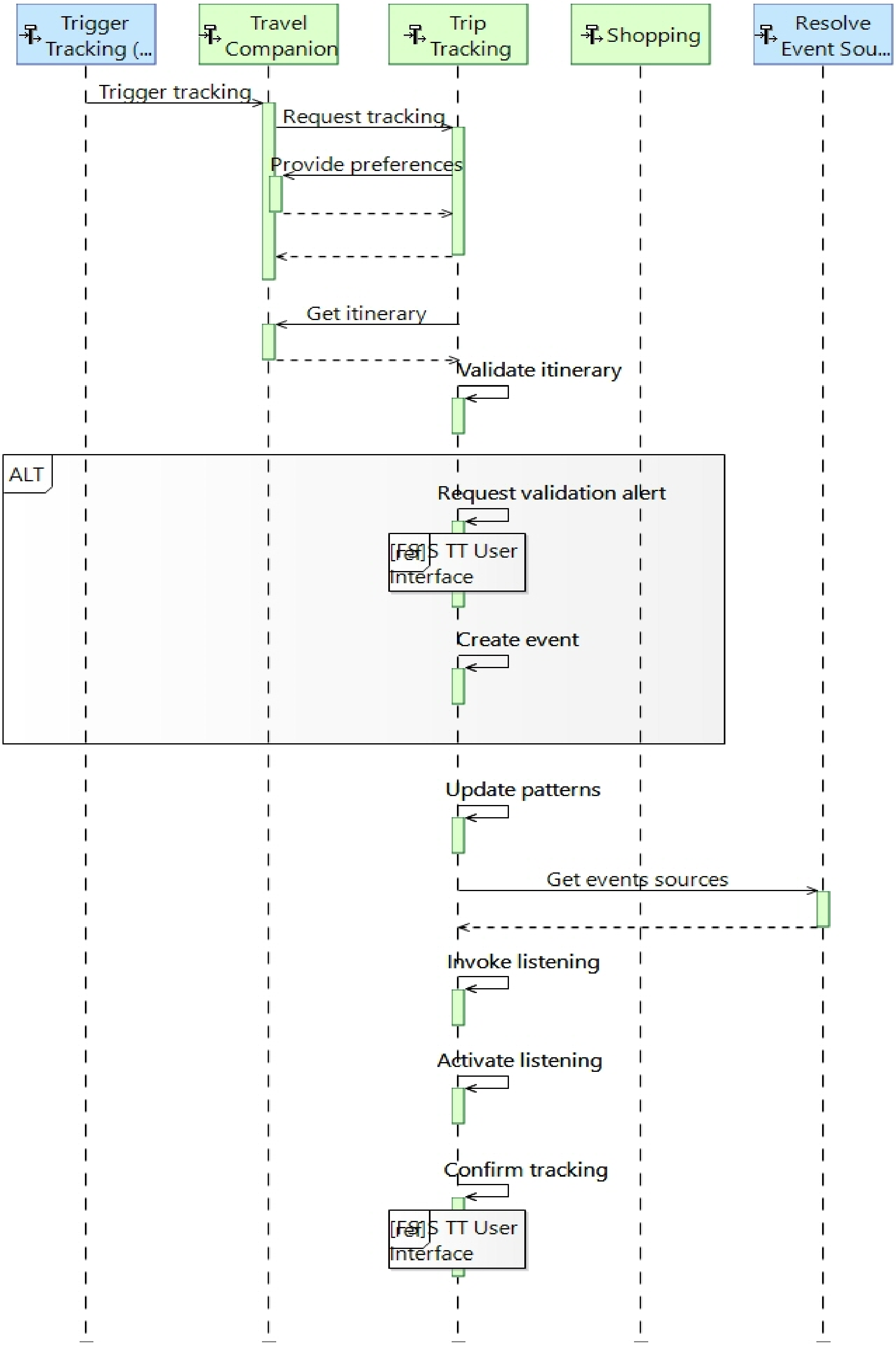

Figure 6-1: [FS]S Activate Tracking

6.1.2 Perform Event Processing

This scenario describes at system level the operation of the Complex Event Processing engine.

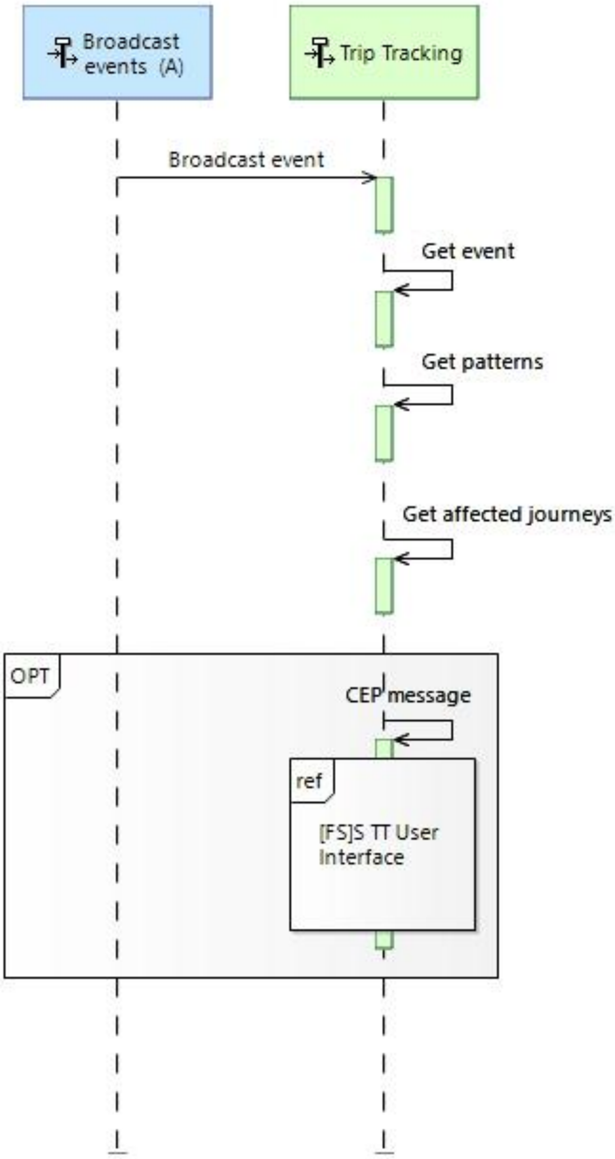

Figure 6-2: [FS]S Perform Event Processing

When an event broadcasted by the Travel Event Provider is captured by the Trip Tracker, it passes to a 2-step process of evaluation. First, it is necessary to identify, whether it may affect anyone's travel plans. Only events matching the pre-set filters are released for further processing. In the second step, an analysis of the potential impact of identified disruptions is performed to assess the severity of identified conflicts and to decide, if alternative solutions have to be requested, because the original travel plan is not applicable anymore.

In the end, a message is sent to each affected Traveller (through the Travel Companion), informing them on an event occurrence, on its impact level and what are the possible next steps.

6.1.3 User Interface

This scenario describes at system level the process of sending messages to Travel Companion.

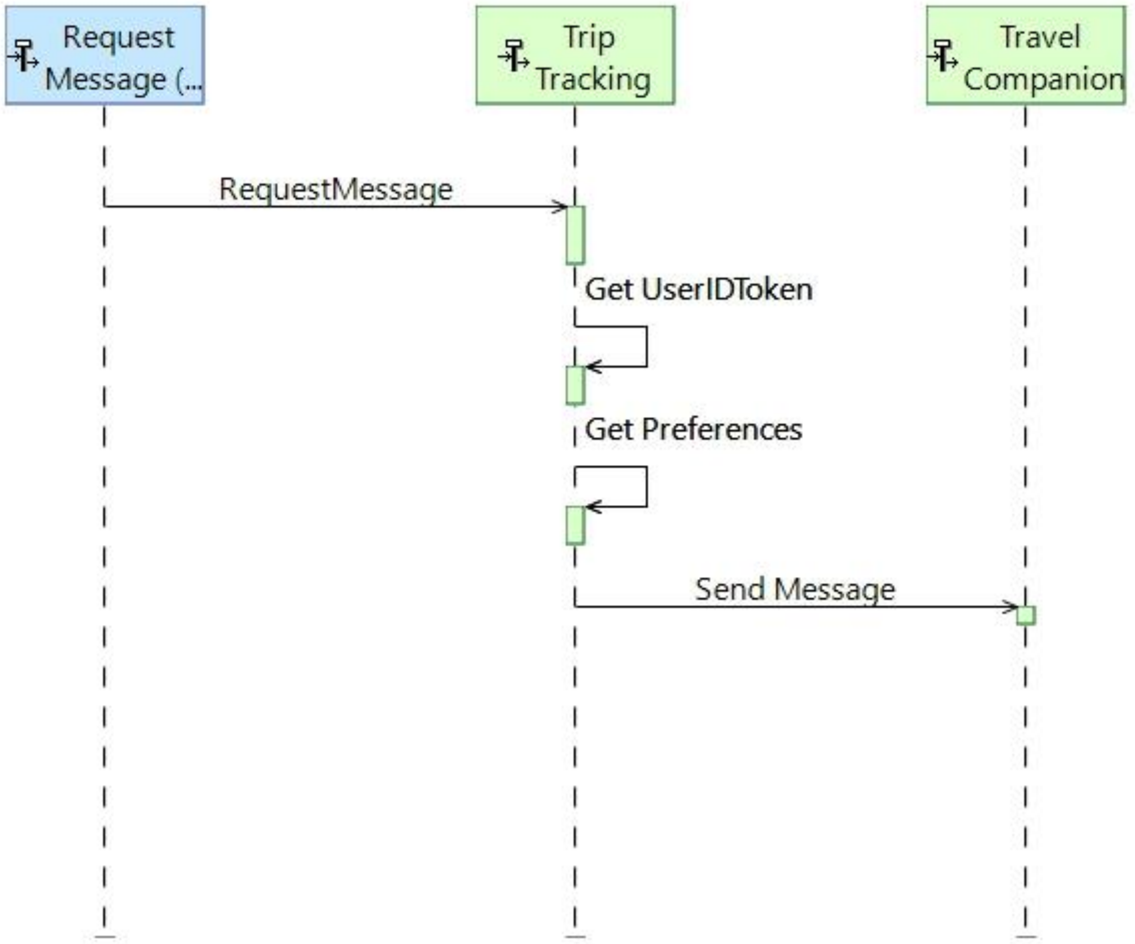

Figure 6-3: [FS]S User Interface

Sending of a message is triggered by a request, which arises in various contexts. It reflects the Traveller’s preferences retrieved during the tracking activation. At the end of processing the request, the message is sent to Travel Companion, to be displayed on Traveller’s device finally.

6.1.4 Manage Alternatives

This scenario describes the alternatives management at system level.

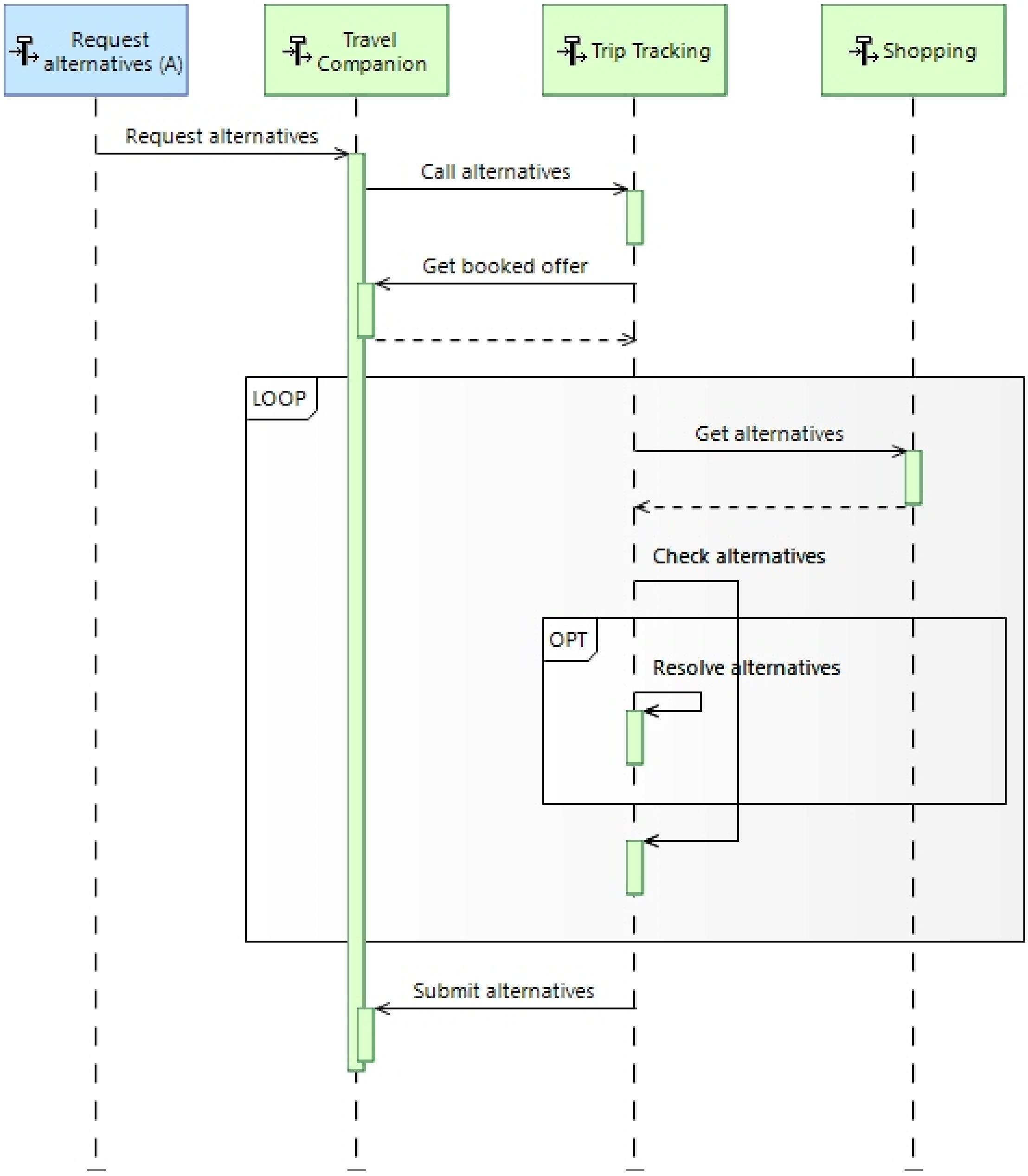

Figure 6-4: [FS]S Manage Alternatives

When a disruption with the highest impact on Traveller's journey has been identified, the original travel plan is not applicable anymore. That means that the recalculation of the original journey is necessary, if the affected Traveller wants to continue traveling in terms of reaching the original destination as soon as possible. In that case (upon Traveller's decision) the Travel Companion sends a request for alternatives to the Trip Tracker.

After being activated the Trip Tracker asks the Travel Shopper for alternatives. It is assumed, that the same Offer Provider who offered the original itinerary, will serve the request for alternatives. When the offer of alternatives is retrieved from the Travel Shopper, it's necessary to check its content. The reason is that sometimes it may happen, that the retrieved offer of alternatives doesn't take into account the event, which originally caused the request for alternatives provision. In such cases the offer must be considered inconvenient and the offer of alternatives must be corrected. A new attempt can be made to get the offer from another provider (recommended by the appropriate resolver of the Interoperability Framework), or an updated mobility request can be send to the same provider, or the list of alternatives can be shortlisted by excluding the invalid offers. Of course, such loop may be repeated several times if necessary, but the execution time must be considered carefully. Therefore, it is assumed that if the first attempt to get the offer of alternatives fails, the repetition will typically take place only once.

In the end, either the correct offer of alternatives is submitted or a report of failure with some suggestions is sent to the Traveller (through the Travel Companion).

6.1.5 BA Cooperation

At system level, this scenario simply says that the Trip Tracker will support the provision of data to Business Analytics.

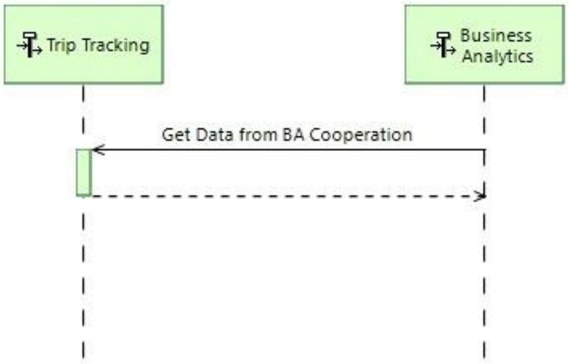

Figure 6-5: [FS]S BA Cooperation

The BA module obtains Trip Tracker's event data by analysing the content of appropriate logs.

6.2 LOGICAL LEVEL

At this level, Trip Tracker activities corresponding to respective use case are described in terms of interactions between logical functions. This way it brings a clear view, which logical parts of related IT2Rail modules are involved in carrying out the given activity.

6.2.1 Activate Tracking

This scenario describes tracking activation at logical level.

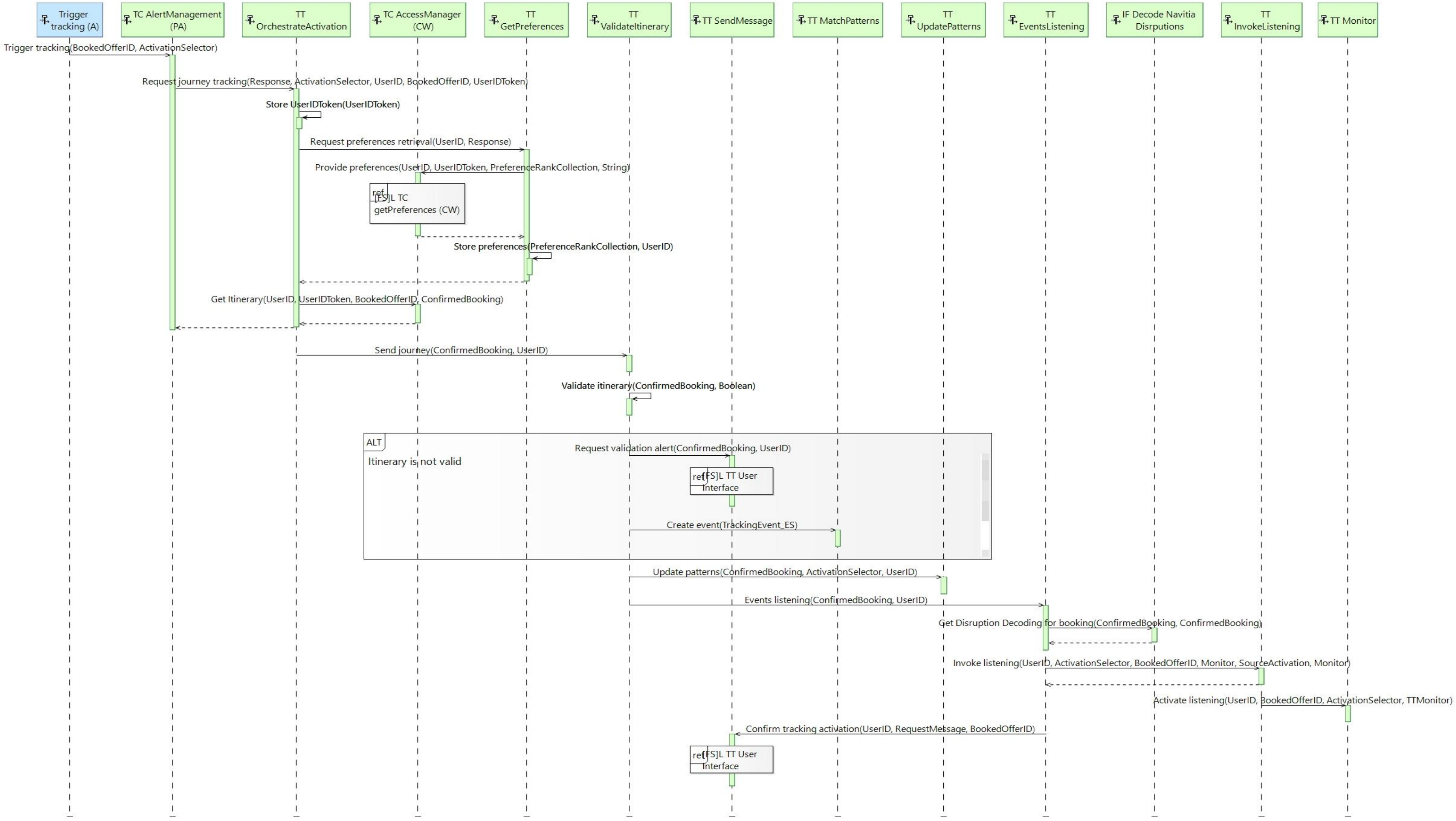
Trigger tracking (A)
TC AlertManagement (PA)
TT OrchestrateActivation
TC AccessManager (CW)
TT GetPreferences
TT ValidateItinerary
TT SendMessage
TT MatchPatterns
TT UpdatePatterns
TT EventsListening
IF Decode Navitia Disrputions
TT InvokeListening
TT Monitor
Trigger tracking(BookedOfferID, ActivationSelector)
Request journey tracking(Response, ActivationSelector, UserID, BookedOfferID, UserIDToken)
Store UserIDToken(UserIDToken)
Request preferences retrieval(UserID, Response)
Provide preferences(UserID, UserIDToken, PreferenceRankCollection, String)
ref
[FS]L TC getPreferences (CW)
Store preferences(PreferenceRankCollection, UserID)
Get Itinerary(UserID, UserIDToken, BookedOfferID, ConfirmedBooking)
Send journey(ConfirmedBooking, UserID)
Validate itinerary(ConfirmedBooking, Boolean)
ALT
Itinerary is not valid
Request validation alert(ConfirmedBooking, UserID)
ref
[FS]L TT User Interface
Create event(TrackingEvent_ES)
Update patterns(ConfirmedBooking, ActivationSelector, UserID)
Events listening(ConfirmedBooking, UserID)
Get Disruption Decoding for booking(ConfirmedBooking, ConfirmedBooking)
Invoke listening(UserID, ActivationSelector, BookedOfferID, Monitor, SourceActivation, Monitor)
Activate listening(UserID, BookedOfferID, ActivationSelector, TTMonitor)
Confirm tracking activation(UserID, RequestMessage, BookedOfferID)
ref
[FS]L TT User Interface

Figure 6-6: [FS]L Activate Tracking

This scenario aims at reflecting Traveller's decision on journey tracking. When triggered by the respective actor's function (Trigger tracking), the Travel Companion Personal Application (hereinafter referred as a Personal Application only) sends a request for journey tracking to the Trip Tracker. When tracking of more than one journey is requested by the Traveller at one time (an itinerary often consists of 2 journeys there and back), the Personal Application splits such request into multiple journey tracking requests, which are consequently processed by the Trip Tracker independently of one another. Certainly, every journey may comprise more than one Travel Episodes.

First of all, the request for journey tracking is sent to the OrchestrateActivation function with Traveller's unique userID together with userIDToken, itinerary's identification (offerId) and an activation status (activation/deactivation of journey tracking). Traveller's unique userID is passed to the GetPreferences function. This enables Trip Tracker to retrieve Traveller's preferences and thus to customize its own behaviour. To enable the trip tracking, the GetItinerary function asks the Cloud Wallet (part of the Travel Companion) for a whole structure of itinerary through a standalone request with parameters offerId and userID. Then, having all the necessary data collected, Trip Tracker responds to the tracking request. Such response must be considered only as an acknowledgment of receipt of the request, not as a confirmation that the tracking of given journey has been already started.

As a next step, the validity of given journey which is to be tracked must be verified by the ValidateItinerary function, before the tracking of the journey may be activated. The proposed concept assumes, that the validity of a journey will be verified through the Travel Expert Journey Planner function of the Travel Shopper, which may provide either a special validation service returning the final decisions, or common journey planner outputs, which could have been used by the Trip Tracker to make the decisions itself. This functionality is out of IT2Rail scope, but it is easily simulated by returning a random indication, whether the journey is valid or not.

If a journey proves to be invalid, the SendMessage function from the Trip Tracker's UserInterface component is called in order to immediately send an alert to the relevant Traveller, informing him/her, that (and why) the requested tracking activation has failed. Moreover, Trip Tracker not only sends an alert to the relevant Traveller, but also considers such failure to be a new event. Such event is transmitted to the MatchPatterns function. This enables the Complex Event Processing engine to evaluate the event's impact not only to the related Traveller, but to other travellers as well.

When the validation of journey which is to be tracked ends successfully, relevant filters are updated within the UpdatePatterns function to enable the Complex Event Processing engine to match incoming events properly. For this reason, Traveller's unique userID together with userIDToken, the whole structure of itinerary and activation type are sent to this function.

Then the EvensListening function may be called. To be able to activate the listening to proper Travel Event Providers, available sources of events must be identified and their descriptors must be provided to the Trip Tracker first. This is done by the Event Source Resolver of the Interoperability Framework. Once the Trip Tracker is aware of the sources of events, listening to those sources may be started. The listening itself is performed by so called Monitors. However, the Analysis of events sources revealed that various event sources use different event structures. Therefore each Monitor, listening to the specific event source, has to translate the incoming events before transmitting them for further processing. In the end, the confirmation of tracking activation is sent to the relevant Traveller by the SendMessage function (through the Cloud Wallet part of the Travel Companion).

6.2.2 Perform Event Processing

This scenario describes at logical level the operation of the Complex Event Processing engine.

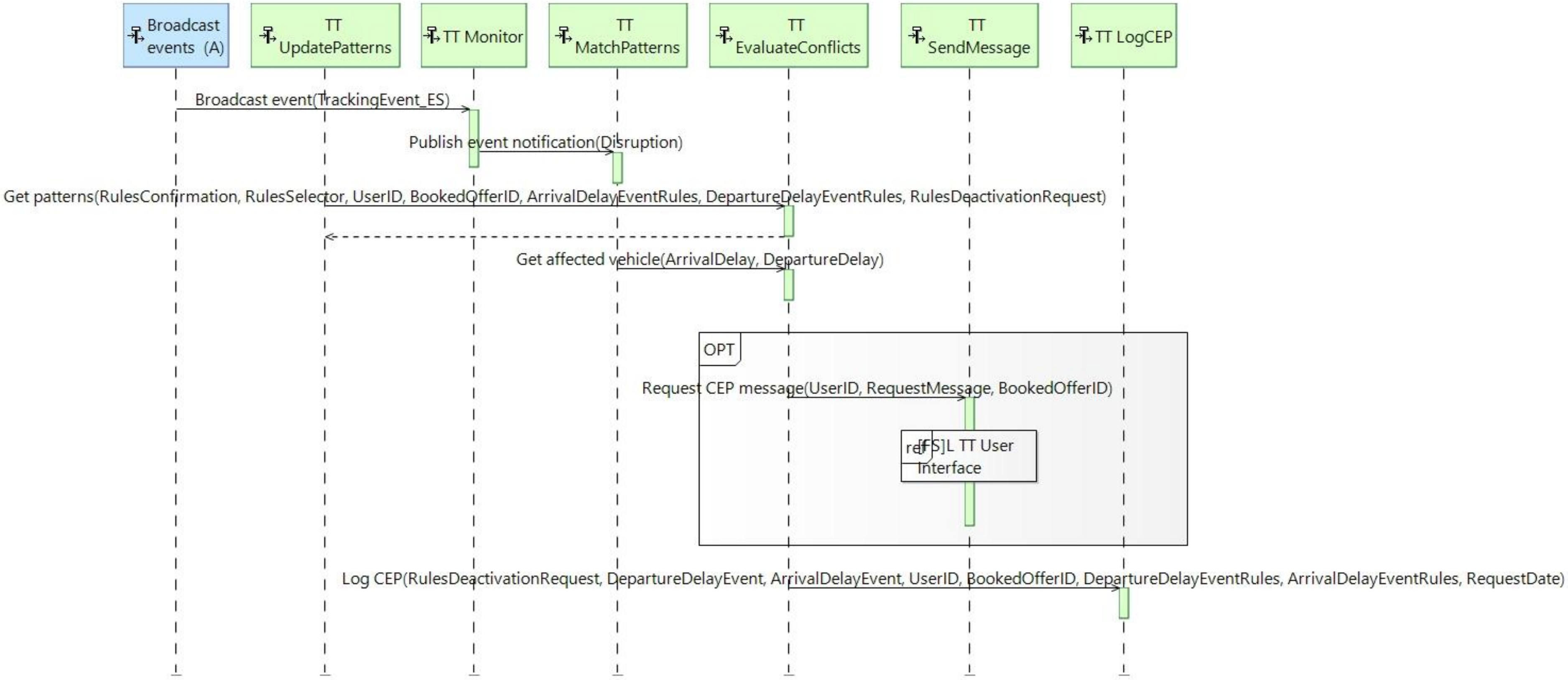
Broadcast events (A)
TT UpdatePatterns
TT Monitor
TT MatchPatterns
TT EvaluateConflicts
TT SendMessage
TT LogCEP
Broadcast event(TrackingEvent_ES)
Publish event notification(Disruption)
Get patterns(RulesConfirmation, RulesSelector, UserID, BookedOfferID, ArrivalDelayEventRules, DepartureDelayEventRules, RulesDeactivationRequest)
Get affected vehicle(ArrivalDelay, DepartureDelay)
OPT
Request CEP message(UserID, RequestMessage, BookedOfferID)
ref
[FS]L TT User Interface
Log CEP(RulesDeactivationRequest, DepartureDelayEvent, ArrivalDelayEvent, UserID, BookedOfferID, DepartureDelayEventRules, ArrivalDelayEventRules, RequestDate)

Figure 6-7: [FS]L Perform Event Processing

When an event broadcasted by the respective actor’s function (Broadcast events) is captured by the Trip Tracker, it passes to a 2-step process of evaluation called Complex Event Processing.

These events can be distinguished as short-term or mid-term. A list of these events is drafted below:

- Schedule information process data service: Real-time data from the operational procedure for short-term enquiries:
    - pure real-time data (e.g. schedule status).
    - cancellation or addition of trips.
    - change to pattern, stop out of use.
    - change to mode of transport / vehicle equipment.
    - bay / platform change.
    - schedule deviations (delayed/early) upon violation of a threshold value; stop specific, traffic jams.
    - vehicle capacities / passenger loads.
    - changes to attributes (passage, no alighting and no boarding, extra trip, provision for cycles, unscheduled stops, info texts).
- Schedule information reference data service: Up-to-date planned schedules for mid-term enquiries:
    - cancellation or addition of trips.
    - change to pattern, stop out of use.
    - change to mode of transport / vehicle equipment.
    - bay / platform change.
    - changes to attributes (passage, no alighting and no boarding, extra trip, provision for cycles, unscheduled stops, info texts).

The next figure describes the orchestration of Complex Event Processing.

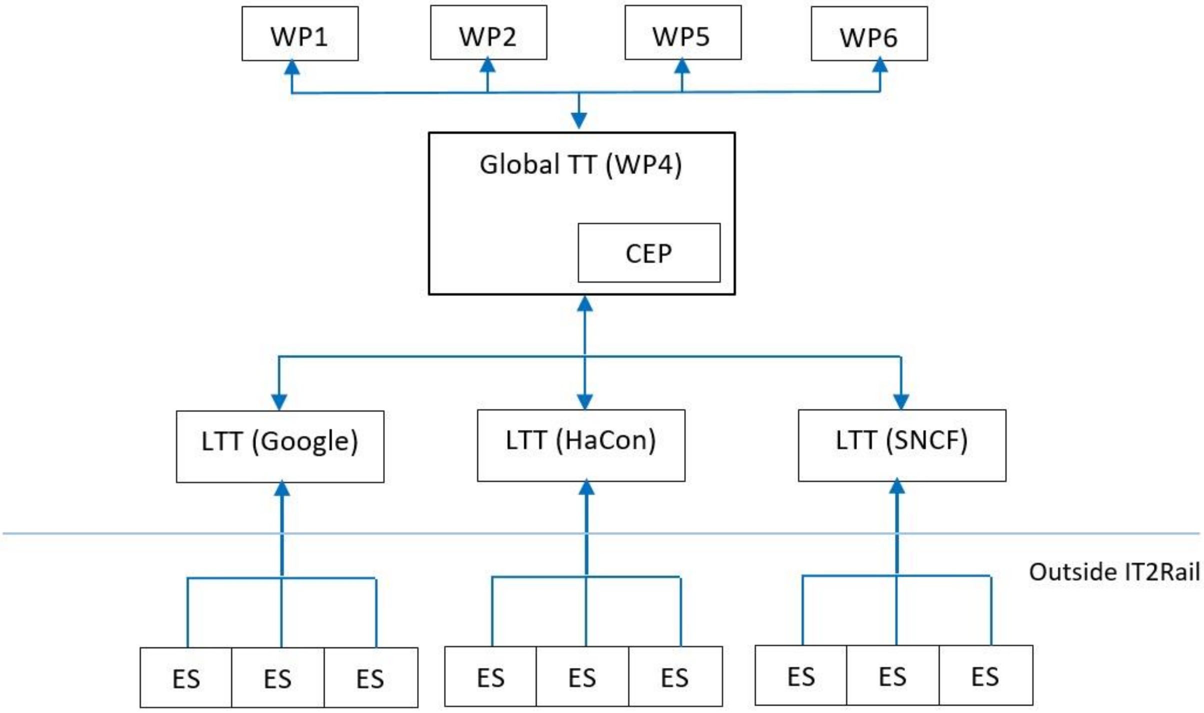

Figure 6-8: Complex Event Processing

CEP has been conceived by means of two levels, thus local Trip Tracker and global Trip Tracker. In this figure is illustrated only one global Trip Tracker but is assumed the existence of a large amount of such global Trip Tracker. In the position of the decision maker will work the resolver who will decide which global Trip Tracker will handle the particular itinerary.

First, as it is writing in previous chapter, the Monitor converters the event from events source into suitable structure because it is necessary to identify, whether the event may affect anyone's travel plans. This is the aim of the MatchPatterns function and in the Figure 6-7 is this function represented such as local Trip Tracker, which plays the role of the more sophisticated event source. Only events matching the preset filters are released for further processing. The MatchPatterns function works with the general rules such as e. g. the significant delay.

In the second step, an analysis of the potential impact of identified disruptions is performed by the EvaluateConflicts function in order to assess the severity of identified conflicts and to decide, if alternative solutions have to be requested, because the original travel plan is not applicable anymore. The EvaluateConflicts function is influenced by the MatchPatterns function with the general rules and the UpdatePatterns with the specific rules. These specific rules are activated and deactivated with set parameters.

In the end, a message is sent to each affected Traveller by the SendMessage function (Display CEP message) called from the Trip Tracker's UserInterface component, informing them (through the Travel Companion's messaging subsystem of the Cloud Wallet) on an event occurrence, on its impact level and what are the possible next steps. Therefor the message also contains information on Trip Tracker's decision, whether the original travel plan is still applicable, or if alternative solutions have to be requested. The message also contains unique Trip Tracker ID for this reason: the conflict (the event) triggers the need of the alternatives at the request of the Traveller whom is displayed the notification about the event.

6.2.3 User Interface

This scenario describes at logical level the process of sending messages to Travel Companion.

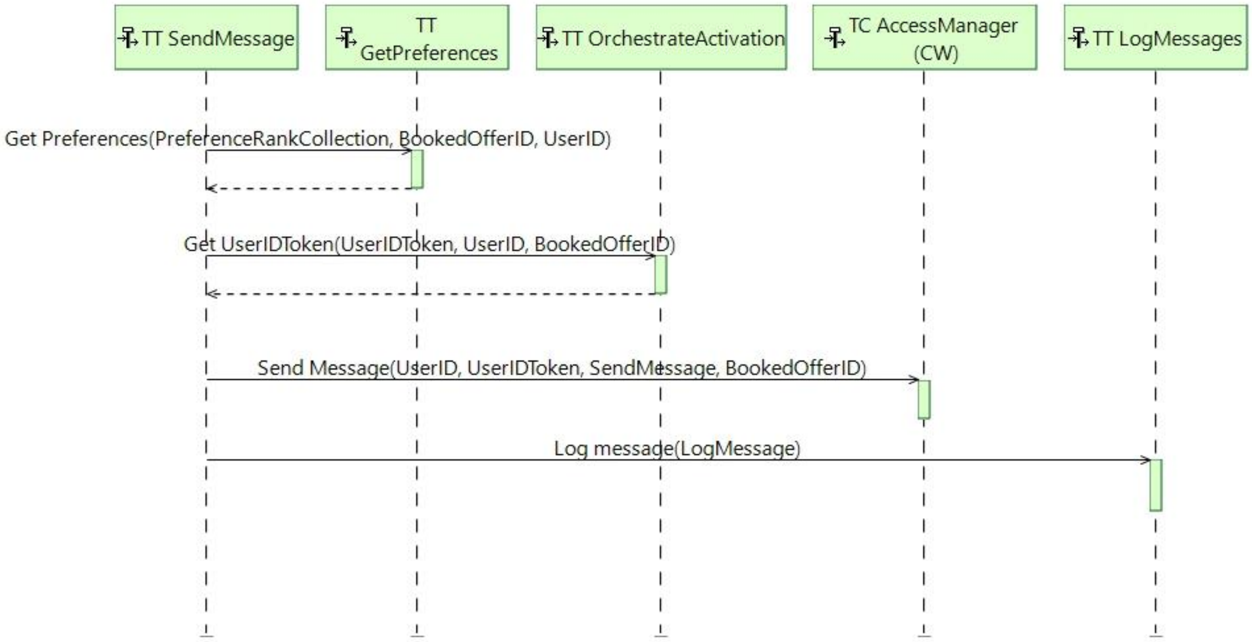
TT SendMessage
TT GetPreferences
TT OrchestrateActivation
TC AccessManager (CW)
TT LogMessages
Get Preferences(PreferenceRankCollection, BookedOfferID, UserID)
Get UserIDToken(UserIDToken, UserID, BookedOfferID)
Send Message(UserID, UserIDToken, SendMessage, BookedOfferID)
Log message(LogMessage)

Figure 6-9: [FS]L User Interface

Sending of a message is always triggered by a request, which arises in various contexts. Whenever a component of Trip Tracker intends to send a message to the Traveller, it requests the User Interface component to perform the process. The User Interface component retrieves the Traveller's preferences, stored during the tracking activation. These preferences affect the decision, whether the requested message will be sent or not at the end. When the message is sent to Travel Companion to be displayed on Traveller's device, it is also logged for future use by Business Analytics (through the BAinterface component).

6.2.4 Manage Alternatives

This scenario describes the alternatives management at logical level.

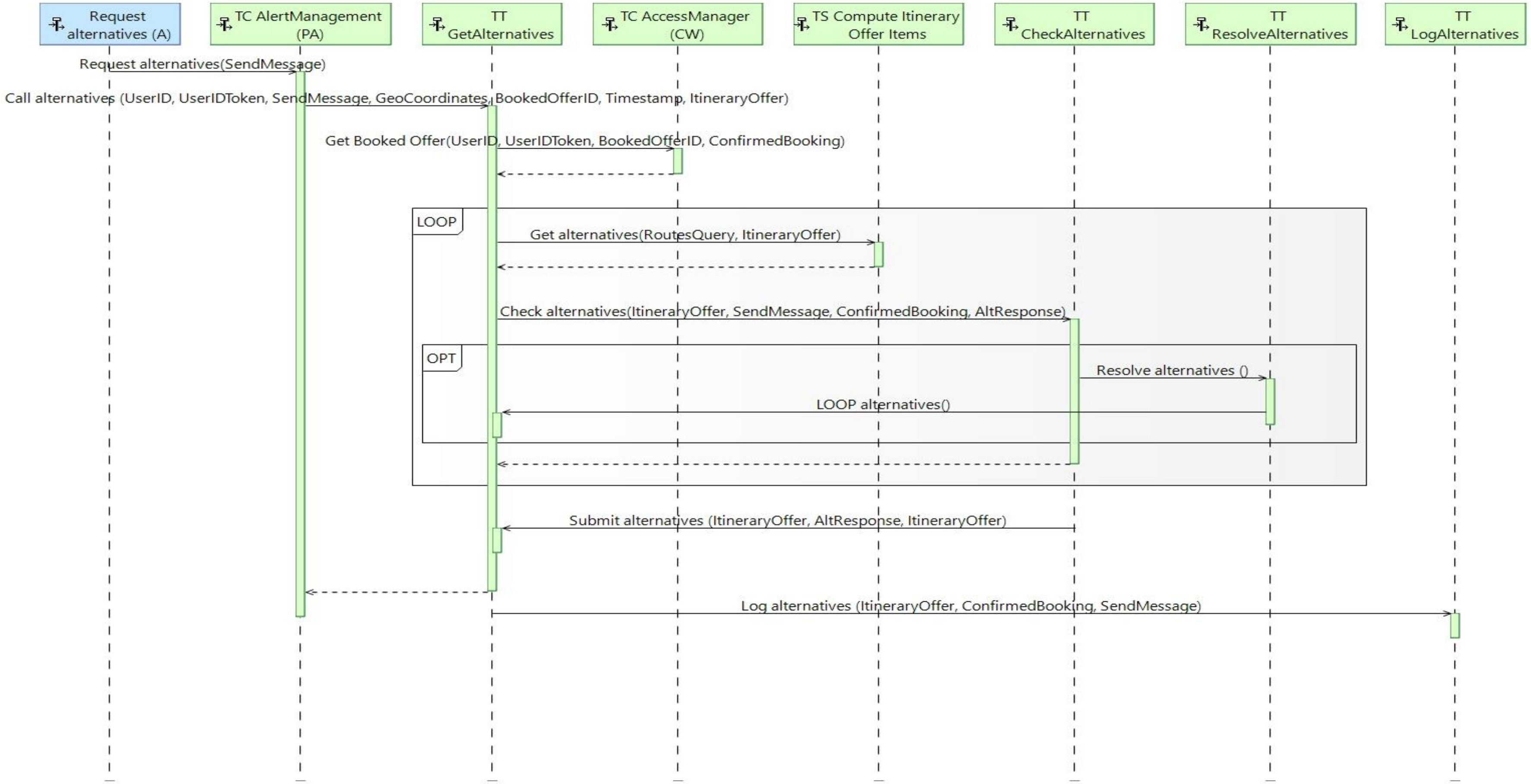
Request alternatives (A)
TC AlertManagement (PA)
TT GetAlternatives
TC AccessManager (CW)
TS Compute Itinerary Offer Items
TT CheckAlternatives
TT ResolveAlternatives
TT LogAlternatives
Request alternatives(SendMessage)
Call alternatives (UserID, UserIDToken, SendMessage, GeoCoordinates, BookedOfferID, Timestamp, ItineraryOffer)
Get Booked Offer(UserID, UserIDToken, BookedOfferID, ConfirmedBooking)
LOOP
Get alternatives(RoutesQuery, ItineraryOffer)
Check alternatives(ItineraryOffer, SendMessage, ConfirmedBooking, AltResponse)
OPT
Resolve alternatives ()
LOOP alternatives()
Submit alternatives (ItineraryOffer, AltResponse, ItineraryOffer)
Log alternatives (ItineraryOffer, ConfirmedBooking, SendMessage)

Figure 6-10: [FS]L Manage Alternatives

During the events processing, every Traveller affected by an event is informed on its impact level (see also Table 4-2: Impact levels & Message types). When a disruption with the highest impact on Traveller's journey has been identified, the original travel plan is not applicable anymore. That means that the recalculation of such journeys is necessary, if the affected Traveller wants to continue traveling. In that case, the Traveller may activate the respective actor's function (Request alternatives) with a request for alternatives suggestions. The Personal Application of the Travel Companion sends through its AlertManagement function such request to the Trip Tracker. The Personal Application has to forward the following information to the TripTracker to enable it calculate the desired alternatives:

- UserID token.
- GeoCoordinates.
- TravelExpert.
- BookedOfferID.

After being invoked the Alternative Manager retrieves the original (affected) BookedOffer based on the UserID and the BookedOfferID by accessing the TC Cloud Wallet. Trip Tracker, thanks to its GetAlternatives function, asks the Travel Shopper (through its Travel Expert Offer Builder function) to calculate an offer of alternatives. It is assumed, that the same Offer Provider who offered the original itinerary, will serve the request for alternatives first. When the offer of alternatives is retrieved from the Travel Shopper, it's necessary to check its content. That's what the Trip Tracker's CheckAlternatives function is designed for.

Sometimes it may happen, that the retrieved offer of alternatives doesn't take into account the event, which originally caused the request for alternatives provision. In such cases the offer must be considered inconvenient and the whole offer of alternatives will be ignored. Another Offer Provider is requested from the appropriate resolver of the Interoperability Framework by the ResolveAlternatives function of the Trip Tracker, or a mobility request is updated. Then a new attempt to get the offer of alternatives can be made. Of course, this loop may be repeated several times if necessary, but the execution time must be considered carefully. Therefore it is assumed that if the first attempt to get the offer of alternatives fail, the repetition will typically take place only once. The list of alternatives can also be shortlisted by excluding the invalid alternatives from the offer.

In the end, as a response to the Traveller's original request for alternatives, either the correct offer of alternatives is submitted or a report of failure with some suggestions is sent by the SubmitAlternatives function of the Trip Tracker to the Traveller (through the AlertManagement function of the Personal Application). In both cases the result of this activity of the Trip Tracker is logged by the LogAlternatives function in order to collect data for Business Analytics purposes.

6.2.5 BA Cooperation

This scenario describes at logical level the Trip Tracker's support to Business Analytics.

Logging

For each logging of different data, it is necessary to find a way to centralize all the logs so they can be further analysed easily. This can be achieved, by using various open source tools.

Structured logs should be used to facilitate their parsing. Fields that can have metrics should be defined for the extraction rather than using a message as a general dump for information as this cannot be easily analysed later by the analytics.

The following figure depicts the architecture overview of an open source tool applied in the IT2Rail project: Elastic Stack (provided by HERE).

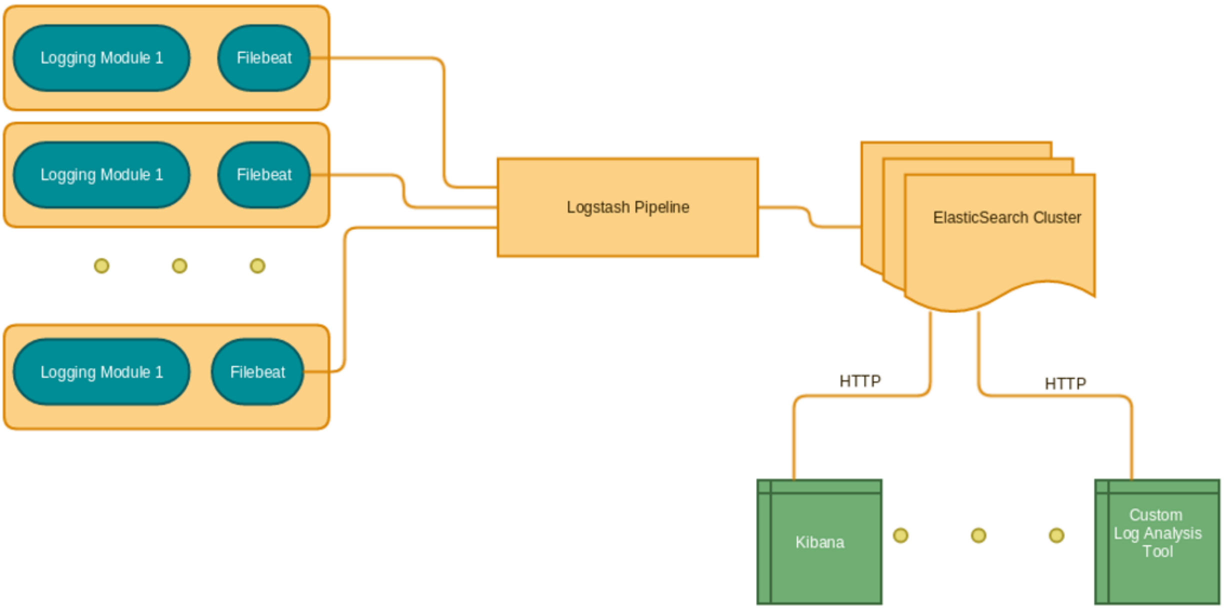

Figure 6-11: Architecture overview of the Elastic Stack

Filebeat is a client that can watch the log files and every time there is new information being logged, it forwards it to the logstash instance. It is a simple datashipper. It is possible to send the different logs - what requests have been made, the decisions done by the different components, the notifications to store them and later extract the different analytics.

Logstash is used as a pipeline. While Filebeat can directly upload the data to Elasticsearch, the Logstash pipeline is needed for filtering/modifying the data in case this is needed.

Elasticsearch stores, searches and analyzes the data and it can be used to store and get real time analysis of the logs.

Kibana is a simple visualization tool and it provides an easy to use interface for elasticsearch. It can also create visualization for queries to elasticsearch. It is possible save often used visualizations and store them in a meaningful dashboard.

During its operation, Trip Tracker collects data, which may be used by the Business Analytics module to compute various statistics and KPIs. These records may be requested by the BA Data Collection function of Business Analytics module. Such requests are served by the OrchestrateBA function and distributed to other Trip Tracker's functions providing requested outputs:

- **ProvideBAdataCEP** function provides information on events' evaluation from Complex Event Processing engine. Information on events is provided in an aggregated format and indicates the number of Travellers affected by a specific event.
- **ProvideBAdataMessages** function provides information on all messages that were sent to Travel Companion to be displayed to the Traveller. Information on messages is supplemented by a set of related information, including a link to the causal event, where appropriate.
- **ProvideBAdataAlternatives** function provides information on alternatives offered upon Traveller's request. It enables Business Analytics to compute statistics related to proposed travel plan reallocation for each customer and all required aggregations (by customer, by transport/service provider, by time). This way Business Analytics gets at least any indication, whether some alternatives were offered to the Traveller or not.

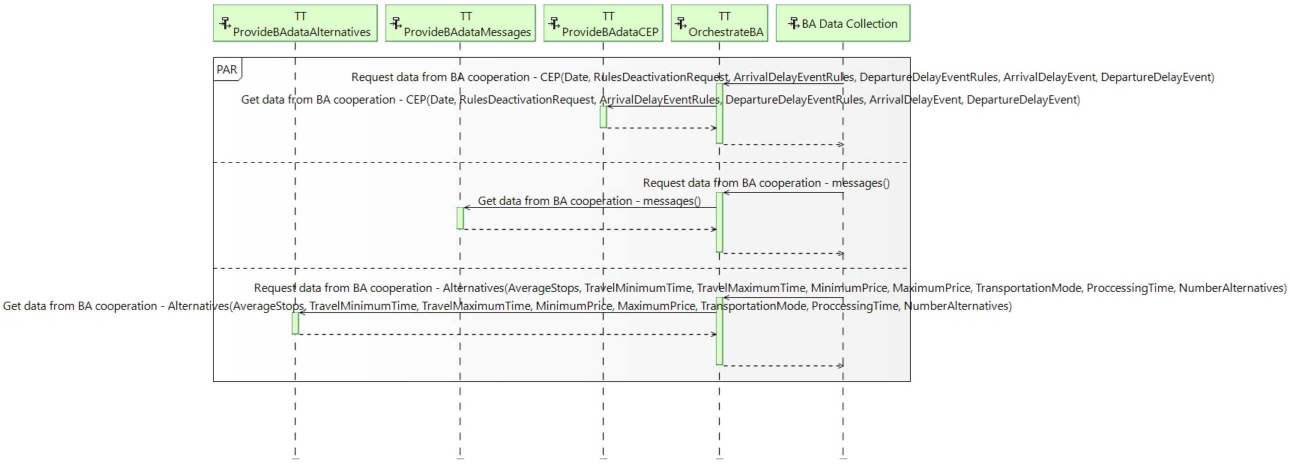
TT ProvideBAdataAlternatives
TT ProvideBAdataMessages
TT ProvideBAdataCEP
TT OrchestrateBA
BA Data Collection
PAR
Request data from BA cooperation - CEP(Date, RulesDeactivationRequest, ArrivalDelayEventRules, DepartureDelayEventRules, ArrivalDelayEvent, DepartureDelayEvent)
Get data from BA cooperation - CEP(Date, RulesDeactivationRequest, ArrivalDelayEventRules, DepartureDelayEventRules, ArrivalDelayEvent, DepartureDelayEvent)
Request data from BA cooperation - messages()
Get data from BA cooperation - messages()
Request data from BA cooperation - Alternatives(AverageStops, TravelMinimumTime, TravelMaximumTime, MinimumPrice, MaximumPrice, TransportationMode, ProccessingTime, NumberAlternatives)
Get data from BA cooperation - Alternatives(AverageStops, TravelMinimumTime, TravelMaximumTime, MinimumPrice, MaximumPrice, TransportationMode, ProccessingTime, NumberAlternatives)

Figure 6-12: [FS]L BA Cooperation

7. EXCHANGES

This chapter consists of three sub-chapters.

- Data model: ensures mutual consistency of the functional or conceptual data model among all the exchanges.
- Interfaces: specifies the interfaces of functions of the Trip Tracker.
- Interfaces details: describes all the details of all involved interfaces.

7.1 DATA MODEL

This chapter contains the functional / conceptual data model of the Trip Tracker interfaces. It must be linked with the ontology.

7.1.1 Activate Tracking

This diagram shows the main exchange items, classes and their relationships concerning the request of the activate tracking data model.

Some class attributes aim at clarifying the contents and use of the class.

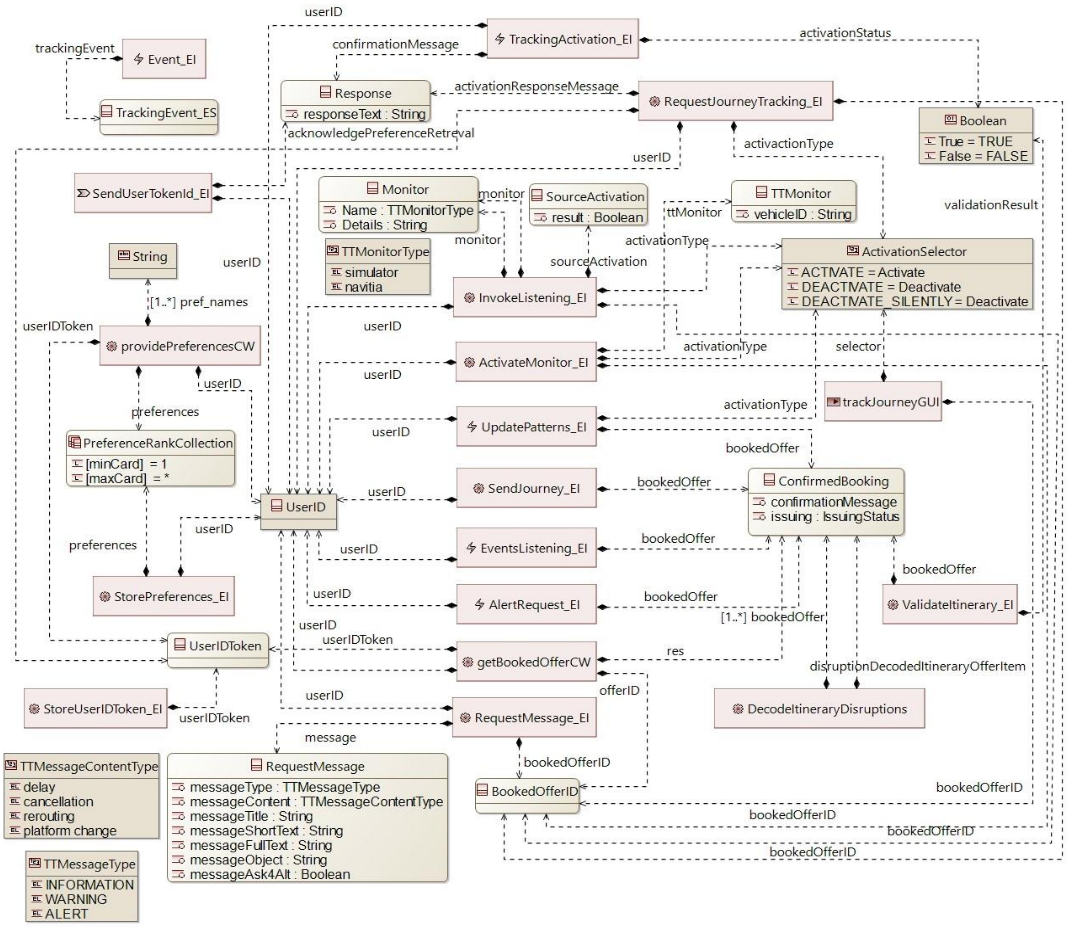

Figure 7-1: Activate Tracking data model

Model reference: *[CDB]L TT Activate Tracking*

This data model specifies the organization of the Activate Tracking with the following classes:

- ActivationSelector – is an Enumeration type of values: activate, deactivate, deactivate_silently.
- Boolean – is a Boolean type with the meaning of the activation or the deactivation of the tracking or the itinerary validation.
- BookedOfferID – is used to uniquely distinguish each BookedOffer.
- ConfirmedBooking – is a sequence of consecutive TravelEpisode(s). Each journey is based on a couple of origin and destination.
- Monitor – identifies the event source.
- PreferenceRankCollection – is a collection of user preferences within the tracking activation process and the aim is to receive those preferences, which are relevant for trip tracking. This enables to customize Trip Tracker's behaviour according to Traveller's preferences.
- RequestMessage – enables to request the Trip Tracker for sending the message to Travel Companion.
- Response – informs about the success/failure of taking the request into account.
- SourceActivation – is a Boolean type with the meaning of the activation or the deactivation of the listening of the event´s source.
- String – is enumerating of the preferences in request.
- TrackingEvent_ES – The event when the validation of the itinerary is not valid.
- TTMessageContentType – corresponds to „Message content“ in TT Preferences.
- TTMessageType – is also defined to easily distinguish the impact of the message (see Table 4-2: Impact levels & Message types).
- TTMonitor – the activation of the particular event source based on vehicleID.
- TTMonitorType – the determination of the particular event source.
- UserID – is used to uniquely distinguish each user.
- UserIDtoken – is used to uniquely distinguish each user.

7.1.2 Perform Event Processing

This diagram shows the main exchange items, classes and their relationships concerning the display of the Perform Event Processing data model.

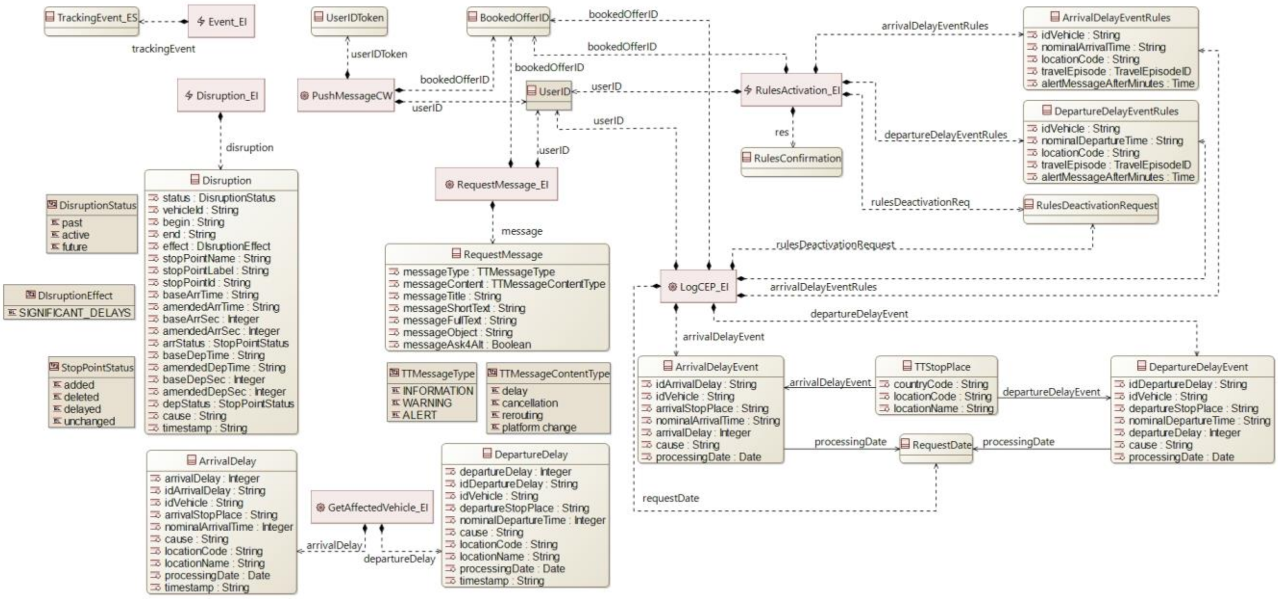

Figure 7-2: Perform Event Processing data model

Model reference: *[CDB]L TT Perform Event Processing*

This data model specifies the organization of the Perform Event Processing with the following classes:

- ArrivalDelay – the event related to the delay of the arrival.
- ArrivalDelayEvent – the structure of the event related to the delay of the arrival which is logged for further processing in the Business Analytics.
- ArrivalDelayEventRules – the activation of the rules related to ArrivalDelayEvent. When the rules are matched, they will be used as a threshold for determining which type of message.
- BookedOfferID – is used to uniquely distinguish each BookedOffer.
- DepartureDelay – the event related to the delay of the departure.
- DepartureDelayEvent – the structure the event related to the delay of the departure which is logged for further processing in the Business Analytics.
- DepartureDelayEventRules – the activation of the rules related to DepartureDelayEvent. When the rules are matched, they will be used as a threshold for determining which type of message.
- Disruption – the disruption identified by event source which may have the impact on a journey.
- DisruptionEffect – the type of the disruption's effect which Trip Tracker is able to recognize.
- DisruptionStatus – determines the disruption's status, whether the disruption is active, whether it happened in the past or is expected/scheduled in the future.
- RequestDate – the timestamp identifying when the request for the activation/deactivation of the rules was sent.
- RequestMessage – enables to request the Trip Tracker for sending the message to Travel Companion.
- RulesConfirmation – confirmation of the rule's activation or deactivation.

- RulesDeactivationRequest – the request related to the deactivation of the rules.
- StopPointStatus – the determination of the departure´s status: added, deleted, delayed, unchanged.
- TrackingEvent_ES – is the detected event from the events sources.
- TTMessageContentType – corresponds to „Message content“ in TT Preferences.
- TTMessageType – is also defined to easily distinguish the impact of the message (see Table 4-2: Impact levels & Message types).
- TTStopPlace – determines the place where the delay of the arrival/departure occurred.
- UserID – is used to uniquely distinguish each user.
- UserIDtoken – is used to uniquely distinguish each user.

7.1.3 User Interface

This diagram shows the main exchange items, classes and their relationships concerning the request of the User Interface data model.

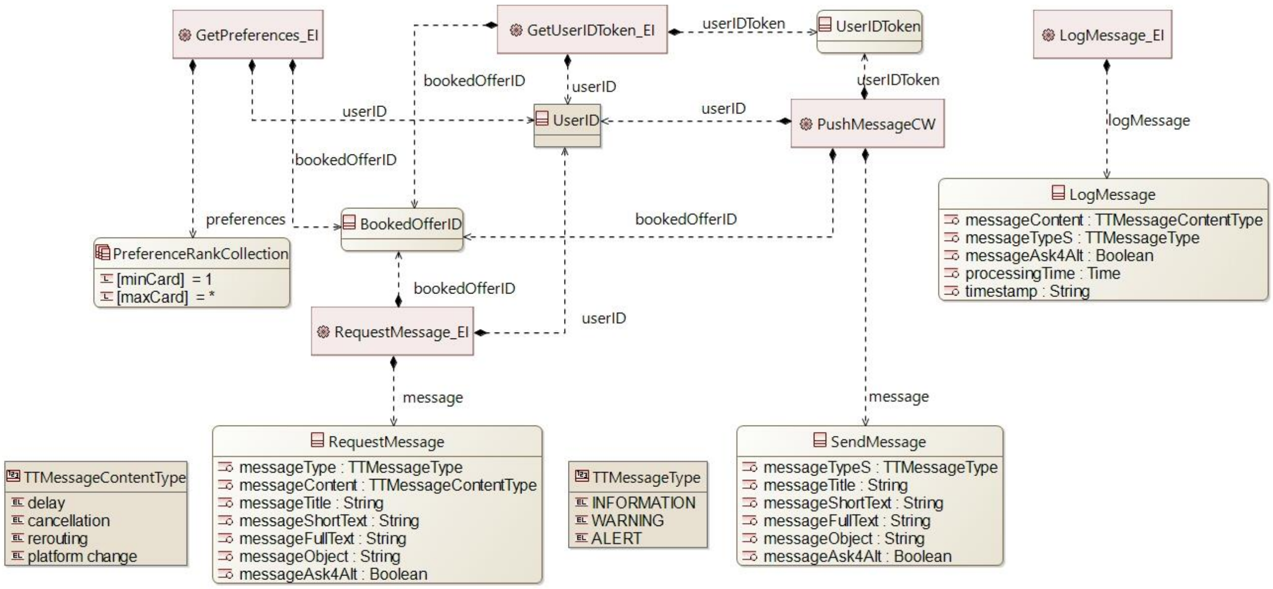

Figure 7-3: User Interface data model

Model reference: *[CDB]L TT User Interface*

This data model specifies the organization of the User Interface with the following classes:

- BookedOfferID – is used to uniquely distinguish each BookedOffer.
- LogMessage – the data structure related to the message which is logged for further processing in the Business Analytics.
- PreferenceRankCollection – is a collection of user preferences within the tracking activation process and the aim is to receive those preferences, which are relevant for trip tracking. This enables to customize Trip Tracker’s behaviour according to Traveller’s preferences.
- RequestMessage – enables to request the Trip Tracker for sending the message to Travel Companion.

- SendMessage – aims at sending a message to the Traveller through the Travel Companion.
- TTMessageContentType – corresponds to Message content in TT Preferences.
- TTMessageType – is also defined to easily distinguish the impact of the message (see Table 4-2: Impact levels & Message types).
- UserID – is used to uniquely distinguish each user.
- UserIDtoken – is used to uniquely distinguish each user.

7.1.4 Trip Tracker's Preferences

This diagram shows the main exchange items, classes and their relationships related to the Trip Tracker's preferences. This set of Trip Tracker's preferences is only a sample for the project purposes. These preferences can be modified and extended in the follow-up projects. The summary of Trip Tracker's preferences is listed in the Table 4-1: Trip Tracker's Preferences.

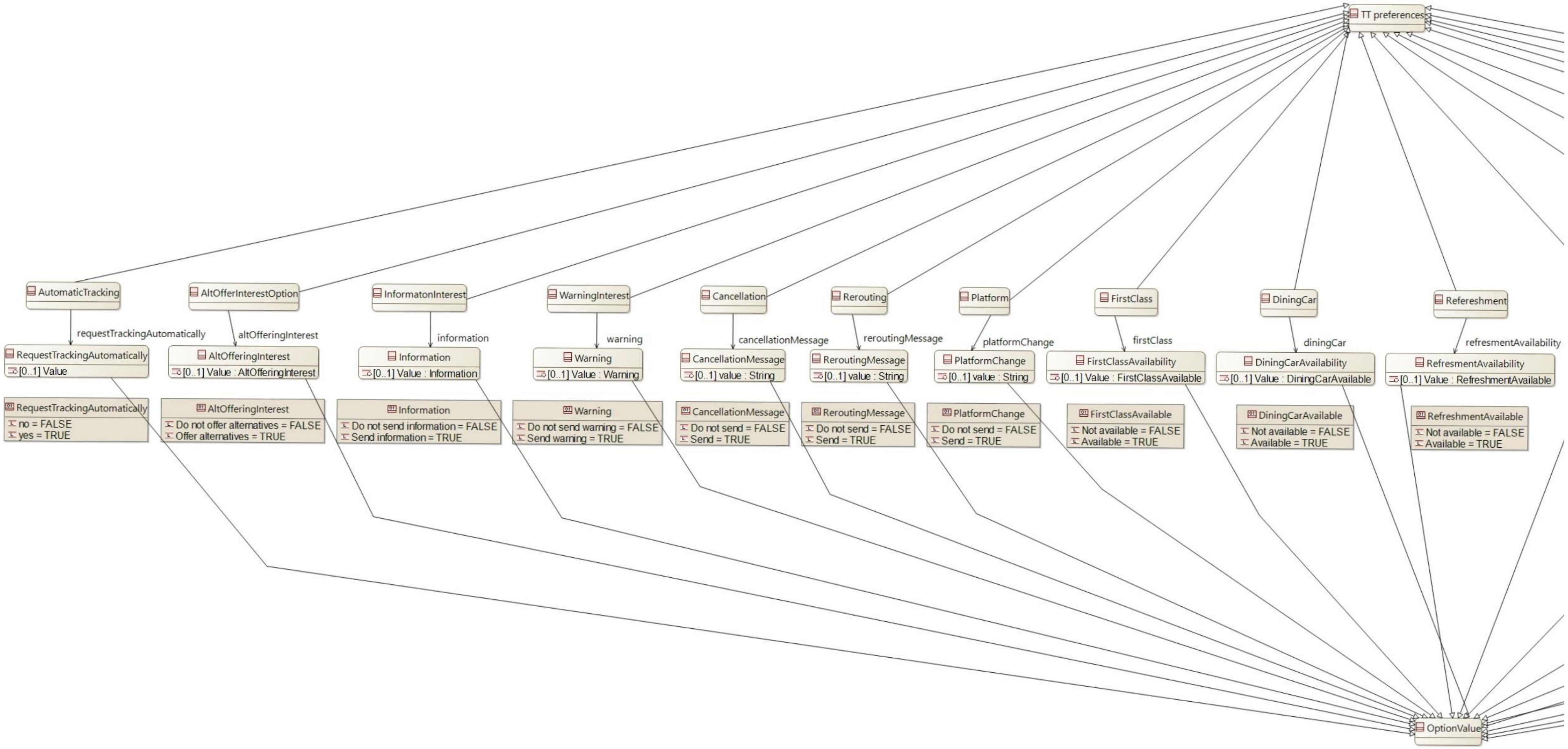

Figure 7-4: Trip Tracker's Preferences data model (part 1)

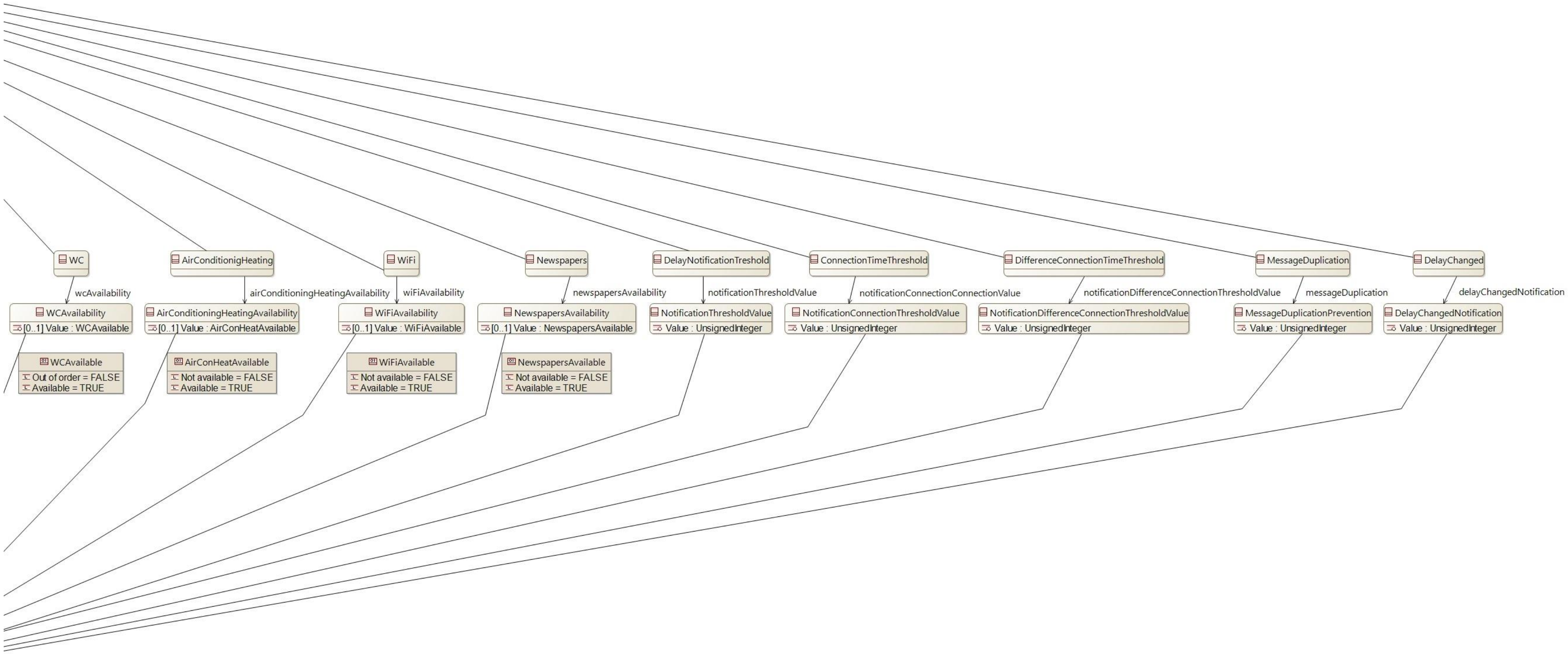

Figure 7-5: Trip Tracker's Preferences data model (part 2)

Model reference: *[CDB]L TT Preferences*

This data model specifies the organization of Trip Tracker's preferences with the following classes:

- AutomaticTracking – Request journey tracking automatically, when a new itinerary is stored in the CW.
- AltOfferInterestOption – Ask about interest in alternatives offering, where applicable.
- InformationInterest; WarningInterest – Enable/disable receiving messages of selected type.
- Cancellation; Rerouting; Platform; FirstClass; DiningCar; Refreshment; WC; AirConditioningHeating; WiFi; Newspapers – display only messages selected by the user, i.e. avoid displaying unwanted messages.
- DelayNotificationTreshold – Don't display a warning, when the delay is less than given time.
- ConnectionTimeTreshold – Don't display a warning, when remaining connection time is less than given value.
- DifferenceConnectionTimeTreshold – Don't display a warning, when the difference between remaining and minimum connection time is less than given value.
- MessageDuplication – Don't display the same message again, if less than given time since the last display elapsed.
- DelayChanged – Consider the delay unchanged (so the delay message still the same), if it has changed by less than given value.

7.1.5 Manage Alternatives

This diagram shows the main exchange items, classes and their relationships concerning the request of the Manage Alternatives data model.

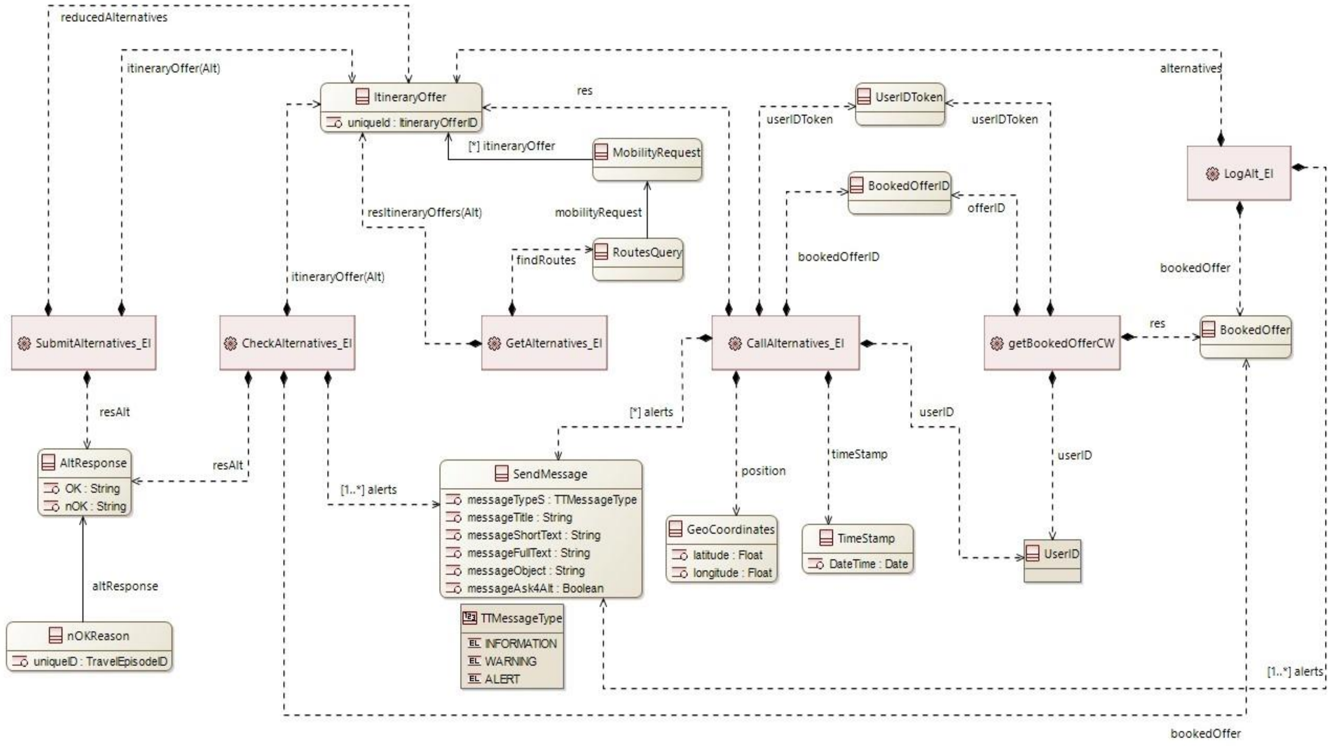

Figure 7-6: Manage Alternatives data model

Model reference: *[CDB]L TT Manage Alternatives*

This data model specifies the organization of the Manage Alternatives with the following classes:

- AltResponse – is the response whether the check of alternatives was successful or not.
- BookedOffer – is a sequence of consecutive TravelEpisode(s). Each journey is based on a couple of origin and destination.
- BookedOfferID – is used to uniquely distinguish each BookedOffer.
- GeoCoordinates – Coordinates of a point expressed in decimal format (WS84).
- ItineraryOffer – is the response in the form of the offer of the alternatives to the request of the alternatives.
- MobilityRequest – is the Traveller’s query for travel information about a specific itinerary.
- nOKReason – identifies travel episodes for which the checking was not successful.
- RoutesQuery – the selected route by the Traveller – the origin, the destination and the transport mode.
- SendMessage – aims at sending a message to the Traveller through the Travel Companion.
- TimeStamp – the moment when the Trip Tracker received the request for the calculation of alternatives.
- TTMessageType – is also defined to easily distinguish the impact of the message (see Table 4-2: Impact levels & Message types).
- UserID – is used to uniquely distinguish each user.
- UserIDToken – is used to uniquely distinguish each user.

7.1.6 BA Cooperation

The following diagram shows the main exchange items, classes and their relationships concerning the request of the BA Cooperation - CEP data model.

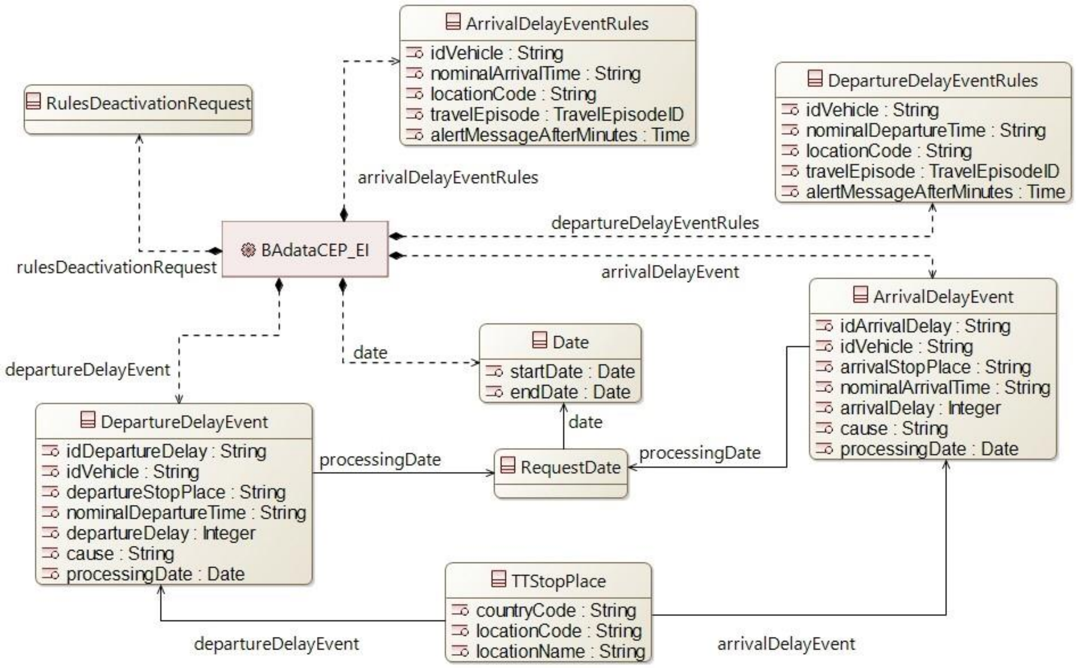

Figure 7-7: BA Cooperation - CEP data model

Model reference: *[CDB]L TT BA Cooperation CEP*

This data model specifies the organization of the BA Cooperation - CEP with the following classes:

- ArrivalDelayEvent – the event related to the delay of the arrival.
- ArrivalDelayEventRules – the activation of the rules related to ArrivalDelayEvent. When the rules are matched, they will be used as a threshold for determining which type of message.
- DepartureDelayEvent – the event related to the delay of the departure.
- Date – date on the basis of which all the required information about the events shall be sent.
- DepartureDelayEventRules – the activation of the rules related to DepartureDelayEvent. When the rules are matched, they will be used as a threshold for determining which type of message.
- RequestDate – identifies the date when the request for the activation/deactivation of the rules was sent.
- RulesDeactivationRequest – the request related to the deactivation of the rules.
- TTStopPlace – the determination of the place where the delay of the arrival/departure occurred.

The following diagram shows the main exchange items, classes and their relationships concerning the request of the BA Cooperation - alternatives data model.

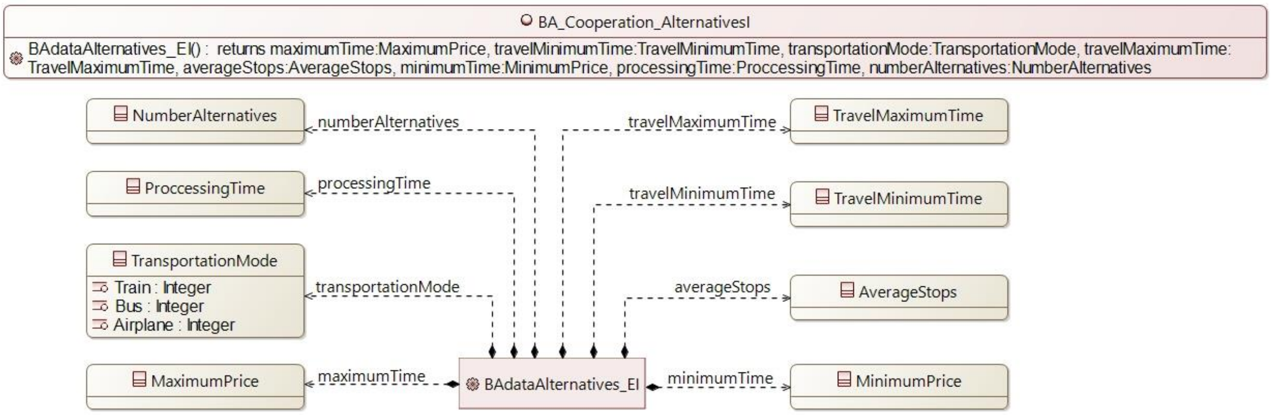
BA_Cooperation_AlternativesI
BAdataAlternatives_EI() : returns maximumTime:MaximumPrice, travelMinimumTime:TravelMinimumTime, transportationMode:TransportationMode, travelMaximumTime:TravelMaximumTime, averageStops:AverageStops, minimumTime:MinimumPrice, processingTime:ProccessingTime, numberAlternatives:NumberAlternatives
NumberAlternatives
numberAlternatives
travelMaximumTime
TravelMaximumTime
ProccessingTime
processingTime
travelMinimumTime
TravelMinimumTime
TransportationMode
Train : Integer
Bus : Integer
Airplane : Integer
transportationMode
averageStops
AverageStops
MaximumPrice
maximumTime
BAdataAlternatives_EI
minimumTime
MinimumPrice

Figure 7-8: BA Cooperation - alternatives data model

Model reference: *[CDB]L TT BA Cooperation Alternatives*

This data model specifies the organization of the BA Cooperation - alternatives with the following classes:

- AverageStops – the average of the AverageAlternatives field.
- MaximumPrice – the maximum of the AveragePrice field.
- MinimumPrice – the minimum of the AveragePrice field.
- NumberAlternatives – number of alternatives submitted in given period.
- ProccessingTime – average of the ProcessingTime field.
- TransportationMode – count of the TransportStops field by type.
- TravelMaximumTime – the average of the TravelMaximumTime field.
- TravelMinimumTime – the average of the TravelMinimumTime field.

7.2 COMPONENTS AND INTERFACES

As introduced earlier, the Trip Tracker consists of six main parts, which correspond to the six main components described in the chapter 4.1. They interact with each other and with other IT2Rail modules through a series of interfaces (some of them provided by the Trip Tracker components, other provided by other IT2Rail modules and used, or required, by the Trip Tracker components), which are also depicted in Figure 7-9 and which are described in this section.

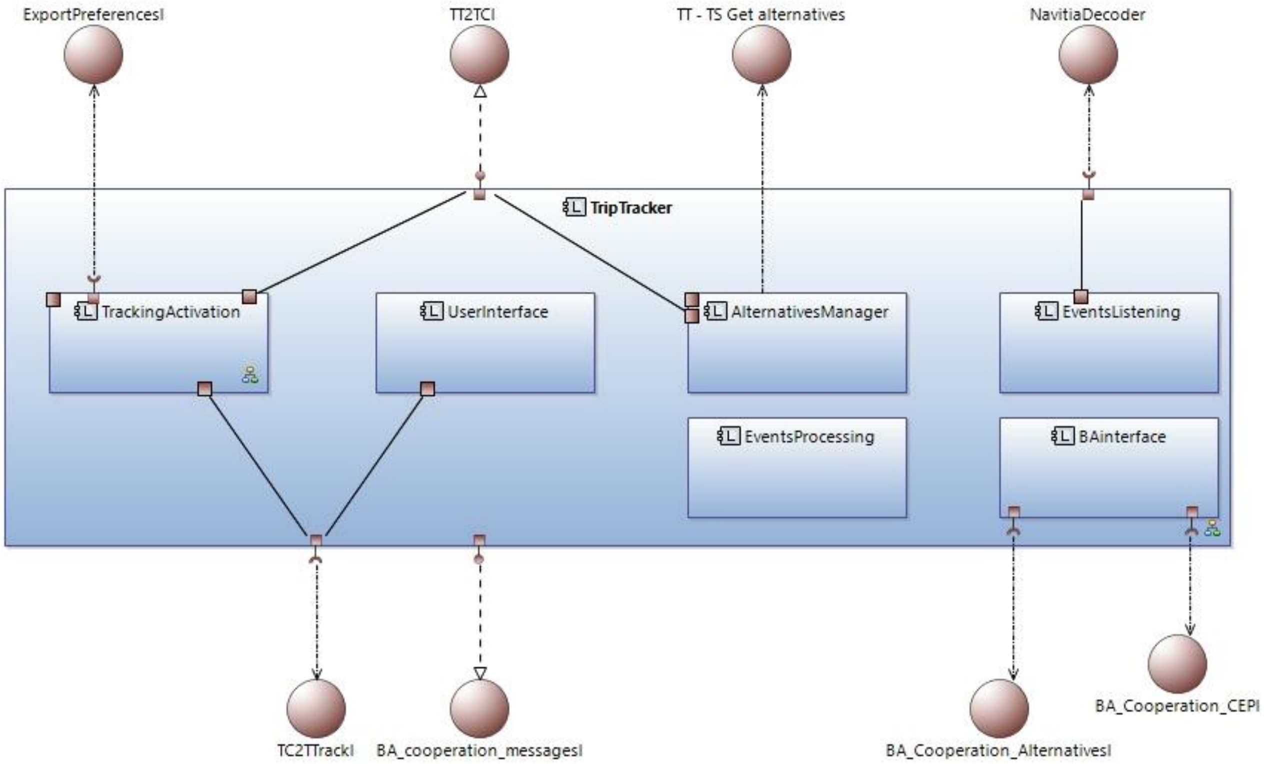

Figure 7-9: Trip Tracker's main components and interfaces

7.2.1 Trip Tracker interfaces

This figure depicts the interfaces with their inputs and outputs between the Trip Tracker and other IT2Rail modules.

These interfaces are depicted in Figure 7-10. In particular, the interfaces are as follows:

- BA Cooperation AlternativesI – this interface represents an exchange of logged alternatives data between the Trip Tracker and the Business Analytics.
- BA Cooperation CEPI – this interface represents an exchange of logged CEP data between the Trip Tracker and the Business Analytics.
- NavitiaDecoder – an interface between the Trip Tracker and the Interoperability Framework represents the encoding/decoding of the event structure retrieved from the Navitia platform.
- TT-TS Get alternatives – an interface between the Trip Tracker and the Travel Shopper which represents the request for alternatives.
- TT2TCI – specifies interfaces between the Trip Tracker and the Personal Application of the Travel Companion which represent the request for tracking activation and the request for alternatives calculation.
- TC2TTrackI – this interface offers the Trip Tracker to push messages to the Travel Companion and to request the BookedOffer.
- ExportPreferencesI – this interface supports the retrieval of user preferences by the Travel Shopper and the Trip Tracker. The interface supports two mechanisms to retrieve

preferences: by listing the names of the preferences to be retrieved; or simply by asking for all the preferences to be retrieved.

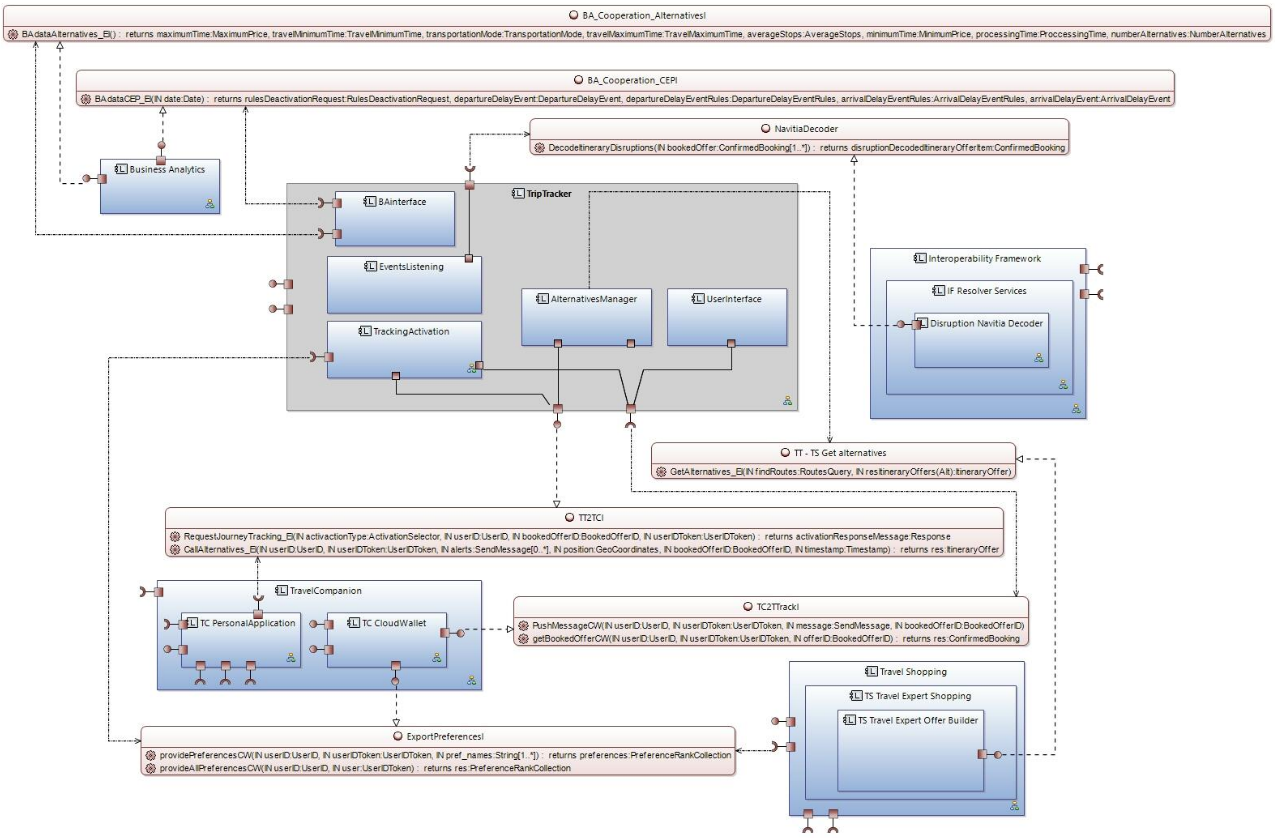
BA_Cooperation_AlternativesI
BAdataAlternatives_EI() : returns maximumTime:MaximumPrice, travelMinimumTime:TravelMinimumTime, transportationMode:TransportationMode, travelMaximumTime:TravelMaximumTime, averageStops:AverageStops, minimumTime:MinimumPrice, processingTime:ProccessingTime, numberAlternatives:NumberAlternatives
BA_Cooperation_CEPI
BAdataCEP_EI(IN date:Date) : returns rulesDeactivationRequest:RulesDeactivationRequest, departureDelayEvent:DepartureDelayEvent, departureDelayEventRules:DepartureDelayEventRules, arrivalDelayEventRules:ArrivalDelayEventRules, arrivalDelayEvent:ArrivalDelayEvent
NavitiaDecoder
DecodeItineraryDisruptions(IN bookedOffer:ConfirmedBooking[1..*]) : returns disruptionDecodedItineraryOfferItem:ConfirmedBooking
Business Analytics
TripTracker
BAinterface
EventsListening
AlternativesManager
UserInterface
TrackingActivation
Interoperability Framework
IF Resolver Services
Disruption Navitia Decoder
TT - TS Get alternatives
GetAlternatives_EI(IN findRoutes:RoutesQuery, IN resItineraryOffers(Alt):ItineraryOffer)
TT2TCI
RequestJourneyTracking_EI(IN activactionType:ActivationSelector, IN userID:UserID, IN bookedOfferID:BookedOfferID, IN userIDToken:UserIDToken) : returns activationResponseMessage:Response
CallAlternatives_EI(IN userID:UserID, IN userIDToken:UserIDToken, IN alerts:SendMessage[0..*], IN position:GeoCoordinates, IN bookedOfferID:BookedOfferID, IN timestamp:Timestamp) : returns res:ItineraryOffer
TravelCompanion
TC PersonalApplication
TC CloudWallet
TC2TTrackI
PushMessageCW(IN userID:UserID, IN userIDToken:UserIDToken, IN message:SendMessage, IN bookedOfferID:BookedOfferID)
getBookedOfferCW(IN userID:UserID, IN userIDToken:UserIDToken, IN offerID:BookedOfferID) : returns res:ConfirmedBooking
Travel Shopping
TS Travel Expert Shopping
TS Travel Expert Offer Builder
ExportPreferencesI
providePreferencesCW(IN userID:UserID, IN userIDToken:UserIDToken, IN pref_names:String[1..*]) : returns preferences:PreferenceRankCollection
provideAllPreferencesCW(IN userID:UserID, IN user:UserIDToken) : returns res:PreferenceRankCollection

Figure 7-10: Trip Tracker interfaces

7.3 INTERFACES DETAILS

This chapter contains the details of each interface defined in Trip Tracker's class diagrams in the previous chapter.

Interface name	**BAdataAlternatives**		
Description	Exchange of logged alternatives data between the Trip Tracker and the Business Analytics.		
Requestor	BA Data Collection		
Provider	TT OrchestrateBA		
Release	**CREL**	**AREL**	**FREL**
			Complete
Pre-conditions	Logging of alternatives data		
Post-conditions	Further processing in the Business Analytics		
Request / Input			
Response / Output	AverageStops		
	MaximumPrice		
	MinimumPrice		
	NumberAlternatives		
	ProcessingTime		
	TransportationMode		
	TravelMaximumTime		
	TravelMinimumTime		
Exceptions			
Notes			

Interface name	**BAdataCEP**		
Description	Exchange of logged CEP data between the Trip Tracker and the Business Analytics.		
Requestor	BA Data Collection		
Provider	TT OrchestrateBA		
Release	**CREL**	**AREL**	**FREL**
			Complete
Pre-conditions	Logging of CEP data		
Post-conditions	Further processing in the Business Analytics		
Request / Input	Date		
Response / Output	ArrivalDelayEvent		
	ArrivalDelayEventRules		
	DepartureDelayEvent		
	DepartureDelayEventRules		
	RulesDeactivationRequest		
Exceptions			
Notes			

<table>
<tr><td><b>Interface name</b></td><td colspan="3"><b>CallAlternatives</b></td></tr>
<tr><td><b>Description</b></td><td colspan="3">Triggering journey recalculation</td></tr>
<tr><td><b>Requestor</b></td><td colspan="3">TC AlertManagement (PA)</td></tr>
<tr><td><b>Provider</b></td><td colspan="3">TT GetAlternatives</td></tr>
<tr><td rowspan="2"><b>Release</b></td><td><b>CREL</b></td><td><b>AREL</b></td><td><b>FREL</b></td></tr>
<tr><td></td><td></td><td>Complete</td></tr>
<tr><td><b>Pre-conditions</b></td><td colspan="3">Request alternatives</td></tr>
<tr><td><b>Post-conditions</b></td><td colspan="3">Get booked offer</td></tr>
<tr><td rowspan="5"><b>Request / Input</b></td><td colspan="3">UserID</td></tr>
<tr><td colspan="3">UserIDToken</td></tr>
<tr><td colspan="3">SendMessage</td></tr>
<tr><td colspan="3">GeoCoordinates</td></tr>
<tr><td colspan="3">BookedOfferID</td></tr>
<tr><td><b>Response / Output</b></td><td colspan="3">ItineraryOffer</td></tr>
<tr><td><b>Exceptions</b></td><td colspan="3"></td></tr>
<tr><td><b>Notes</b></td><td colspan="3"></td></tr>
</table>

<table>
<tr><th>Interface name</th><th colspan="3">DecodeItineraryDisruptions</th></tr>
<tr><td><b>Description</b></td><td colspan="3">Decoding of the event structure provided by the Navitia platform</td></tr>
<tr><td><b>Requestor</b></td><td colspan="3">TT EventsListening</td></tr>
<tr><td><b>Provider</b></td><td colspan="3">IF Decode Navitia Disruptions</td></tr>
<tr><td rowspan="2"><b>Release</b></td><td><b>CREL</b></td><td><b>AREL</b></td><td><b>FREL</b></td></tr>
<tr><td></td><td></td><td>Complete</td></tr>
<tr><td><b>Pre-conditions</b></td><td colspan="3">Identify available sources of events</td></tr>
<tr><td><b>Post-conditions</b></td><td colspan="3">Invoke listening of Navitia event´s source</td></tr>
<tr><td><b>Request / Input</b></td><td colspan="3">ConfirmedBooking</td></tr>
<tr><td><b>Response / Output</b></td><td colspan="3">ConfirmedBooking</td></tr>
<tr><td><b>Exceptions</b></td><td colspan="3"></td></tr>
<tr><td><b>Notes</b></td><td colspan="3"></td></tr>
</table>

Interface name	**GetAlternatives**		
Description	The request for alternatives calculation		
Requestor	TT GetAlternatives		
Provider	TS Compute Itinerary Offer Items		
Release	**CREL**	**AREL**	**FREL**
			Complete
Pre-conditions	Get booked offer		
Post-conditions	Check alternatives		
Request / Input	RoutesQuery		
Response / Output	ItineraryOffer		
Exceptions			
Notes			

Interface name	**getBookedOfferCW**		
Description	Request for the entire structure of the itinerary		
Requestor	TT, OrchestrateActivation, TT GetAlternatives		
Provider	TC AccessManager (CW)		
Release	**CREL**	**AREL**	**FREL**
			Complete
Pre-conditions	Provide preferences, Call Alternatives		
Post-conditions	Get Alternatives		
Request / Input	UserID		
	UserIDToken		
	BookedOfferID		
Response / Output	BookedOffer		
Exceptions			
Notes			

<table>
<tr><td><b>Interface name</b></td><td colspan="3"><b>providePreferencesCW</b></td></tr>
<tr><td><b>Description</b></td><td colspan="3">Retrieve all the preferences of a given user, which are relevant for trip tracking, from the Travel Companion</td></tr>
<tr><td><b>Requestor</b></td><td colspan="3">TC AccessManager (CW)</td></tr>
<tr><td><b>Provider</b></td><td colspan="3">TT GetPreferences</td></tr>
<tr><td rowspan="2"><b>Release</b></td><td><b>CREL</b></td><td><b>AREL</b></td><td><b>FREL</b></td></tr>
<tr><td></td><td></td><td>Complete</td></tr>
<tr><td><b>Pre-conditions</b></td><td colspan="3">Request preferences retrieval</td></tr>
<tr><td><b>Post-conditions</b></td><td colspan="3">Store preferences</td></tr>
<tr><td rowspan="3"><b>Request / Input</b></td><td colspan="3">UserID</td></tr>
<tr><td colspan="3">UserIDToken</td></tr>
<tr><td colspan="3">String</td></tr>
<tr><td><b>Response / Output</b></td><td colspan="3">PreferenceRankCollection</td></tr>
<tr><td><b>Exceptions</b></td><td colspan="3"></td></tr>
<tr><td><b>Notes</b></td><td colspan="3"></td></tr>
</table>

<table>
<tr><th>Interface name</th><th colspan="3">PushMessageCW</th></tr>
<tr><td><b>Description</b></td><td colspan="3">Sends the message into the Cloud Wallet to display it at user’s device</td></tr>
<tr><td><b>Requestor</b></td><td colspan="3">TT SendMessage</td></tr>
<tr><td><b>Provider</b></td><td colspan="3">TC AccessManager (CW)</td></tr>
<tr><td rowspan="2"><b>Release</b></td><td><b>CREL</b></td><td><b>AREL</b></td><td><b>FREL</b></td></tr>
<tr><td></td><td>Partial</td><td>Complete</td></tr>
<tr><td><b>Pre-conditions</b></td><td colspan="3">Request for the sending of the message</td></tr>
<tr><td><b>Post-conditions</b></td><td colspan="3">Log message</td></tr>
<tr><td rowspan="4"><b>Request / Input</b></td><td colspan="3">UserID</td></tr>
<tr><td colspan="3">UserIDToken</td></tr>
<tr><td colspan="3">BookedOfferID</td></tr>
<tr><td colspan="3">SendMessage</td></tr>
<tr><td><b>Response / Output</b></td><td colspan="3"></td></tr>
<tr><td><b>Exceptions</b></td><td colspan="3"></td></tr>
<tr><td><b>Notes</b></td><td colspan="3"></td></tr>
</table>

Interface name	**RequestJourneyTracking**		
Description	Triggering request for activation of given journey tracking		
Requestor	TC AlertManagement (PA)		
Provider	TT OrchestrateActivation		
Release	**CREL**	**AREL**	**FREL**
	Partial	Partial	Complete
Pre-conditions	Tracking requested by the Traveller		
Post-conditions	Request preferences retrieval		
Request / Input	ActivationSelector		
	UserID		
	UserIDToken		
	BookedOfferD		
Response / Output	Response		
Exceptions			
Notes			

8. ANNEX

8.1 IMPLEMENTATION CHOICES

N°	Decisions	Status
1	The transfer of the itinerary content between the Travel Companion and the Trip Tracker, as it represents a large amount of data, was resolved as follows: • Within the Request journey tracking, only the itineraryID is sent • After calling GetItinerary, the entire itinerary content is returned from the Cloud Wallet of Travel Companion to the Trip Tracker	Done
2	CEP assesses the impact of events CEP doesn't take the user's preferences into account, as that pertains to the User interface component CEP can't decide whether to send a message to the user or not, this is based on user's preferences known to the User interface component only	Done
3	When an event causes a search for alternatives, the Travel Companion does not directly ask the Travel Shopper for it, but the Trip Tracker. The Trip Tracker is fully responsible for the offer of alternatives suggested to the Traveller (through the Travel Companion).	Done

Table 8-1: Trip Tracker's implementation choices

9. REFERENCES

[1] IT2Rail consortium. Deliverable D5.2: Travel Companion Specifications.