VDV Die Verkehrs-
unternehmen

VDV-Schrift 431-2

09/2019

Echtzeit-Kommunikations und Auskunftsplattform EKAP / Real time communication and assistance platform EKAP

Teil 2: EKAP interface Beschreibung V1.3 / Part 2: EKAP interface description V1.3

Gesamtbearbeitung
Ausschuss für Telematik und Informationssysteme (ATI)
Ausschuss für Kunden-dialog, -service und –information (K^3)

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

Das dieser VDV-Schrift zugrundeliegende Vorhaben IP-KOM-ÖV wurde mit Mitteln des Bundesministeriums für Wirtschaft und Energie unter dem Förderkennzeichen 19P10003 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

VDV-Schrift 431-2 09/2019

Echtzeit-Kommunikations und Auskunfts-plattform EKAP / Real time communication and assistance platform EKAP

Teil 2: EKAP Schnittstellenbeschreibung V1.3/ Part 2: EKAP interface description V1.3

Sachbearbeitung
Unterausschuss für intermodal transport control systems (UA itcs)
K^3-Kernteam Information

Autorenverzeichnis
M. Sc. Frank Englert, TU Darmstadt, Darmstadt (V1.0)
B. Sc. Stephan Großberndt, side by site GmbH & Co. KG, Cologne (from V1.1)
Dipl.-Inf. Günther Gruber, MENTZ GmbH, Munich (V1.0)
Dipl.-Math. Peter von Grumbkow, HaCon Ing.-Ges. mbH, Hannover
M. Sc. Malte Herlitze, MENTZ GmbH, Munich (from V1.3)
Dipl.-Ing. Stephan Hörold, TU Ilmenau, Ilmenau (V1.0)
Dipl.-Ing. (FH) Waldemar Isajkin, Init, Karlsruhe (V1.2)
Dipl.-Inf. Christine Keller, TU Dresden, Dresden (V1.0)
Dipl.-Math. Werner Kohl, MENTZ GmbH, Munich (up to V1.2)
Dipl.-Medieninf. Romina Kühn, TU Dresden, Dresden (V1.0)
Dipl.-Medienwiss. Cindy Mayas, TU Ilmenau, Ilmenau (V1.0)
Dipl.-Ing. ETH Walter Meier-Leu, Weisskopf Engineering AG, Schaffhausen (V1.0)
Dipl.-Ing. Berthold Radermacher, VDV, Cologne
Dipl.-Inform. Anselmo Stelzer, TU Darmstadt, Darmstadt (V1.0)
Dipl.-Inf. Katja Tietze, TU Dresden, Dresden (V1.0)
Dipl.-Ing. (FH) Andreas Wehrmann, VDV, Cologne (V1.0)
Dipl.-Ing. Dirk Weißer, Init, Karlsruhe

Der Anwender ist für die sorgfältige und ordnungsgemäße Anwendung der Schrift verantwortlich. Stellt der Anwender Gefährdungen oder Unregelmäßigkeiten im Zusammenhang mit der Anwendung dieser Schrift fest, wird eine unmittelbare Benachrichtigung an den VDV erbeten. Eine Haftung des VDV oder der Mitwirkenden an der Schrift ist, soweit gesetzlich zulässig, ausgeschlossen.

© Verband Deutscher Verkehrsunternehmen e. V. Köln 2015 | Alle Rechte, einschließlich des Nachdrucks von Auszügen, der fotomechanischen oder datenverarbeitungstechnischen Wiedergabe und der Übersetzung, vorbehalten.

The user is responsible for diligent and correct application of the guideline. If the user discovers dangers or abnormalities with regard to the application of this guideline, a direct notification to VDV is requested. Liability of VDV or participants engaged in the guideline is excluded to the extent permitted by law.

© Verband Deutscher Verkehrsunternehmen e. V. Cologne 2017 | All rights reserved, including rights of reproduction of excerpts, photomechanical reproduction or reproduction related to technical data processing and of translation.

Vorwort

Auf Initiative des VDV und gefördert durch das BMWi begann im September 2010 das Forschungs- und Standardisierungsprojekt

Internet Protokoll basierte Kommunikationsdienste im öffentlichen Verkehr (IP-KOM-ÖV).

Das Projekt wird von 14 Partnern aus Industrie, Universitäten und Verkehrsunternehmen getragen. Es dient der Erarbeitung moderner Kommunikationskonzepte für die umfassende und kontinuierliche Fahrgastinformation.

Eine umfassende Fahrgastinformation stellt heutzutage ein entscheidendes Wettbewerbsmerkmal im öffentlichen Personenverkehr dar, nicht nur im Vergleich mit anderen Verkehrsunternehmen, sondern auch im Vergleich zum Individualverkehr.

Bereits heute ist es üblich, dass Verkehrsunternehmen ihre Fahrgäste nicht nur über die geplanten Fahrten informieren, sondern auch Echtzeitinformationen z. B. zu Verspätungen, Störungen oder Fahrtzieländerungen bereitstellen. Diese Informationen werden zum einen über öffentliche Anzeiger bzw. Ansagen in Fahrzeugen oder an Haltestellen allen dort befindlichen Personen zur Verfügung gestellt. Zum anderen lassen sich solche Informationen mit speziellen Applikationen oder über Web-Angebote individuell abfragen.

Bislang ist es aber nicht möglich, Fahrgäste im öffentlichen Verkehr direkt mit Informationen zu ihrer persönlich relevanten Fahrt zu versorgen, den Fahrgast also auch im Störungsfall mit Hilfe des öffentlichen Verkehrs auf dem schnellsten Weg zu seinem Ziel zu führen.

Die weit verbreiteten Smartphones und Tablets bieten hierfür vielfältige Möglichkeiten und ermöglichen eine hohe Akzeptanz der Benutzer. Die Informationsübertragung erfolgt dabei IP-basiert und sollte bevorzugt zwischen einem zentralen Informations-Server und dem Kundenendgerät erfolgen. Für den Fall, dass der zentrale Datenserver nicht erreichbar ist, sollte auch eine Kommunikation zwischen Kundenendgerät und Fahrzeug möglich sein.

Das Forschungs- und Standardisierungsprojekt IP-KOM-ÖV arbeitet deshalb an drei Schwerpunkten (vgl. Abbildung 1).

Erster Schwerpunkt (grün in Abbildung 1) ist die Spezifikation eines performanten IP-basierten Kommunikationsprotokolls im Fahrzeug (IBIS-IP, VDV301). Dabei geht es zum einen darum, den gewachsenen Bedürfnissen der Fahrgastinformation gerecht zu werden und zum anderen um die Definition einer IP-basierten Schnittstelle zur Übertragung der Informationen vom Fahrzeug zum mobilen Kundenendgerät. Hierzu wird der in den achtziger Jahren entwickelte IBIS-Wagenbus aus der VDV-Schrift 300 auf eine moderne Ethernet-Informationsarchitektur umgesetzt.

Zweiter Schwerpunkt (rot in Abbildung 1) ist die individuelle Fahrgastinformation unter Verwendung mobiler Geräte des Fahrgasts (Smartphones, Tablet-PC u. ä.) Hierzu wurden im ersten Schritt die Bedürfnisse von Fahrgästen zu individuellen Informationen ermittelt. Im zweiten Schritt werden einheitliche Schnittstellen zwischen der Echtzeit-Kommunikations- und Auskunftsplattform (EKAP) und den mobilen Kundenendgeräten bzw. zwischen der EKAP und den Hintergrundsystemen entwickelt. Hierbei werden ausschließlich die Datenmodellierungen und Architekturen erforscht und spezifiziert. Aufbauend auf diesen Datenmodellierungen werden semantische Modelle erarbeitet, die helfen, die Fahrgastinformationsdaten für

Kommunikationsdienste auf Basis von innovativen Technologien des Semantic Web zur Verfügung zu stellen. Die Entwicklung einer Applikation für mobile Endgeräte ist ausdrücklich nicht vorgesehen.

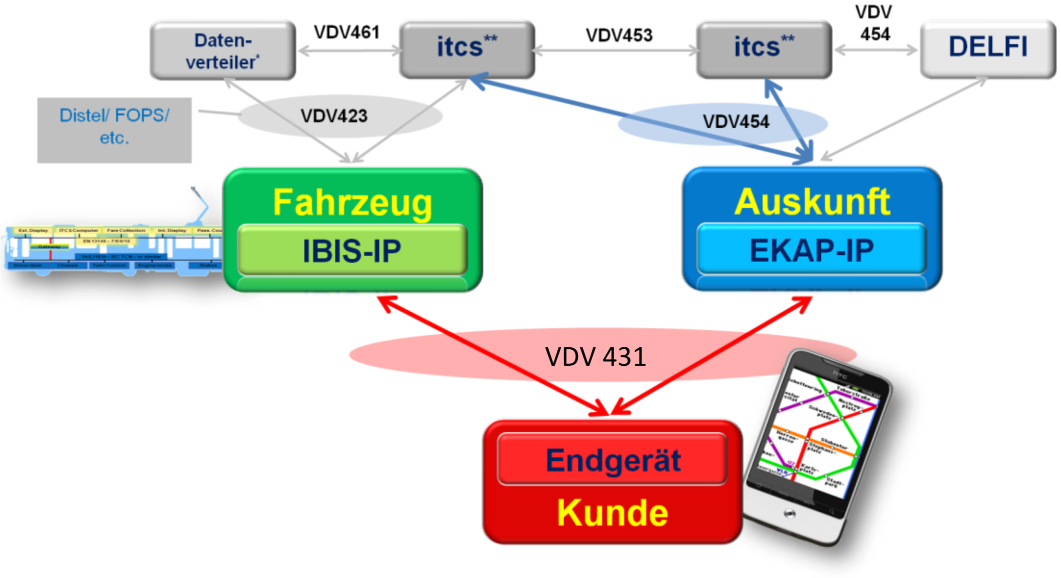

Abbildung 1: Umfeld und Schwerpunkte im Projekt IP-KOM-ÖV

Dritter Schwerpunkt (blau in Abbildung 1) ist die Definition und Schaffung einer Echtzeit-Kommunikations- und Auskunftsplattform (EKAP). Die EKAP bündelt Informationen von itcs- und anderen Auskunfts- und Informationssystemen und stellt die Vielzahl an Informationen über geeignete Schnittstellen den Applikationen auf den Kundenendgeräten zur Verfügung. Diese Plattform ermöglicht es, Kunden dynamisch mit individuellen Störungsmeldungen versorgen zu können.

Neben den Forschungsarbeiten ist die Standardisierung der Ergebnisse ein wesentliches Ziel des Projektes, um eine nachhaltige Nutzung zu gewährleisten.

Darüber hinaus wird die Praxistauglichkeit dieses neuen Standards in Labor- und Feldtests verifiziert.

Foreword

The research and standardisation project began in September 2010 at the initiative of VDV and promoted by BMWi:

Internet-protocol based communication services in public transport (IP-KOM-ÖV).

The project is executed by 14 partners from the industry, universities and transport companies. It helps in the development of more modern communication concepts for comprehensive and ongoing passenger information.

Nowadays, comprehensive passenger information is an essential factor of competition in the sector of public passenger transport, not only in comparison with other transport companies but also in comparison with individual transport.

It is common practice that transport companies inform their passengers not only about planned journeys but also real-time information, such as delays, accidents or destination changes. This information is provided either on a public display or in the form of announcements in vehicles or at stops to all the persons present there. Such information can also be individually requested using special applications or websites.

However, it is still not possible to provide information about personal journeys directly to the passengers in public transport and to guide the passenger to his destination the fastest way via public transport in case of a breakdown.

The widely used smartphones and tablets offer diverse options and facilitate high acceptance of users. In this case, information is transmitted in an “IP-based” way and this should preferably be done between a central information server and the customer’s end device. In case the central data server cannot be reached, communication between the customer’s end device and the vehicle should be possible.

Hence, the research and standardisation project IP-KOM-ÖV words on three focus areas (see Figure 1).

The first area of focus (depicted in green in figure 1) is the specification of an efficient IP-based communication protocol in the vehicle (IBIS-IP, VDV301). On the one hand, this involves responding to the increased needs of passenger information and, on the other, defining an IP-based interface for information transmission from vehicle to customer’s mobile end device. To this end, the IBIS vehicle bus developed in the eighties from VDV guideline 300 is converted to have a modern Ethernet information architecture.

The second area of focus (depicted in red in figure 1) is individual passenger information using mobile devices of the passenger (smartphones, tablet-PC etc.) For this, the needs of the passengers with regard to individual information have been determined in the first step. In the second step, uniform interfaces are developed between the real-time communication and assistance platform (EKAP) and the mobile end devices of the customers or between EKAP and background systems. Only data models and architectures are researched and specified here.

Semantic models are developed on the basis of these data models, which help in providing passenger information for communication services on the basis of innovative technologies of the Semantic Web. The development of an application for mobile end devices is expressly not provided.

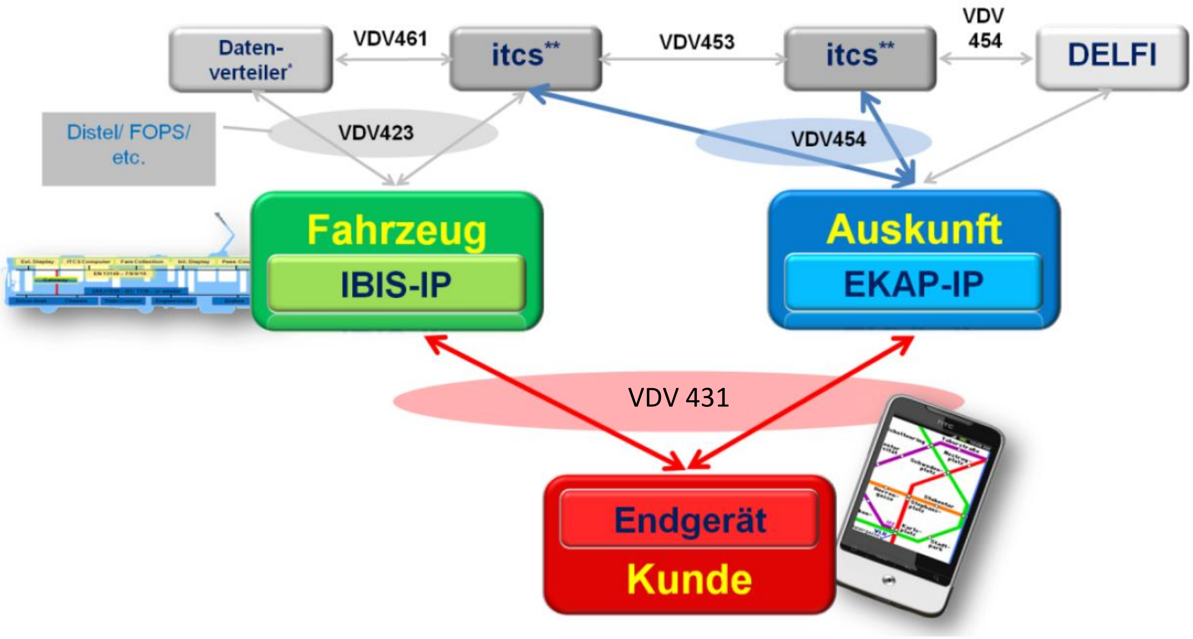

Figure 1: Environment and areas of focus in project IP-KOM-ÖV

Fahrzeug (IBIS-IP) -> Vehicle (IBIS-IP)
Endgerät (Kunde) -> End device (Customer)
Auskunft (EKAP-IP) -> Information (EKAP-IP)

The third area of focus (depicted in blue in Figure 1) is the definition and establishment of a real-time communication and assistance platform (EKAP). EKAP pools information from itcs and other assistance and information systems and provides a multitude of information to applications on the end devices of the customers via suitable interfaces. This platform makes it possible to provide the customers with individual malfunction messages dynamically.

In addition to research activities, standardisation of the results is an essential objective of the project in order to ensure sustainable use.

Furthermore the practicality of these new standards is verified in laboratory and field tests.

Inhaltsverzeichnis / Content

Abkürzungen, Begriffe / Abbreviations, Terms

Es gelten die in VDV-Schrift 430 Teil 1, 2013 und in VDV-Schrift 431, Teil 1, 2013 festgelegten Begriffsbestimmungen.

The abbreviations stated in VDV regulation 430 part 1, 2013 and in VDV regulation 431, part 1, 2013 shall apply. Moreover, the abbreviations stated in table 283 are used.

Begriff/Term	Beschreibung	Description
HTTP	Hypertext Transfer Protocol	
	Das auf TCP/IP basierende Übertragungsprotokoll wird hauptsächlich im Internet zum Informationsaustausch eingesetzt.	Transmission protocol based on TCP/IP is mainly used on the Internet for exchanging information
IFOPT	Identification of Fixed Objects in Public Transport (CEN, EN 28701:2012, 2012)	
JourneyWeb		British standard for linking regional timetable information systems for a national information across Great Britain (Department for Transport, 2012)
Transmodel	Reference Data Model for Public Transport (CEN, EN 12896:2006)	
TRIAS	Travellers‘ Realtime Information and Advisory Standard	
	Dieses Akronym bezeichnet die in diesem Dokument definierte Familie von Schnittstellendiensten.	This acronym means the family of interface services defined in this document.
UML	Unified Modeling Language	
	Standardisierte grafische Modellierungssprache zur Spezifikation von Software.	Standardised graphic modelling language for specification of software.

1 Einleitung

In diesem Dokument werden die Dienste, die in VDV-Schrift 430 und VDV-Schrift 431-1 beschrieben sind, als XML-Schnittstellen definiert. Dadurch entstehen Schnittstellenstandards, die es Software-Entwicklern und Unternehmen erlauben, Anwendungen zu realisieren, womit mobile Apps der Fahrgäste, Fahrzeuge, Portalsysteme und echtzeitfähige Auskunftssysteme (EKAPs) miteinander kommunizieren.

Bei der Ausarbeitung dieser Schnittstellendefinitionen wurde Wert darauf gelegt, Kompatibilität zu anderen Standards auf dem Sektor des öffentlichen Verkehrs herzustellen. Hier sind vor allem TransModel als Begriffsglossar, IFOPT für die Modellierung von Haltestellen, SIRI für den Austausch von Echtzeitdaten und für sein ausgefeiltes Nachrichtenaustauschverfahren, sowie JourneyWeb und DELFI als Schnittstellen zum Abrufen von Fahrplaninformationen und Verbindungsauskünften zu nennen.

1 Introduction

The services, which are described in VDV guideline 430 and VDV guideline 431-1, are defined as XML interfaces in this document. This results in interface standards, which allow software developers and companies to develop applications by means of which mobile apps of the passengers, vehicles, portal systems and real-time information systems (EKAPs) communicate with one another.

When preparing these interface definitions, emphasis has been placed on establishing compatibility with other standards in the sector of public transport. TransModel as glossary of terms, IFOPT for modelling stops, SIRI for exchanging real-time data and for its sophisticated messaging process as well as JourneyWeb and DELFI as interfaces to access schedule information and route planners are worth mentioning here.

2 Anwendungsbereich

Die in diesem Dokument definierten Schnittstellen spezifizieren Dienste, die

- zwischen mobilen Apps und Fahrzeugen des ÖV
- zwischen Portalsystemen und Auskunftssystemen (EKAPs) als Hintergrundsystem und
- EKAP und Fahrgastinformationsystemen (z.B. Anschlussanzeigen in Fahrzeugen, Abfahrtsanzeigen und Übersichtsanzeigen an Haltestellen, etc.)

Verwendung finden.

In erster Linie soll der Fahrgast informiert werden. Es gibt aber auch Dienste, bei denen der Fahrgast von sich aus aktiv wird, so z. B. beim Haltewunsch oder der Anschlussvoranmeldung.

2 Area of application

The interfaces defined in this document specify services that are used

- between mobile apps and vehicles of public transport
- between portal systems and information systems (EKAPs) as background system and
- EKAP und passenger information systems (e.g. connection displays in vehicles, departure displays and overview displays at stops, etc.).

Mainly, the passenger should be informed. But there are also services, in which the passenger is voluntarily active, e.g. stop request or pre-announcement of connection.

3 Notation der XML-Elemente und -Strukturen

Die in diesem Dokument vorgestellen TRIAS[1]-Schnittstellen werden mit Hilfe von XML-Schema definiert. Die Objekte, die über die Schnittstelle ausgetauscht werden, liegen folglich als XML-Elemente vor. Die Beschreibung der XML-Elemente wird in diesem Dokument in einer Tabellenform vorgenommen, die aus SIRI (CEN, TS 15531 Part 1, 2011) stammt. Sie ist sehr kompakt und übersichtlich und bietet eine Vielzahl an strukturellen Informationen, die ansonsten nur in der XML-Schema-Definition sichtbar wird. Dieses Kapitel erläutert die Notation der Tabellenform, die ab Kapitel 7 intensiv genutzt wird.

Alle Namen von Elementen, Datentypen und Attributen sind in Englisch gehalten, um eine etwaige Normierung auf europäischer Ebene vorzubereiten und den Austausch mit europäischen Partnern zu erleichtern.

3.1 Darstellung von XML-Elementen im Text

In diesem Dokument soll eine konsistente Notation der XML-Elemente helfen, technisch wichtige Information beim Lesen bereit zu stellen.

- XML-Elemente werden in Groß-Klein-Schreibweise (Upper Camel Case) fett und kursiv geschrieben, z. B.: ***VehicleJourneyRef***. Die Elementnamen sind – wo immer möglich und sinnvoll – an Begriffe aus TransModel angelehnt. Fehlt in TransModel ein geeigneter Begriff für ein Konzept oder Objekt, so wurde versucht, den entsprechenden Begriff aus JourneyWeb oder das passende Konzept aus DELFI zu übernehmen.
- Datenypen werden kursiv dargestellt, z. B.: *xsd:boolean*.
- Code-Beispiele werden in kleinerer Schrift wiedergegeben.

3.2 Darstellung von Beziehungen

Beziehungen zwischen Objekten können mittels

- impliziter Mechanismen,
- internen Referenzen oder
- externen Referenzen

ausgedrückt werden. Ein impliziter Mechanismus ist z. B. das Enthaltensein eines Elements in einem anderen. Damit wird eine unmittelbare Kindbeziehung ausgedrückt. Eine interne Referenz ist ein Objektschlüssel, der innerhalb der Schnittstelle definiert wird (z. B. ein Identifikator einer Meldung). Eine externe Referenz ist ein Objektschlüssel, der außerhalb der Schnittstelle festgelegt wird (z. B. eine Haltestellennummer). Externe Referenzen bestehen manchmal auch aus zusammengesetzten Schlüsseln (siehe die ausführliche Darstellung in Kapitel 5).

Es ist wichtig, den Unterschied zwischen einem Identifikator (Objektschlüssel) und einer Referenz auf das Objekt festzuhalten. In TRIAS gelten folgende Regeln:

[1] Travellers Realtime Information and Advisory Standard

- Ein Identifikator ist ein Kindelement des definierenden Elements, das einen eindeutigen Code (Primärschlüssel) für das definierende Element angibt. Diese Identifikatoren enden auf ein signalisierendes Hauptwort wie „Code“ oder „Identifier“ (manchmal auch „Number“ in SIRI), z. B. erhält eine Fahrplanfahrt (Journey) den Schlüssel ***JourneyCode***.
- Wird ein Objekt von einem anderen Objekt aus referenziert, endet das referenzierende Element (Fremdschlüssel) auf “Ref”. Zum Bespiel lautet die Referenz auf eine Fahrplanfahrt (etwas aus einer Abfahrtstafel heraus): JourneyRef.
- Die Instanz eines Objekts und die Referenz darauf verwenden einen gemeinsamen zugrunde liegenden Datentyp. Zum Bespiel sind JourneyCode und JourneyRef beide vom Typ JourneyCodeType.

Tabellennotation von XML-Strukturen

In diesem Dokument werden XML-Strukturen in einer Tabellennotation dargestellt (vgl. Tabelle 1). Für jedes wichtige TRIAS-Anfrage/Antwort-Element findet sich eine eigene Tabelle. Weitere Tabellen werden für alle wesentlichen Kindelemente, aus denen die komplexen Strukturen aufgebaut sind, angegeben. Um Platz zu sparen, werden die Spaltenüberschriften nur im Beispiel in Tabelle 1 angezeigt und bei allen folgenden Tabellen nicht wiederholt. In den Tabellen wird ein konsistenter Satz an Regeln zur Beschreibung der XML-Elemente und der daran geknüpften Bedingungen verwendet.

Gruppierung	Elementname	Min: Max	Datentyp	Erläuterung
ContinuousServiceStructure			*+Structure*	Eine Fahrgastbewegung mit Hilfe eines kontinuierlichen, nicht fahrplangebundenen Verkehrsmittels.
	a ***ContinuousMode***	**-1:1**	*walk | demandResponsive | replacementService*	Modalität für kontinuierliche Verkehre
	b ***IndividualMode***		*walk | cycle | taxi | self-drive-car | others-drive-car | motorcycle | truck*	Verkehrsmittelmodalität für Individualverkehr
DatedService	***OperatingDay***	**1:1**	*→OperatingDay*	Betriebstag der Fahrt.
	VehicleRef	0:1	*→Vehicle*	Fahrzeug-ID.
ServiceJourney	***JourneyRef***	**1:1**	*→Journey*	Fahrt-ID.
LineIdentity	***LineRef***	**1:1**	*→Line*	Linien-ID.
	DirectionRef	**1:1**	*→Direction*	Richtungs-ID.
Service	***Mode***	**1:1**	*+Mode*	Verkehrsmitteltyp.
	PublishedLineName	**1:***	*InternationalText*	Liniennummer oder -name, wie in der Öffentlichkeit bekannt.
	OperatorRef	0:1	*→Operator*	Operator-ID.
	RouteDescription	0:*	*InternationalText*	Beschreibung des Fahrwegs.
	Via	0:*	*+ServiceViaPoint*	Wichtige Halte auf dem Fahrweg.
	Attribute	0:*	*+GeneralAttribute*	Hinweise und Attribute (mit Klassifikationen) zur Fahrt.
ServiceOrigin	*OriginStopPointRef*	0:1	*→StopPoint*	ID des ersten Haltepunkts der Fahrt; Starthaltestelle.
	OriginText	0:*	*InternationalText*	Name des ersten Haltepunkts der Fahrt, der Starthaltestelle.
ServiceDestination	*DestinationStopPointRef*	0:1	*→StopPoint*	ID des letzten Haltepunkts der Fahrt; Endhaltestelle.
	DestinationText	1:*	*InternationalText*	Name des letzten Haltepunkts der Fahrt, der Endhaltestelle oder Fahrtziel.
	SituationFullRef	0:*	*+SituationFullRef*	Verweis auf eine Störungsnachricht. Diese Nachricht kann im Kontext der Meldung (ResponseContext) zu finden sein oder auf anderem Wege bekannt gemacht werden.

Tabelle 1: Beispiel (aus einem späteren Abschnitt) für die tabellarische Notation einer XML-Struktur

3.2.1 Gruppierung

In der ersten Spalte befindet sich gelegentlich ein Bezeichner, der die Elemente in sinnvolle Gruppierungen einteilt, z. B. *Service* oder *ServiceOrigin*. Dies dient rein zu Dokumentationszwecken und entspricht in den meisten Fällen den Namen einer XML-Gruppe, die im XML-Schema verwendet wurde. Die Verwendung von Gruppierungen hat nur den Zweck, die Elemente zu organisieren und damit für mehr Klarheit und bessere Wiederverwendbarkeit zu sorgen.

3.2.2 Elementname

Elementnamen werden kursiv in der zweiten Spalte wiedergegeben, z. B. *OperatingDay*. Handelt es sich um ein verpflichtendes Element, so wird es **fett** gedruckt. Optionale Elemente werden nicht fett gedruckt. Der Name der Struktur selbst ist links oben in der Tabelle angegeben.

Elemente, die geerbt (XML: "derived by extension") oder anonym verwendet werden, tragen im Namensfeld drei Doppelpunkte ":::" zur Kennzeichnung (siehe beispielhaft Table 5).

Multiplizität & Choice (Min:Max)

Die Bedingungen, ob ein Element verpflichtend oder optional ist oder ob es einfach oder mehrfach innerhalb des übergeordneten Elements auftreten kann, werden in der dritten Spalte Min:Max angegeben. Dabei werden die üblichen UML-Konventionen „min:max" angewendet, so steht z. B. „0:1" für ein optionales, einfaches Element, „**1:1**" zeigt ein verpflichtendes, einfaches Element an, „0:*" steht für ein optionales, mehrfaches Element usw. Verpflichtende Elemente werden **fett** gedruckt.

In manchen Fällen muss ein Element aus seiner Menge ausgewählt werden (XML-Choice). Dies wird durch ein vorangestelltes Minuszeichen symbolisiert, z. B. „**-1:1**". In diesem Fall steht vor dem Elementnamen noch ein Kleinbuchstabe, der die Auflistung der Wahlmöglichkeiten anzeigt. Bei optionalen Auswahlmöglichkeiten (Choices) steht im Min-Wert eine Null: „-0:1".

3.2.3 Datentyp

Die Datentypen werden in der vierten Spalte kursiv angegeben, z. B. *InternationalText*. Falls der Namensraum (namespace) vom TRIAS-Namensraum abweicht, wird er mitangegeben, z. B. *„xs:dateTime"* oder *„siri:PtSituationElement"*.

- Ein komplexer Datentyp, der selbst Strukturen als Kindelemente enthält, wird in der Spalte Datentyp mit *„+Structure"* gekennzeichnet.
- Wo Elemente als Referenzen (Fremdschlüssel) auf andere Objekte verwendet werden, wird als Datentyp der Typ des referenzierten Objekts mit vorangestelltem Pfeil verwendet. Zum Beispiel *„→StopPoint"* als Typ einer Referenz (*StopPointRefStructure*) auf ein Objekt vom Typ „StopPointType".
- Aufzählungstypen (Enumerated types) werden an den meisten Stellen unmittelbar mit den verwendbaren Werten dargestellt, z. B. *„walk | cycle"*. Nur in einigen Fällen mit sehr umfangreichen Aufzählungen, die an mehreren Stellen wiederverwendet werden, wird ein Typ deklariert und referenziert.
- Um Platz zu sparen, werden bei der Angabe der Datentypen Abkürzungen verwendet, z. B. wird auf die Endungen "Structure" und "Type" durchgehend verzichtet. Statt bespielsweise *„InternationalTextStructure"* wird also immer *„InternationalText"* als Datentyp angegeben.

3.2.4 Erläuterung

Alle Elemente erhalten in der letzten Spalte eine Erläuterung ihres Verwendungszwecks. An vielen Stellen wird auf weitere Passagen im Text hingewiesen, so z. B. bei komplexen Kindelementen an die Stelle, an der ihre Tabellenbeschreibung zu finden ist. An einigen Stellen ist die Erläuterung zu umfangreich und würde die Tabellenform sprengen. Dann finden sich diese Anmerkungen im Text unterhalb der Tabelle.

3 Notation of XML elements and structures

The TRIAS[2] interfaces introduced in this document are defined with the help of XML schema. The objects, which are exchanged via the interface, are thus available as XML elements. The XML elements are described in a tabular form in this document which originates from SIRI (CEN, TS 15531 part 1, 2011). The description is very concise and clear and offers a range of structural information which is otherwise visible only in the XML schema definition. This chapter explains the notation of tabular form which is intensively used from chapter 7 onwards.

All the names of elements, data types and attributes are in English to facilitate standardisation at European level and to simplify the exchange with European partners.

3.1 Representation of XML elements in text

In this document, a consistent notation of XML elements should help provide technically important information when reading.

- XML elements are written in upper camel case, bold and cursive, e.g.: **VehicleJourneyRef.** The element name are based on terms from TransModel wherever possible and meaningful. If a suitable term for a concept or object is not available in TransModel, we have tried to acquire an appropriate term from JourneyWeb or an appropriate concept from DELFI.
- Data types are in cursive, e.g.: xsd:boolean.
- Code examples are given in lower case.

3.2 Representation of relations

Relations between objects can be expressed with the help of

- implicit mechanisms,
- internal references or
- external references.

An implicit mechanism is, for example, the inclusion of an element into another. This way a direct child relation is expressed. An internal reference is an object key which is defined within the interface (e.g. an identifier of a message). An external reference is an object key which is defined outside the interface (e.g. a stop number). Sometimes external references also consist of combined keys (see the detailed representation in chapter 5).

It is important to determine the difference between an identifier (object key) and a reference to the object. The following rules apply in TRIAS:

- An identifier is a child element of the defining element which specifies a unique code (primary key) for the defining element. These identifiers end with a signal word like “code” or “identifier” (sometimes even “number” in SIRI), e.g. a journey contains the key **JourneyCode**.

[2] Travellers Real-time Information and Advisory Standard

- If an object is referenced by another object, the referencing element (foreign key) ends with “Ref”. For example, the reference to a journey (from a stop event) is: JourneyRef.
- An object instance and its reference use a common underlying data type. For example, **JourneyCode** and **JourneyRef** are both of the type **JourneyCodeType**.

Table notation of XML structures

The XML structures are given in a table notation in this document (see Table 5). There is a separate table for every important TRIAS request/response element. Additional tables are provided for all the important child elements, from which the complex structures are built. In order to save space, the column headings are only displayed in table 1 and not repeated in all the following tables. A consistent set of rules is used in the tables to describe XML elements and the related conditions.

Grouping	Element name	Min: Max.	Data type	Description
ContinuousServiceStructure			*+Structure*	A passenger movement using a continuous, non-timetabled service
	a ***ContinuousMode***	**-1:1**	*walk | demandResponsive | replacementService*	Modality for continuous transport operations
	b ***IndividualMode***		*walk | cycle | taxi | self-drive-car | others-drive- car | motorcycle | truck*	Public transport modality for individual transport
DatedService	***OperatingDay***	**1:1**	*→OperatingDay*	Operating day of the journey.
	VehicleRef	0:1	*→Vehicle*	Vehicle-ID.
ServiceJourney	***JourneyRef***	**1:1**	*→Journey*	Journey-ID.
LineIdentity	***LineRef***	**1:1**	*→Line*	Line ID.
	DirectionRef	**1:1**	*→Direction*	Direction ID
Service	***Mode***	**1:1**	*+Mode*	Type of public transport.
	PublishedLineName	**1:***	*InternationalText*	Line number or name, as is known publicly.
	OperatorRef	0:1	*→Operator*	Operator-ID.
	RouteDescription	0:*	*InternationalText*	Description of route.
	Via	0:*	*+ServiceViaPoint*	Important stops on the route.
	Attribute	0:*	*+GeneralAttribute*	Information and attributes (with classifications) about the journey.
ServiceOrigin	*OriginStopPointRef*	0:1	*→StopPoint*	ID of the first stopping point of the journey; starting stop.
	OriginText	0:*	*InternationalText*	Name of the first stopping point of the journey; starting stop.
ServiceDestination	*DestinationStopPointRef*	0:1	*→StopPoint*	ID of the last stopping point of the journey; last
	DestinationText	1:*	*InternationalText*	Name of the last stopping point of the journey; last stop or destination.
	SituationFullRef	0:*	*+SituationFullRef*	Reference to an error message. This message can be found in the context of the response (ResponseContext) or made known through other channels.

Table 1: Example (from section 7.6) for the tabular notation of an XML structure

3.2.1 Grouping

The first column contains an identifier, which organises the elements into meaningful groups, e.g. Service or ServiceOrigin. This is purely for documentation purposes and in most cases corresponds to the name of an XML group which was used in the XML schema. The only purpose of using groups is to organise the elements and thus ensure more clarity and better reusability.

3.2.2 Element name

Element names are written in cursive in the second column, e.g. OperatingDay. If an element is mandatory, it is written in **bold**. Optional elements are not written in bold. The name of the structure itself is stated on the top left side in the table.

Elements, which are derived by extension or used anonymously, have three colons ":::" in the name field for the sake of identification (see Table 1).

3.2.3 Multiplicity & choice (Min:Max)

The conditions, whether an element is mandatory or optional, or whether it can appears once or multiple times in the parent element, are stated in the third column Min:Max. Here, the usual UML conventions "min:max" are used' e.g. "0:1" stands for an optional single element, "1:1" indicates a mandatory single element, "0:*" stands for an optional, multiple element or mandatory elements are written in **bold**.

In some cases, an element has to be selected from its set (XML choice). It is symbolised using a prefixed minus, e.g. "-1:1". In this case, another lowercase letter precedes the element name which shows a list of choices. A zero in the min-value indicates optional choices: "-0:1".

3.2.4 Data type

The data types are stated in cursive in the fourth column, e.g. *InternationalText*. If the namespace is different from TRIAS namespace, it is also stated, e.g. "xs:dateTime" or "siri:PtSituationElement".

- A complex data type, which contains structures as child elements, is marked in the column data type with "+*Structure*".
- Wherever elements are used as references (foreign key) to other objects, the type of the referenced objects with prefixed arrow is used as data type. For example, "->*StopPoint*" as type of a reference (*StopPointRefStructure*) to an object of type "*StopPointType*".
- In most places, enumerated types are given directly with the usable values, e.g. "walk | cycle". Only in some cases, a type is declared and referenced with extensive enumerations which are reused in several places.
- Abbreviations are used when entering data types to save space, e.g. "Structure" and "*Type*" at the endings are directly omitted. For example, instead of "*InternationalTextStructure*" is always entered as "*InternationalText*" as the data type.

3.2.5 Description

The intended use of all elements is explained in the last column. Other passages in the text are referred to in many places, for example, in case of complex child elements in a place where their table description can be found. In some places, the explanation is too elaborate and would go beyond the limits of table form. In this case, notes can be found in the text below the tables.

4 Nachrichtenübermittlung

In diesem Kapitel wird erläutert, wie TRIAS-Nachrichten ausgetauscht werden. Es kommen zwei grundlegende Verfahren zum Einsatz

- Anfrage mit synchroner Antwort (Request-Response-Verfahren),
- Abonnements mit asynchronen Nachrichten (Publish-Subscribe-Verfahren).

Diese Verfahren sind bereits etabliert und im Einsatz, z. B. in den SIRI-Schnittstellen.

4.1 Einsatz der SIRI-Verfahren

In SIRI wurden die eingangs aufgezählten Nachrichtenübermittlungsverfahren bereits definiert und beschrieben, vgl. (CEN, TS 15531 Part 2, 2011). Daher werden diese Verfahren hier aufgegriffen. Das hat zum einen den Vorteil, dass bereits getestete Verfahren verwendet werden können, zum anderen kann bei der Implementierung der TRIAS-Dienste evtl. auf eine bereits vorhandene SIRI-Implementierung zurückgegriffen werden, was Kosten und Zeit sparen kann.

Das grundlegende Verfahren ist die Anfrage mit synchroner Antwort. Ein Client stellt eine Anfrage an einen Server, der unmittelbar antwortet. In der SIRI-Terminologie ist der Anfrager der *Data Consumer*, der antwortende Server wird mit *Data Producer* bezeichnet (vgl. Abbildung 2).

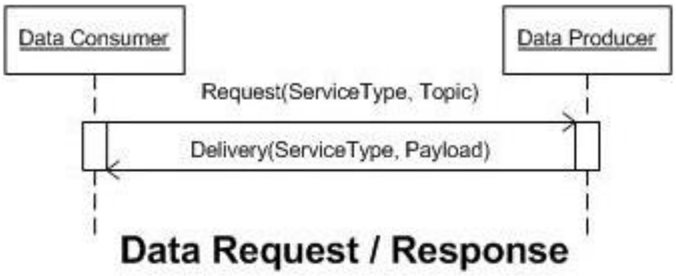

Abbildung 2: Anfrage mit synchroner Antwort (Abbildung entnommen aus SIRI, (CEN, TS 15531 Part 2, 2011)).

Anfragen mit synchroner Antwort werden bei fast allen TRIAS-Diensten verwendet (eine Ausnahme ist nur der Benachrichtigungsdienst). Die Rolle des Anfragers übernimmt z. B. das Portalsystem, das Anfragen an die EKAP richtet. Aber auch die mobile App stellt Anfragen an das Fahrzeug oder EKAP-Komponenten stellen untereinander Anfragen.

Etwas komplizierter ist der Abonnement-Mechanismus. Ein Datenkonsument interessiert sich für neue Nachrichten, weiß aber nicht, wann diese auftreten werden. Statt regelmäßig nachzufragen und so eine Grundlast zu erzeugen (und zu riskieren, dass er von der neuen Nachricht erst erfährt, wenn er das nächste Mal nachfragt), kann er ein Abonnement einrichten.

Abbildung 3 zeigt die grundlegenden Zusammenhänge. Der Datenkonsument hat zwei Rollen zu erfüllen, die des Abonnenten (Subscriber) und die des Empfängers von Nachrichten (Notification Consumer). Der Datenkonsument bittet den Server um die Einrichtung eines Abonnements (Subscription Request). Dabei teilt er dem Server mit, bei welcher Art Ereignisse er infomiert werden möchte. Der Server richtet das Abonnement ein, indem er es beim Abo-Verwalter (Subscription Manager) registriert. Danach geschieht erst etwas, wenn ein Ereignis eintritt, das dem Konsumenten zu melden ist. In diesem Fall schickt der Server als Benachrichtigungsersteller (Notification Producer) dem Datenkonsumenten die Nachricht mit dem neuen Ereignis (Delivery).

Dies wiederholt sich so lange, bis das Abonnement ausläuft oder vom Datenkonsumenten beendet wird.

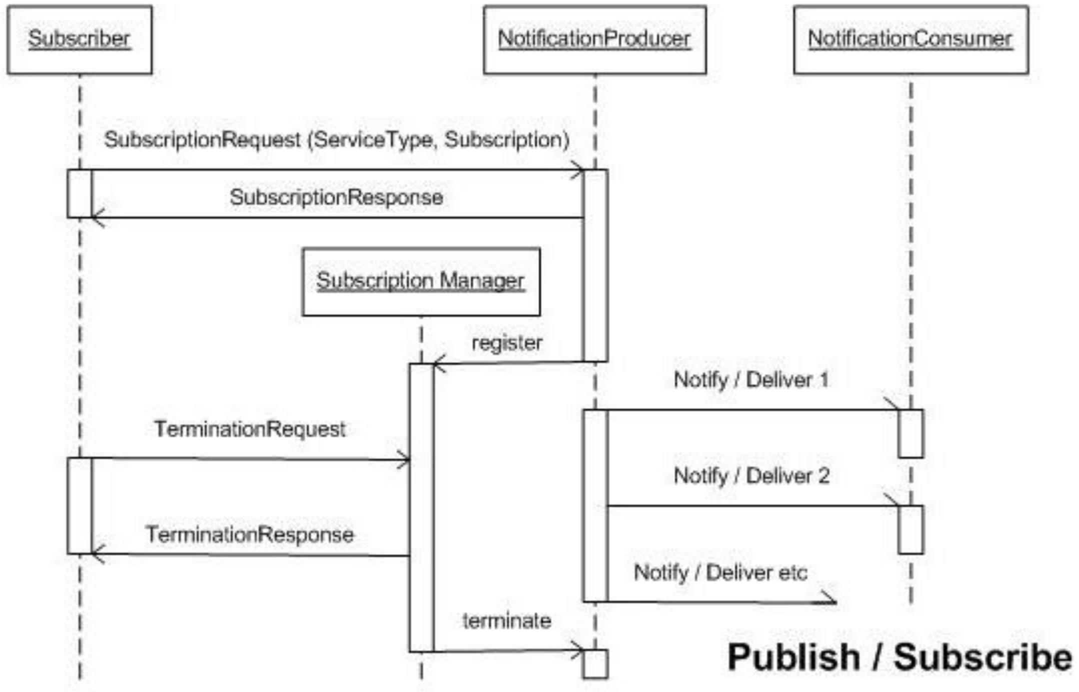

Abbildung 3: Abonnement-Verfahren mit asynchronen Benachrichtigungen (Abbildung entnommen aus SIRI, (CEN, TS 15531 Part 2, 2011)).

Sowohl Client als auch Server haben zwei Rollen zu erfüllen, nämlich der Client die Rolle des Subscribers und des Notification Consumers, der Server die Rolle des Notification Producers und des Subscription Managers. In den meisten Implementierungen wird dies aber nicht unterschieden und jeweils eine einzige Softwarekomponente erfüllt beide Rollen.

Das Abo-Verfahren wird komplettiert durch weitere Anfragen. Die Statusanfrage erlaubt es, den Status des Schnittstellenpartners abzufragen und dessen Verfügbarkeit zu testen. Die Heartbeat-Anfrage, die ein Server regelmäßig sendet, ermöglicht es dem Datenkonsumenten umgekehrt zu erkennen, wann ein Server verfügbar ist und Signale (Ping oder Heartbeat) aussendet. Details dazu finden sich in SIRI (CEN, TS 15531 Part 2, 2011), Kapitel 5.

In den TRIAS-Diensten kommt der Abo-Mechanismus beim Benachrichtigungsdienst vor, über den sich ein Datenkonsument (z. B. Portalsystem, mobile App, Fahrgastinformationssystem, etc.) über Störungen oder andere Ereignisse und Vorkommnisse informieren lassen will.

4.2 HTTP

Die Umsetzung der SIRI-Nachrichtenverfahren geschieht in TRIAS mit Hilfe von HTTP/1.1 (Hypertext Transfer Protocol[3]) als Transportprotokoll und XML (Extensible Markup Language[4]) für die Nachrichteninhalte.

[3] http://tools.ietf.org/html/rfc2616

Eine HTTP-Anfrage wird vom Server unmittelbar unter Nutzung des schon geöffneten IP-Ports beantwortet. Zum Beispiel sendet ein Client eine Anfrage nach einer Verbindungsauskunft als HTTP-Anfrage mit dem XML-Element *Trias* und *TripRequest* als einem der Kindelemente im POST-Block. Der Server antwortet synchron in der HTTP-Antwort mit dem XML-Element *Trias* und *TripResponse* als einem der Kindelemente.

Falls mehrere Anfragen in schneller Folge abgesendet werden, kann der HTTP-Mechanismus „Keep-Alive" zum Einsatz kommen, bei dem der bereits geöffnete Port eine Zeit lang leben bleibt und wiederbenutzt werden kann, um häufiges Öffnen und Schließen des Ports zu sparen.

Für größere Nachrichten empfiehlt sich der Einsatz eines Komprimierungsverfahrens. Solche Methoden sind ebenfalls für HTTP spezifiziert.

4.3 Nachrichten-Kodierung

Zur Nachrichten-Kodierung ist grundsätzlich "UTF-8" zu verwenden. Dies gilt sowohl für die Header, das XML-Element und den Inhalt von Anfragen und Antworten. Dies gewährleistet eine problemlose Nutzung auch bei Verwendung von Sprachen mit nicht-lateinischen Zeichen. Zusätzlich müssen sich Clients, die mit mehreren TRIAS-Schnittstellen verschiedener Anbieter kommunizieren, nicht um eine unterschiedliche, Anbieter-spezifische Konvertierung der empfangenen Daten kümmern.

4.4 Rollen von Server und Client

Bei der Nutzung des synchronen Anfrage-Antwort-Verfahrens ist der Datenkonsument (der Anfrager) ein HTTP-Client, der Datenproduzent (der antwortende Server) ein HTTP-Server.

Lediglich beim Benachrichtigungsdienst, wenn das Abonnement-Verfahren zum Einsatz kommt, ist die Lage komplizierter. Hier müssen sowohl Datenkonsument als auch Datenproduzent die Rollen von Client und Server im HTTP-Sinne beide ausfüllen. Wenn der Datenproduzent (Notification Producer) eine neue Nachricht an den Datenkonsumenten senden will, wird er zum Client im HTTP-Sinn und der Datenkonsument zum Server im HTTP-Sinn.

4 Messaging

The exchange of TRIAS messages is explained in this chapter. Two basic procedures are used:

- Request with synchronous response (request response procedure),
- Subscriptions with asynchronous messages (publish subscribe procedure).

These procedures are already established and in use, for example in the SIRI interfaces.

4.1 Use of SIRI procedures

In SIRI, the message exchange procedure mentioned above has already been defined and described, see (CEN, TS 15531 part 2, 2011). Therefore, these procedures are continued here. On the one hand, it has the advantage that already tested procedures can be used and on the other

[4] http://www.w3.org/XML/

an already existing SIRI implementation can be used when implementing TRIAS services, something that can save time and cost.

The basic procedure is request with synchronous response. A client sends a request to a server which answers immediately. In SIRI terminology, the requester is the Data Consumer, the responding server is called the Data Producer (see Figure 2).

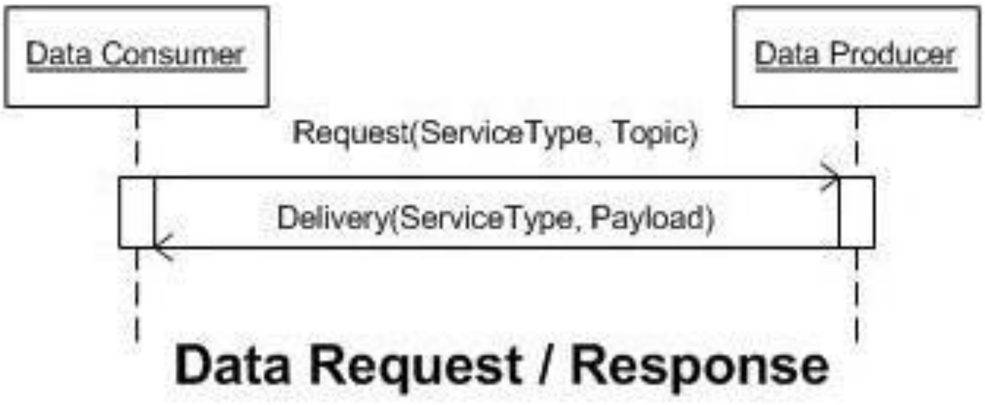

Figure 2: Request with synchronous response (figure obtained from SIRI, (CEN, TS 15531 part 2, 2011))

Requests with synchronous response are used in almost all the TRIAS services (the only exception is the notification service). The role of the requester is undertaken, for example, by the portal system which sends requests to EKAP. But the mobile app also sends requests to the vehicle or EKAP components send requests to one another.

The subscription mechanism is somewhat more complicated. The data consumer is interested in new messages but does not know when they will appear. Instead of sending requests regularly and thus creating a basic load (and risking that learning about the new message only as a result of the next request), the consumer can set up a subscription.

Figure 3 shows the basic connections. The data consumer has two roles to fulfil, that of the subscriber and that of the notification consumer. The data consumer requests the server to set up a subscription (subscription request). In the process, the consumer informs the server about the type of events it would like to be informed. The server sets up the subscription by registering him with the subscription manager. After this, the consumer must be notified of the event when it occurs. In this case, the server sends messages with the new event (delivery) to the data consumer as a notification producer. This is repeated until the subscription runs out or is ended by the data consumer.

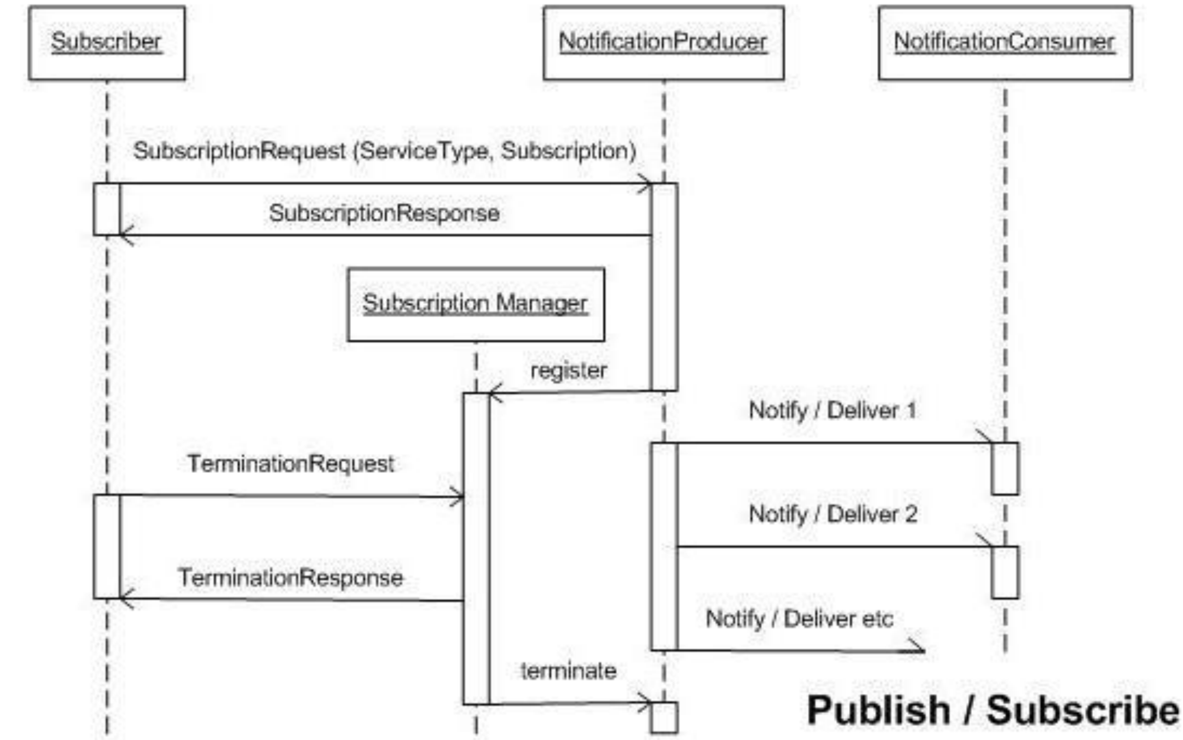

Figure 3: Subscription with asynchronous notifications (figure obtained from SIRI, (CEN, TS 15531 part 2, 2011))

The client as well as the server have two roles to fulfil, namely the client has the role of the subscriber and notification consumer and the server has the role of notification producer and subscription manager. In most implementations, they are not different and a single software component fulfils both roles.

The subscription procedure is completed with additional requests. The status request allows the status of the interface partner to be requested and its availability to be tested. The heartbeat request, which is sent by a server regularly, makes it possible for the data consumer in turn to detect the availability of a server and sending of signals (ping or heartbeat). Details regarding this are available in SIRI (CEN, TS 15531 part 2, 2011), chapter 5.

The subscription mechanism during the notification service occurs in the TRIAS services, by means of which a data consumer (e.g. portal system, mobile app, passenger information system etc.) wants to inform of faults or other events and occurrences.

4.2 HTTP

The SIRI message procedure is implemented in TRIAS with the help of HTTP/1.1 (Hypertext Transfer Protocol[5]) as the transport protocol and XML (Extensible Markup Language[6]) for message contents

An HTTP request is answered by the server by directly using the IP port already opened. For example, a client sends a request for a planned journey as HTTP request with the XML element

[5] http://tools.ietf.org/html/rfc2616

[6] http://www.w3.org/XML/

Trias and TripRequest as a child element in the POST block. The server answers synchronously in the HTTP response with XML element Trias and TripResponse as one of the child elements.

If multiple requests are sent in rapid succession, the HTTP mechanism “keep alive” can be used, in which the already opened port remains open for a long time and can be reused to prevent frequent opening and closing of the port.

For larger messages, it is recommended to use a compression procedure. Such methods are also specified for HTTP.

4.3 Message coding

In principle, “UTF-8” must be used for message coding. This applies to the header, the XML element and content of requests and responses. This ensures problem-free use even when using languages with non-Latin characters. In addition, the clients that communicate with several TRIAS interfaces of different providers, must not have to arrange for provider-specific conversion of the received data.

4.4 Roles of server and client

Using the synchronous request-response-procedure, a data consumer (the requester) is an HTTP client, the data provider (the responding server) an HTTP server.

The situation is more complicated only for the notification service if the subscription procedure is used. In this case, the data consumer and the data provider must both fulfil the roles of client and server according to HTTP. If the data provider (notification producer) wants to send a new message to the data consumers, it becomes a client according to HTTP and a data consumer to the server according to HTTP.

5 Identifikation von Objekten über Systemgrenzen hinweg

Damit verschiedene Systeme dasselbe Objekt referenzieren können, ist eine Objekt-ID notwendig, die allen Systemen bekannt ist. Im Rahmen der TRIAS-Schnittstellen sind Haltestellen, Linien und Verkehrsunternehmen Beispiele für solche Objekttypen, zu denen Informationen über die Schnittstellendienste ausgetauscht werden. Daher braucht man für sie (und weitere Objekttypen) Referenzierungssysteme, die allgemein bekannt sind und verwendet werden können.

Das bedeutet nicht notwendigerweise, dass ein Softwaresystem diese Objektschlüssel auch selbst im Betrieb verwenden muss. Es genügt, wenn es die allgemeinen Objektreferenzen verstehen und auf die intern verwendeten Identifikatoren abbilden kann.

Für die in diesem Kapitel vorgestellten Schemata zur Objektreferenzierung wird eine an IFOPT angelehnte Syntax verwendet. Sie benutzt den Doppelpunkt zur Abgrenzung von Namensräumen. Aus diesem Grund ist ein Doppelpunkt ein syntaktisches Trennzeichen und darf in Identifikatoren nicht verwendet werden.

In den folgenden Abschnitten wird für verschiedene Objekttypen vorgestellt, welche Referenzierungssysteme verwendet werden sollen.

5.1 Haltestellen und Haltepunkte

Für die Referenzierung von Haltestellen und Haltepunkten gibt es von CEN die europäische Norm IFOPT (CEN, EN 28701:2012, 2012). Dort ist in Kapitel 6.8.1 eine Syntax für den Aufbau eines Referenzierungsschlüssels vorgesehen. Einige Systeme in Deutschland unterstützen diese Syntax bereits. Eine bundesweite Einführung wird im BMVBS-Projekt DELFIplus vorbereitet. In den TRIAS-Schnittstellen sollen die Ergebnisse aus diesem Projekt zur Anwendung kommen.

Aufbau eines IFOPT-Objektschlüssels
Länderkürzel:Region:Haltestellennummer:Bereich:Haltepunkt

Das folgende Beispiel zeigt den (hierarchischen) Aufbau der Schlüssel für eine Haltestelle, einen Haltestellenbereich und einen Haltepunkt

Praxis-Beispiel: Haltestelle Karlsplatz (Stachus) in München:

Haltestellenobjekt	Eindeutige ID
Haltestelle Karlsplatz (Stachus) in München	de:9162:1
Haltestellenbereich U-Bahn U4/5	de:9162:1:2
Haltepunkt U4/5 Richtung Odeonsplatz	de:9162:1:2:URiOd

Client-Systeme, die selbst keine eigene Datenversorgung haben, können die Objektreferenzen für Haltestellen und Haltepunkte mit Hilfe des TRIAS-Schnittstellendiensts Ortsauflösung (vgl. 8.1) von der EKAP beziehen.

Basierend auf der Norm IFOPT wurde in Deutschland die Deutschlandweite Haltestellen ID entwickelt. Das Format ist in der VDV-Schrift 432 (VDV-Schrift 432, 07/2016) beschrieben.

5.2 Orte und Gemeinden

Zur eindeutigen Referenzierung von Gemeinden existiert in Deutschland der Amtliche Gemeindeschlüssel (AGS[7]), früher auch Gemeindekennziffer (GKZ) genannt. Für die Orte innerhalb einer Gemeinde ist die Situation je nach Bundesland unterschiedlich. In Bayern z. B. gibt es je Gemeinde eine Liste von Orten mit amtlich festgelegten Orts-IDs. Wo diese Identifikatoren fehlen, müssen eigene Festlegungen getroffen werden, damit systemübergreifend ein gleiches Verständnis von Orten vorliegt, so dies notwendig ist. Dabei kann z. B. die Ortsliste aus dem Bestand der DELFI-Meta-Daten verwendet werden.

Für den Betrieb von TRIAS-Schnittstellen empfiehlt sich die Verwendung eines Ortsschlüssels, der sich vom Aufbau her an die IFOPT-Norm für Haltestellen anlehnt:

Aufbau eines Ortschlüssels
Länderkürzel:AGS:Ort

Beispiel	
Ilmenau	de:16070029:1

Client-Systeme, die selbst keine eigene Datenversorgung haben, können die Objektreferenzen für Gemeinden und Orte mit Hilfe des TRIAS-Schnittstellendiensts Ortsauflösung von der EKAP beziehen.

5.3 Adressen und POIs

Für den Betrieb von TRIAS-Schnittstellen ist es nicht notwendig, dass Adressen und wichtige Punkte (Points Of Interest, POI) systemübergreifend referenziert werden können. Es genügt, deren Lage durch Koordinatenpositionen mitzuteilen.

Die Kategorisierung von POI basiert auf dem Tagging-Schema von OpenStreetMap[8]. Ein POI kann mehrere Schlüssel-Wert-Paare zugeordnet bekommen (z.B. für eine Fahrrad-Ladestation amenity=charging_station und bike=yes).

[7] Siehe auch: Statistisches Bundesamt, https://www.destatis.de/DE/ZahlenFakten/LaenderRegionen/Regionales/Gemeindeverzeichnis/Gemeindeverzeichnis.html

[8] http://wiki.openstreetmap.org/wiki/DE:Map_Features

5.4 Organisationen

Zur eindeutigen Referenzierung von Organisationen (z. B. Verkehrsunternehmen und Verkehrsverbünden, Aufgabenträger, etc.) wird ein Organisationscode verwendet. Damit diese Codes über mehrere Systeme hinweg eindeutig bleiben, empfiehlt sich der Aufbau einer übergreifenden Datenbank von Organisationen. Damit die Organisations-IDs über mehrere Länder hinweg eindeutig bleiben, wird dem Organisationscode ein Länderkürzel als Namensraum vorangestellt.

Aufbau einer Organisations-ID
Länderkürzel:Organisationscode

Beispiele	
Verkehrs- und Tarifverbund Stuttgart	de:vvs
Stuttgarter Straßenbahn AG	de:ssb
Fernverkehr Deutsche Bahn	de:dbag
DB Regio Baden-Württemberg	de:dbregiobw

5.5 Linien und Linienrichtungen

Zur eindeutigen Referenzierung von Linien wird der Linienschlüssel des verantwortlichen Datenlieferanten verwendet. Als verantwortlicher Datenlieferant kommt das beauftragte Verkehrsunternehmen (Konzessionär) oder der zuständige Verkehrsverbund in Frage.[9] Damit die Linien-IDs über mehrere Datenlieferanten hinweg eindeutig bleiben, wird dem Linienschlüssel die Organisations-ID (vgl. 5.4) als Namensraum vorangestellt.

Aufbau einer Linien-ID
Länderkürzel:Organisationscode:Linienschlüssel

Beispiel	
Stadtbahn-Linie U1 in Stuttgart	de:vvs:20001

Ist es nicht möglich einer Fahrt eine Linien-ID zuzuordnen, so muss der Wert „NO_LINE“ eingetragen werden.

[9] Damit in Datensammelsystemen Dopplungen von Datenlieferungen zu einer Linie von mehreren Datenlieferanten vermieden werden können, empfiehlt sich der Aufbau einer übergreifenden (im besten Falle nationalen) Datenbank von Linien.

Zur eindeutigen Referenzierung von Linienrichtungen wird der Richtungscode des verantwortlichen Datenlieferanten verwendet. Der Richtungscode ist vom Datenlieferanten frei wählbar und wird für den Fahrgast erst durch begleitende Texte verständlich. Die Richtungs-ID wird nur im Kontext einer Linie verwendet, so dass das Voranstellen des Linienschlüssels als Namensraum nicht notwendig ist.

Aufbau einer Richtungs-ID
Richtungscode

Beispiele	
Hin	H
Rück	R
Hin	1
Rück	2
Stadteinwärts	E
Stadtauswärts	A

Ist es nicht möglich einer Fahrt eine Linienrichtungs-ID zuzuordnen, so muss der Wert „NO_DIRECTION“ eingetragen werden.

5.6 Fahrten

Zur eindeutigen Referenzierung von Fahrten (engl. *Vehicle journey* oder kurz: *Journey*) wird der Fahrtenschlüssel des verantwortlichen Datenlieferanten verwendet. Damit die Fahrt-IDs über mehrere Datenlieferanten hinweg eindeutig bleiben, wird dem Fahrtenschlüssel die Organisations-ID (vgl. 5.4) als Namensraum vorangestellt.

Der Fahrtenschlüssel ist vom Datenlieferanten frei wählbar, solange er im Namensraum einer Linie eindeutig ist.

Aufbau einer Fahrt-ID
Länderkürzel:Organisationscode:Linienschlüssel:Fahrtenschlüssel

Falls eine Organisation (z. B. Verkehrsunternehmen) ihre Fahrten nicht in Linien organisiert (z. B. Bahnfernverkehr), kann der Linienschlüssel leer bleiben.

Beispiele	
Fahrt 1512 der Linie U1 in Stuttgart	de:vvs:20001:1512
ICE 612 der DB AG	de:dbag::612

5.7 Fahrzeuge

Zur eindeutigen Referenzierung von Fahrzeugen (engl. *Vehicle*) wird der Fahrzeugcode des verantwortlichen Datenlieferanten verwendet. Damit die Fahrzeug-IDs über mehrere Datenlieferanten hinweg eindeutig bleiben, wird dem Fahrzeugcode die Organisations-ID (vgl. 5.4) als Namensraum vorangestellt.

In Echtzeitschnittstellen (VDV 454, SIRI ET) teilen Leitstellen zu einer Fahrplanfahrt die Fahrzeug-ID mit, so dass eine EKAP für einen bestimmten Betriebstag wissen kann, welche Fahrt von welchem Fahrzeug durchgeführt wird. Für jeden Betriebstag muss daher die Zuordnung von Fahrzeug-ID zur Fahrt-ID eindeutig sein.

Aufbau einer Fahrzeug-ID
Länderkürzel:Organisationscode:Fahrzeugcode

Beispiel	
Fahrzeug 5812 der SSB AG	de:ssb:5812

5.8 Fahrzeugtypen

Der Fahrzeugtyp (engl. *Vehicle type*) und die damit verbundenen Fahrzeugausstattungsattribute werden vom Fahrzeug an die mobile App mitgeteilt (in Form von Code und menschenlesbaren Text). Der Fahrzeugtypcode wird nicht in Folgeaufrufen verwendet und wird daher im Rahmen von TRIAS nicht weiter betrachtet.

5.9 Betriebstage

Eine Fahrplanfahrt wird erst in Verbindung mit einem Betriebstag (engl. *Operating Day*) zu einer spezifischen Fahrt. Ein Betriebstag kann auch Uhrzeiten nach Mitternacht einschließen und daher von einem Kalendertag abweichen. Ob eine solche Abweichung existiert und wie groß sie ist, ist für die Fahrgastinformation nicht relevant. Den Fahrgästen gegenüber werden nur Uhrzeiten und Datumsangaben nach dem Kalendertagsprinzip bekannt gegeben.

Ein Betriebstag ist in TRIAS die Referenz auf den Betriebstag-Code der Fahrplandaten.

Diese Betriebstag-Codes sollten in TRIAS nach der Norm ISO 8601 dargestellt werden.

Beispiel	
29. März 2013	2013-03-29

5.10 Eigentümer

Mit dem Begriff Eigentümer (engl. *Owner*) sind hier die Betreiber von Haltestelleneinrichtungen und Fahrgastinformationsgeräten gemeint. In der Regel sind das Verkehrsunternehmen, aber auch z. B. Kinobetreiber können einen Monitor für die Anzeige von aktuellen Haltestellenabfahrten aufstellen und betreiben. Die Referenzierung von Eigentümern erfolgt auf genau dieselbe Weise wie die von Verkehrsunternehmen und - verbünden (vgl. 5.4).

5.11 Haltestellen- und Fahrzeugeinrichtungen

Haltestellen- und Fahrzeugeinrichtungen (wie z. B. Aufzüge oder Fahrscheinautomaten) werden durch Codes referenziert, die vom Eigentümer (vgl. 5.10) vergeben werden. Im Kontext eines Eigentümers ist der Code einer Einrichtung also global eindeutig.

5.12 Teilnehmende Systeme / IT-Systeme

Die TRIAS-Dienste werden von IT-Systemen angeboten und in Anspruch genommen. Sie sind die teilnehmenden Systeme (engl. *Participants*) an einem umfassenden Systemverbund zur Steuerung des Betriebs des ÖV und zur Fahrgastinformation. Damit diese Systeme unterscheidbar und ansprechbar sind, benötigen sie Kennungen (in VDV 453/454 als *Leitstellenkennung* bekannt).

Aufbau einer Systemkennung
Länderkürzel:Organisationscode:Systemkennung

Beispiel	
Öffentliche EKAP des VVS	de:vvs:publicEKAP

5.13 Ereignismeldungen

Ereignis- und Störungsmeldungen (engl. *Situations*) werden mit Hilfe der in SIRI SX definierten Strukturen übertragen. Dort ist auch die Vergabe von IDs für die Ereignismeldungen geregelt. Die Meldungs-IDs werden im Kontext des teilnehmenden Systems (vgl. 5.12) übertragen und sind somit global eindeutig.

5.14 Tarifverantwortliche

Eine Organisation, die verantwortlich ist für die Festlegung von Tarifstrukturen und die Entwicklung von Fahrscheinprodukten, wird als Tarifverantwortlicher (engl. *Fares authority*) bezeichnet. Für Verbundtarife sind dies meist die Verkehrs- und Tarifverbünde, für Haustarife die Verkehrsunternehmen selbst. Die Referenzierung von Tarifverantwortlichen erfolgt auf genau dieselbe Weise wie die von Verkehrsunternehmen und - verbünden (vgl. 5.4).

5.15 Tarifzonen

Die Codierung von Tarifzonen (engl. *Fare zones*) liegt in der Obhut der jeweiligen Tarifverantwortlichen (vgl. 5.14). Tarifzonen werden im Kontext der jeweiligen Tarifverantwortlichen angegeben und werden so global eindeutig.

5.16 Fahrscheine und Vielfahrerkarten

Die Codierung von Fahrscheinen (engl. *Ticket*) liegt in der Obhut der jeweiligen Tarifverantwortlichen (vgl. 5.14). IDs von Fahrscheinen werden im Namensraum der jeweiligen Tarifverantwortlichen angegeben und sind so global eindeutig.

Aufbau eines Codes für einen Fahrschein
Länderkürzel:Organisationscode:FahrscheinCode

Beispiel	
Einzel-Ticket für Erwachsene im VVS-Gebiet für 2 Zonen	de:vvs:EinzelErw2Z

Die Codierung von Vielfahrerkarten (engl. *TravellerCard*), z. B. BahnCard50 der Deutschen Bahn AG, liegt in der Obhut der jeweiligen Tarifverantwortlichen (vgl. 5.14). Der Code einer Vielfahrerkarte muss im Namensraum des Tarifverantwortlichens angegeben werden.

Aufbau eines Codes für eine Vielfahrerkarte
Länderkürzel:Organisationscode:TravellerCardCode

Beispiel	
BahnCard50 der DB AG	de:dbag:BC50

5 Identification of objects beyond system borders

In order that different systems are able to reference the same object, an object ID is necessary which is known to all the systems. Within the framework of TRIAS interfaces, stops, lines and transport companies are examples of such object types, for which information is exchanged via the interface services. Hence, referencing systems which are commonly known and usable are required for them (and other object types).

This does not necessarily mean that a software system must use this object key during operation. It suffices, when the system understands the general object references and is able to depict the relevant information using internal identifiers.

An IFOPT-oriented syntax is used for the schemes to reference objects presented in this chapter. It uses a colon to separate namespaces. For this reason, a colon is a syntactic separator and cannot be used in identifiers.

In the following sections, which reference systems should be used for different object types, is presented.

5.1 Stops and stopping points

For referencing stops and stopping points there is a European Standard IFOPT from CEN (CEN, EN 28701:2012, 2012). In section 6.8.1, a syntax for the structure of a referencing key is described. Several systems in Germany already support this syntax. Nationwide introduction is being prepared in the BMVBS Project DELFIplus. The results of this project should be used in the TRIAS interfaces.

Structure of an IFOPT object key
CountryCode:Region:StopNumber:StopArea:StoppingPoint

The following example shows the (hierarchical) structure of keys for a stop, stop area and a stopping point. Practical example: Stop Karlsplatz (Stachus) in Munich.

Stop object	Unique ID
Stop Karlsplatz (Stachus) in Munich	**de:9162:1**
Stop area underground U4/5	**de:9162:1:2**
Stopping point U4/5 direction Odeonsplatz	**de:9162:1:2:URiOd**

Client systems that do not have their own data supply can obtain object references for stops and stopping points using the TRIAS interface service spatial resolution (see 8.1) from EKAP.

The Germany-wide stop ID has been developed in Germany on the basis of the standard IFOPT. The format is described in VDV Guideline 432 (VDV Guideline 432, 07/2016).

5.2 Localities and municipalities

There exists an official municipality code (AGS[10]) for referencing municipalities uniquely in Germany (previously also called as the district code (GKZ)). For localities within a district, the situation is different depending on the Federal State. For example, in Bavaria, there is a list of localities with official locality IDs for every district. Wherever these identifiers are missing, a few decisions have to be made so that there is system-wide and a common understanding of the localities which is required. In the process, for example, the list of localities can be used from the inventory of DELFI metadata.

For the operation of TRIAS interfaces, the use of a locality key is recommended which is structurally similar to the IFOPT standard for stops:

Structure of a locality key
CountryCode:AGS:Location

Example	
Ilmenau	**de:16070029:1**

Client systems that do not have their own data supply can obtain object references for localities and districts using the TRIAS interface service spatial resolution from EKAP.

5.3 Addresses and POIs

To operate TRIAS interfaces it is not necessary that addresses and points of interest (POI) be referenced across systems. It suffices to indicate their location using coordinate positions.

The categorisation of POI is based on the tagging schema of OpenStreetMap[11]. A POI can be assigned to several key-value-pairs (e.g. for a bicycle charging station amenity=charging_station and bike=yes).

5.4 Organisations

An organisation code is used for unique referencing of organisations (e.g. transport companies and transport networks, federal authorities). The structure of a comprehensive database of organisations is recommended so that this code remains unique across several systems. A country code is prefixed as namespace to the organisation code so that the organisation IDs remain unique across several countries.

Structure of an organisation ID
CountryCode:OrganisationID

[10] Also see: Federal Statistical Office, https://www.destatis.de/DE/ZahlenFakten/LaenderRegionen/Regionales/Gemeindeverzeichnis/Gemeindeverzeich nis.html

[11] http://wiki.openstreetmap.org/wiki/DE:Map_Features

Examples	
Traffic and tariff association of Stuttgart	**de:vvs**
Stuttgarter Straßenbahn AG	**de:ssb**
Fernverkehr Deutsche Bahn	**de:dbag**
DB Regio Baden-Württemberg	**de:dbregiobw**

5.5 Lines and line directions

The line key of the responsible data supplier is used for unique referencing of lines. The authorised transport company (concessionaire) or the competent transport association is considered as the data supplier[12]. The organisation ID (compare Section 5.4) is prefixed as namespace to the line ID so that the line IDs remain unique across several data suppliers.

Structure of a line ID
CountryCode:OrganisationID:LineID

Example	
City rail line U1 in Stuttgart	**de:vvs:20001**

If it is not possible to assign a line ID to a trip, the value "NO_LINE" must be entered.

The direction code of the responsible data supplier is used for unique referencing of line directions. The direction code can be freely selected by the data supplier and becomes comprehensible to the passenger only because of the accompanying texts.

The direction ID is only used in the context of a line so that it is not necessary to prefix the line key as namespace.

Structure of a direction ID
DirectionID

[12] In order to avoid duplication of data deliveries to a line from several data suppliers in the data collection systems, the structure of a comprehensive (national in best cases) database of lines is recommended.

Examples	
Outbound	**O**
Inbound	**I**
Outbound	**1**
Inbound	**2**
Towards the city	**T**
Away from the city	**A**

If it is not possible to assign a line direction ID to a trip, the value "NO_DIRECTION" must be entered.

5.6 Journeys

A journey key of the responsible data supplier is used for unique referencing of journeys. The organisation ID (see 5.4) is prefixed as namespace to the journey key so that the journey IDs remain unique across several data suppliers.

The journey key can be freely selected by the data supplier as long as it is unique in the namespace of a line.

Structure of a journey ID
CountryCode:OrganisationID:LineID:JourneyID

If an organisation (e.g. transport company) does not organise its journeys in lines (e.g. long-distance passenger rail), the line key can remain empty.

Examples	
Journey 1512 of line U1 in Stuttgart	**de:vvs:20001:1512**
ICE 612 of DB AG	**de:dbag::612**

5.7 Vehicles

A vehicle code of the responsible data supplier is used for unique referencing of vehicles. The organisation ID (see 5.4) is prefixed as namespace to the vehicle code so that the vehicle IDs remain unique across several data suppliers.

Control rooms for a journey inform the vehicle ID via real-time interfaces (VDV 454, SIRI ET) so that EKAP can know which journey is made from which vehicle for a certain operating day. Hence, allocation of vehicle ID to journey ID must be unique for every operating day.

Structure of a vehicle ID
CountryCode:OrganisationID:VehicleID

Example	
Vehicle 5812 of SSB AG	**de:ssb:5812**

5.8 Vehicle types

The vehicle type and the associated vehicle facility attributes are sent from the vehicle to the mobile app (in the form of code and human-readable text). The vehicle type code is not used in subsequent calls and is hence not taken into consideration within the scope of TRIAS.

5.9 Operating days

A journey becomes a specific journey only in conjunction with an operating day. An operating day can also include hours past midnight and therefore differ from a calendar day. Whether such a discrepancy exists and its extent is not relevant for passenger information. Only time and date details according to the calendar day principle are announced to the passengers.

An operating day in TRIAS is the reference to operating day code of journey data.This operating day should be displayed in TRIAS according to the ISO 8601 standard.

Example	
29 March 2013	**2013-03-29**

5.10 Owners

The term “owner” refers to operators of stop facilities and passenger information devices. Normally they are transport companies, but, for example, even movie theatre owners can set up and operate a monitor for displaying current departures. The owners are referenced in exactly the same way as the transport companies and associations (see 5.4).

5.11 Stop and vehicle facilities

Stop and vehicle facility (such as lifts or ticket machines) are referenced by codes that are assigned by owners (see 5.10). In the context of an owner, the equipment code is globally unique.

5.12 Participating systems / IT systems

The TRIAS services are provided and used by IT systems. They are the participating systems of a comprehensive network to control the operation of public transport and passenger information. To ensure that these systems are distinguishable and responsive, they require codes (called as control centre codes in VDV 453/454)

Structure of a system ID
CountryCode:OrganisationID:SystemID

Example	
Public EKAP of VVS	**de:vvs:publicEKAP**

5.13 Event messages

Event and error messages (situations) are transmitted with the help of the structures defined in SIRI SX. The allocation of IDs for event messages is also controlled there. The message IDs are transmitted in the context of the participating system (see 0) and are hence globally unique.

5.14 Fare authority

An organisation that is responsible for the assignment of fare structures and the development of ticket products is called a fare authority. It is mainly transport and fare associations that are responsible for associations’ fares and transport companies for in-house fares. The fare authorities are referenced in exactly the same way as the transport companies and associations (see 5.4).

5.15 Fare zones

The coding of fare zones is the responsibility of the respective fare authority (see 5.14). Fare zones are specified in the context of the respective fare authority and are therefore globally unique.

5.16 Tickets and traveller cards

The coding of tickets is the responsibility of the respective fare authority (see 5.14). IDs of tickets are specified in the namespace of the respective fare authority and are therefore globally unique.

Structure of a code for a ticket
CountryCode:OrganisationID:TicketID

Example	
Single ticket for adults in VVS area for 2 zones	**de:vvs:EinzelErw2Z**

The coding of traveller cards, e.g. BahnCard50 of Deutschen Bahn AG, is the responsibility of the respective fare authority (see 5.14). The code of a traveller card must be stated in the namespace of the fare authority.

Structure of a code for a traveller card
CountryCode:OrganisationID:TravellerCardCode

Example	
BahnCard50 of DB AG	**de:dbag:BC50**

6 Dienste, XML-Schemata und Konventionen

In diesem Dokument werden Schnittstellendefinitionen für Dienste zwischen Softwarekomponenten dargestellt. Für eine ausführliche Erläuterung der Aufgabenstellung dieser Dienste und der möglichen Systemarchitekturen sei hier auf die grundlegenden VDV-Schriften (VDV-Schrift 431-1, 2014) und (VDV-Schrift 430, 2014) verwiesen.

Die TRIAS-Schnittstellendienste sind als XML-Schemata definiert. Eine Übersicht über die Dienste und ihre Implementierung als XML-Schema bietet der erste Abschnitt dieses Kapitels. Einige Strukturdefinitionen sind in mehreren Diensten nützlich und werden daher in eigenen Schemadateien als gemeinsame Basis hierarchisch definiert, so dass eine Wiederverwendbarkeit ermöglicht wird. Das dabei verfolgte Konzept orientiert sich stark an den Grundsätzen der Objektorientierung. Die gemeinsam genutzten Strukturdefinitionen sind im zweiten Abschnitt beschrieben. Der dritte Abschnitt stellt die XML-Schemata vor, die aus SIRI importiert werden. Eine Klassifikation der Fehlerzustände findet sich im vierten Abschnitt.

6.1 Bereitgestellte Dienste

Die TRIAS-Schnittstellenfamilie umfasst derzeit folgende Dienste:

Dienst	Bezeichnung des Anfrageelements	Schema-Datei	Kapitel
Ortsinformation	LocationInformationRequest	Trias_Locations.xsd	8
Verbindungsauskunft	TripRequest	Trias_Trips.xsd	8
Abfahrtstafeln	StopEventRequest	Trias_StopEvents.xsd	10
Logische Ortung	PositioningRequest	Trias_Positioning.xsd	11
Fahrtinformation (EKAP)	TripInfoRequest	Trias_TripInfo.xsd	12
Anschlussmeldung	ConnectionDemandRequest	Trias_Connections.xsd	13.1.1
Anschlussstatus	ConnectionStatusRequest	Trias_Connections.xsd	13.1.2
Info bei Anschlussverlust	ConnectionStatusResponse	Trias_Connections.xsd	13.1.3
Anschlussrückmeldung	ConnectionReportRequest	Trias_Connections.xsd	13.1.4
Fahrpreis- und Tarifberechnung	FaresRequest	Trias_Fares.xsd	14.2.11 4
Buchungsinformation	BookingInfoRequest	Trias_Booking.xsd	16
IV-Routing	IndividualRouteRequest	Trias_IndividualTrips.xsd	17
Kartendienst	MapServiceRequest ImageCoordinatesRequest GeoCoordinatesRequest	Trias_Maps.xsd	18
Schadensmeldung	FacilityStatusReport	Trias_Facilities.xsd	19

Dienst	Bezeichnung des Anfrageelements	Schema-Datei	Kapitel
Zustand von Einrichtungen	FacilityRequest	Trias_Facilities.xsd	19.4
Benachrichtigungsdienst	SubscriptionRequest	Trias.xsd	7.1.2 + 20
Personalisierungsdienst	PersonalisationRequest	Trias_Personalisation.xsd	21
Fahrzeuginformation	VehicleDataRequest	Trias_VehicleInterface.xsd	22
Fahrzeuginteraktion	VehicleInteractionRequest	Trias_VehicleInterface.xsd	23
Diensteregister	ServiceRegisterRequest	Trias_ServiceRegister.xsd	24
Authentifizierung	AbstractTriasServiceRequest (vererbt auf alle TRIAS-Nachrichten)	Trias_RequestSupport.xsd	7.9 + 25

Tabelle 2: Liste der TRIAS-Dienste und ihrer Anfrageelemente.

6.2 Dienstübergreifend genutzte XML-Schemata

Um Strukturen, die in mehr als einem Dienst verwendet werden, nicht mehrfach und damit redundant definieren zu müssen, werden gemeinsam benutzte Basis-XML-Schemata eingeführt, die sich hierarchisch inkludieren. Die Inklusionsreihenfolge und der Zuschnitt der Schemadateien sind dabei so gewählt, dass inhaltlich verwandte Elemente in einer Datei zusammenstehen und dass jedes Schema möglichst nur so viel inkludiert, wie für die eigenen Aufgaben notwendig ist.

Die gemeinsam genutzten Basis-Schemadateien werden ausführlich in Kapitel 7 erläutert.

6.3 Importierte Siri-Schemata

Aus der SIRI-Schnittstellen-Spezifikation der Version 1.4 werden die Schemadateien

- siri.xsd
- siri_facilities-v1.2.xsd
- siri_situation-v1.1.xsd
- siri_requests-v1.3.xsd
- siri_common-v1.4.xsd
- siri_situationExchange_service.xsd
- siri_facilityMonitoring_service.xsd

nach TRIAS importiert.

Durch diesen Import von SIRI-Definitionen wird erreicht, dass die SIRI-Verfahren für den Austausch von Nachrichten auch für die TRIAS-Meldungen anwendbar sind. Außerdem können bestimmte Strukturdefinitionen aus SIRI wiederverwendet werden, was die Konsistenz zwischen diesen Schnittstellenstandards sicherstellt. Dies betrifft unter anderem die Definition von Verkehrsmittelarten (modes), Störungsereignissen (situations) und Haltestelleneinrichtungen bzw. Fahrzeugausstattungen (facilities).

6.4 Fehlerzustände beim Betrieb von TRIAS-Diensten

Die Fehlerzustände beim Betrieb von TRIAS-Diensten werden durch Fehlercodes signalisiert, die in der Struktur *ErrorMessage* übermittelt werden können. *ErrorMessage* kann an den meisten Stellen mehrfach auftreten und daher auch eine mehrfache, vielschichtige Fehlersituation beschreiben. In ErrorMessage können Fehlercodes auftreten, die

- aus den SIRI-Diensten geerbt werden,
- allgemeine, dienstübergreifende TRIAS-Fehlersituationen beschreiben oder
- dienstspezifische Fehlersituationen anzeigen.

Die TRIAS-Fehlercodes sind durch ein Präfix gekennzeichnet, das den jeweiligen Dienst angibt (z. B. ***STOPEVENT_***) oder anzeigt, dass es sich um einen allgemeinen Fehlerzustand handelt (***TRIASGENERIC_***).

6.4.1 Fehlercodes aus SIRI

In SIRI (CEN, TS 15531 Part 2, 2011), Kapitel 5.7, werden eine Reihe von Fehlercodes definiert, die für das Nachrichtenübermittlungsverfahren eine wichtige Rolle spielen. Diese Codes sind in die Gruppen Erfolg (Success), Systemfehler (Systemic Error) und Anwendungsfehler (Application Error) eingeteilt (vgl. Tabelle 3).

Group	Condition	Description (Beschreibung)
Success	***OK (true)***	Request successful. (Anfrage erfolgreich bearbeitet.)
Systemic Error	***RequestTimeout***	Server not responding. (Server antwortet nicht.)
	InvalidRequest	The server does not "understand" the request. The client should not repeat the request. (Der Server "versteht" die Anfrage nicht. Der Client braucht die Anfrage nicht zu wiederholen.)
	Unauthorized	User name and password are required for the request, or credentials not satisfied. (Benutzername und Passwort sind für die Anfrage erforderlich, oder die Berechtigungen reichen nicht aus.)
	Forbidden	The server "understands" the request, but cannot carry it out. (Der Server "versteht" die Anfrage, kann sie aber nicht ausführen.)
	NotFound	The requested URL was not found. (Die angefragte URL konnte nicht gefunden warden.)
Applicati on Error	***VersionNotSupported***	Service is not available. (Die angefragte Version des Dienstes ist nicht verfügbar.)
	CapabilityNot-Supported	Service does not support the requested capability. (Die angeforderte Funktionalität wird vom Dienst nicht unterstützt.)
	ServiceNotAvailable	Functional service is not available to use (but it is still capable of giving this response). (Der funktionale Dienst kann keine Anfragen abarbeiten (obwohl er in der Lage ist, eine Antwort zu geben).)
	AccessNotAllowed	Requestor is not authorised to the service or data requested. (Der Anfrager ist für den Zugriff auf den Dienst oder die Daten nicht autorisiert.)
	NoInfoForTopic	Valid request was made but service does not hold any data for the requested topic expression. (Die Anfrage ist gültig, der Dienst kann aber über den angrefragten Fachinhalt keine Auskunft geben.)
	UnknownSubscriber	Subscriber not found. (Der Abonnent wurde nicht gefunden.)
	UnknownSubscription	Subscription not found. (Das Abonnement wurde nicht gefunden.)
	AllowedResource-UsageExceeded	Valid request was made, but request would exceed the permitted resource usage of the client. (Die Anfrage ist gültig, sie überschreitet aber das dem Client zugestandene Ressourcen-Limit.)
	OtherError	Other Error Type. (Sonstiger Fehler.)

Tabelle 3: Liste der Fehlercodes, wie sie in SIRI für das Nachrichtenübermittlungsverfahren definiert werden.

6.4.2 Allgemeine TRIAS-Fehlerzustände

In *ErrorMessage* können folgende allgemeine Fehlerzustände auftreten:

FehlerCode	**Fehlerbedeutung**
AUTH_FAILURE	Dieser Fehler tritt auf, wenn eine Anfrage mit ungültiger oder nicht prüfbarer Signatur empfangen wurde.
AUTH_MISSING	Dieser Fehlercode tritt auf, wenn der Server zwingend eine Authentifizierung benötigt, aber eine Nachricht ohne Signatur empfangen wurde.
AUTH_USER_UNKNOWN	Dieser Fehlercode wird zurückgegeben, wenn die Authentifikation fehlschlägt, weil der Benutzer unbekannt ist.
TRIASGENERIC_ERROR	Bei der Verarbeitung der Anfrage ist ein Fehler aufgetreten, der nicht durch einen speziellen Fehlercode abgedeckt wird, Einzelheiten werden im Text der Fehlermeldung genannt.
TRIASGENERIC_SERVICENOTSUPPORTED	In der Anfrage wurde ein Dienst spezifiziert, der vom Server nicht unterstützt wird (z.B. Dienst ConnectionDemand).
TRIASGENERIC_REQUESTNOTSUPPORTED	Es wurde eine Anfrage spezifiziert, die vom Server nicht unterstützt wird (z.B. Anfrage FacilityStatusReport).
TRIASGENERIC_FEATURENOTSUPPORTED	In der Anfrage wurde ein Feature spezifiziert, das vom Server nicht unterstützt wird (z.B. Parameter NotVia in TripRequest)
TRIASGENERIC_LANGUAGENOTSUPPORTED	In der Anfrage wurde eine Sprache für die Anzeige der Ergebnistexte spezifiziert, die vom Server nicht unterstützt wird (zumindest im Kontext der vorliegenden Anfrage).
TRIASGENERIC_EXCEPTIONFROMREQUESTEDLANGUAGE	In der Anfrage wurde eine Sprache für die Anzeige der Ergebnistexte spezifiziert, die vom Server nicht bei allen Textelementen der Antwort unterstützt wird.
TRIASGENERIC_DATAVERSIONNOTAVAILABLE	Die in der Anfrage angeforderte Datenversion konnte vom Server nicht berücksichtigt werden.

Tabelle 4: Generische TRIAS-Fehlermeldungen, die in allen Nachrichten auftreten können.

6.5 Haltsequenznummern und Fahrtabschnitte in TRIAS-Diensten

In diversen TRIAS-Diensten werden Haltsequenznummern oder Fahrtabschnitte verwendet. Eine Haltsequenznummer gibt an, an wievielter Stelle ein Halt in der Haltestellenfolge einer Fahrt steht. Dies wird beispielsweise in den Diensten *StopEvents* und *TripInfo* verwendet. Ein Fahrtabschnitt besteht aus einer Haltsequenznummer, die den Beginn des Abschnitts kennzeichnet und aus einer zweiten Haltsequenznummer, die das Ende des Abschnitts kennzeichnet. Solche Fahrtabschnitte werden genutzt, um deutlich zu machen, dass bestimmte Eigenschaften einer Fahrt nur auf gewissen Abschnitten der Fahrt gelten (siehe *ServiceAttributeStructure, ServiceSectionStructure* und *ParallelServiceStructure*).

Die Verwendung von Haltsequenznummern in TRIAS-Diensten unterliegt gewissen, stets gleichen Konventionen, die im Folgenden zusammengefasst werden:

- Alle Haltsequenznummern beziehen sich stets auf die vollständige Fahrt (wie sie im Fahrplan der EKAP enthalten ist). Auch in Kontexten, in denen nur ein Teil einer Fahrt genutzt wird (z.B. im Dienst *Trips*) beziehen sich die Haltsequenznummern auf die gesamte Fahrt und nicht auf den beauskunfteten Abschnitt.

- Haltsequenznummern sind stets als *StopSeqNumber* bezeichnet.
- Haltsequenznummern werden stets von 1 an gezählt.

Im Bezug auf Fahrtabschnitte gibt es einige zusätzlichen Konventionen, die im Folgenden zusammengefasst werden:

- Fahrtabschnitte werden stets mit der Gruppe *StopSeqIntervalGroup* und ihren beiden Elementen *FromStopSeqNumber* und *ToStopSeqNumber* gekennzeichnet. Für diese Elemente gelten obigen Regeln für Haltsequenznummern.
- *FromStopSeqNumber* und *ToStopSeqNumber* sind jeweils optional. Werden sie angegeben, beziehen sie sich stets auf die komplette Fahrt, auch in Zusammenhängen, in denen nur ein Teil einer Fahrt genutzt wird (z.B. in *Trips*).
- Wird *FromStopSeqNumber* im Dienst *Trips* nicht angegeben, so beginnt der zugehörige Fahrtabschnitt mit dem Fahrtabschnitt, der tatsächlich genutzt wird. Über „davor“ wird keine Aussage getroffen. Wird *ToStopSeqNumber* im Dienst *Trips* nicht angegeben, so endet der zugehörige Fahrtabschnitt mit dem Fahrtabschnitt der tatsächlich genutzt wird. Über „danach“ wird keine Aussage getroffen. Auf diese Weise kann für Fahrteigenschaften, die für den gesamten genutzten Fahrtabschnitt gelten, auf die Angabe eines expliziten Fahrtabschnittes verzichtet werden.
- Wird *FromStopSeqNumber* in den Diensten *StopsEvents* oder *TripInfo* nicht angegeben, so beginnt der zugehörige Fahrtabschnitt mit der Fahrt. Wird *ToStopSeqNumber* in den Diensten *StopEvents* oder *TripInfo* nicht angegeben, so endet der zugehörige Fahrtabschnitt mit der Fahrt. Auf diese Weise kann für Fahrteigenschaften, die für gesamte Fahrt gelten, auf die Angabe eines expliziten Fahrtabschnittes verzichtet werden.

6 Services, XML schemes and conventions

This document describes interface definitions for services between software components. For a more detailed explanation of the task of these services and possible system architectures, see the basic VDV guidelines (VDV guideline 431-1, 2014) and (VDV guideline 430, 2014).

The TRIAS interface services are defined as XML schema. The first part of this chapter provides an overview of the services and their implementation as XML schemes. A few structure definitions are useful in multiple services and therefore are hierarchically defined in several scheme files as a common basis to enable repeated use. The concept followed here is very strongly oriented towards the principles of object orientation. The commonly used structure definitions are described in the second part. The third part introduces the XML schemes that are imported from SIRI. A classification of error states can be found in the fourth part.

6.1 Services provided

The TRIAS interface family currently comprises the following services:

Service	Name of Request element	Schema file	Chapter
Location information	LocationInformationRequest	Trias_Locations.xsd	8
Trip information	TripRequest	Trias_Trips.xsd	8
Stop events	StopEventRequest	Trias_StopEvents.xsd	10
Positioning	PositioningRequest	Trias_Positioning.xsd	11
Trip information (EKAP)	TripInfoRequest	Trias_TripInfo.xsd	12
Connection report	ConnectionDemandRequest	Trias_Connections.xsd	13.1.1
Connection status	ConnectionStatusRequest	Trias_Connections.xsd	13.1.2
Info about Missed connection	ConnectionStatusResponse	Trias_Connections.xsd	13.1.3
Connection response	ConnectionReportRequest	Trias_Connections.xsd	13.1.4
Fare and tariff calculation	FaresRequest	Trias_Fares.xsd	14
Booking information	BookingInfoRequest	Trias_Booking.xsd	16
IV-Routing	IndividualRouteRequest	Trias_-IndividualTrips.xsd	17
Map service	MapServiceRequest ImageCoordinatesRequest GeoCoordinatesRequest	Trias_Maps.xsd	18

Service	Name of Request element	Schema file	Chapter
Damage report	FacilityStatusReport	Trias_Facilities.xsd	19
Status of facilities	FacilityRequest	Trias_Facilities.xsd	19.4
Notification service	SubscriptionRequest	Trias.xsd	7.1.2+ 20
Personalisation service	PersonalisationRequest	Trias_-Personalisation.xsd	21
Vehicle information	VehicleDataRequest	Trias_-VehicleInterface.xsd	22
Vehicle interaction	VehicleInteractionRequest	Trias_-VehicleInterface.xsd	23
Service register	ServiceRegisterRequest	Trias_-ServiceRegister.xsd	24
Authentication	AbstractTriasServiceRequest (inherited from all TRIAS messages)	Trias_-RequestSupport.xsd	7.9+ 25

Table 2: List of TRIAS services and their request elements

6.2 XML schema used across services

In order to avoid the need to repeatedly and redundantly define structures that are used in more than one service, commonly used basic XML schemes are introduced to include them hierarchically. The inclusion sequence and the customisation of scheme files are selected in such a way that content-related elements are placed together in one file and that each scheme includes as much as possible, as necessary for one's own tasks.

The commonly used basic scheme files are described in detail in chapter 7.

6.3 Imported Siri schema

The following scheme files are imported from the SIRI interface specification of version 1.4 to TRIAS:

- siri.xsd
- siri_facilities-v1.2.xsd
- siri_situation-v1.1.xsd
- siri_requests-v1.3.xsd
- siri_common-v1.4.xsd
- siri_situationExchange_service.xsd
- siri_facilityMonitoring_service.xsd.

Thanks to the import of SIRI definitions, the SIRI procedures can also be used for message exchange of TRIAS messages. Additionally, specific structure definitions can be reused from SIRI which ensures consistency between these interface standards. This concerns the definition of modes, situations and stop facilities or passenger facilities among others.

6.4 Error states when operating TRIAS services

The error states when operating TRIAS services are signalled by error codes which can be transferred into the ErrorMessage structure. ErrorMessage can occur multiple times in most places and therefore also describes an often-occurring, multi-layered error situation. In ErrorMessage, error codes can occur that:

- are inherited from the SIRI services,
- describe general TRIAS error situations across services or
- indicate service-specific error situations.

The TRIAS error codes are indicated by a prefix that is specified by the respective service (e.g. **STOPEVENT_**) or shows that there is a general error state (**TRIASGENERIC_**).

6.4.1 Error codes from SIRI

In SIRI (CEN, TS 15531, part 2, 2011), section 5.7, a number of error codes are defined which play an important role in the message transfer procedure. These codes are divided into the following groups: success, systemic error and application error (see table 3).

Group	Condition	Description
Success	***OK (true)***	Request successful.
Systemic Error	***RequestTimeout***	Server not responding.
	InvalidRequest	The server does not "understand" the request. The client should not repeat the request.
	Unauthorized	User name and password are required for the request, or credentials not satisfied.
	Forbidden	The server "understands" the request, but cannot carry it out.
	NotFound	The requested URL was not found.
Applic ation Error	***VersionNotSupported***	Service is not available.
	CapabilityNotSupporte	Service does not support the requested capability.
	ServiceNotAvailable	Functional service is not available to use (but it is still capable of giving this response).
	AccessNotAllowed	Requestor is not authorised to the service or data requested.
	NoInfoForTopic	Valid request was made but service does not hold any data for the requested topic expression.
	UnknownSubscriber	Subscriber not found.
	UnknownSubscription	Subscription not found.
	AllowedResour ceUsageExceed ed	Valid request was made, but request would exceed the permitted resource usage of the client.
	OtherError	Other Error Type.

Table 3: List of error codes as they are defined in SIRI for the message transfer procedure

6.4.2 General TRIAS error states

In ErrorMessage, the following general error states can appear:

Error code	**Error meaning**
AUTH_FAILURE	This error occurs when a request has been received with invalid or unverifiable signature.
AUTH_MISSING	This error code occurs when the server requires an authentication but a message without signature has been received.
AUTH_USER_UNKNOWN	This error code is returned when the authentication fails because the user is unknown.
TRIASGENERIC_ERROR	An error has occurred when processing the request which is not covered by a special error code. Details are given in the text of the error message.
TRIASGENERIC_SERVICENOTSUPPORTED	A service has been specified in the request which is not supported by the server (e.g. ConnectionDemand service).
TRIASGENERIC_REQUESTNOTSUPPORTED	A request has been specified which is not supported by the server (e.g. FacilityStatusReport request).
TRIASGENERIC_FEATURENOTSUPPORTED	A feature has been specified in the request which is not supported by the server (e.g. parameter NotVia in TripRequest).
TRIASGENERIC_LANGUAGENOTSUPPORTED	A language for displaying the result texts has been specified in the request which is not supported by the server (at least in the context of this request).
TRIASGENERIC_EXCEPTIONFROMREQUESTEDLANGU AGE	A language for displaying the result texts has been specified in the request which is not supported by the server for all the text elements of the response.
TRIASGENERIC_DATAVERSIONNOTAVAILABLE	The data version required in the request could not be considered by the server.

Table 4: General TRIAS error messages that can appear in all messages

6.5 Stop sequence numbers and journey sections in TRIAS services

Stop sequence numbers or journey sections are used in many TRIAS services. A stop sequence number specifies the position of a stop in the stop sequence of a journey. It is used in services such as StopEvents and TripInfo. A journey section consists of a stop sequence number, which indicates the beginning of the section, and of another stop sequence number, which indicates the end of the section. Such journey sections are used to emphasise that certain properties of a journey are applicable only to certain sections of the journey (see ServiceAttributeStructure, ServiceSectionStructure and ParallelServiceStructure).

The use of stop sequence numbers in TRIAS services are subject to certain consistent conventions which are summarised below:

- All the stop sequence numbers always refer to the complete journey (as included in the journey timetable of EKAP). Even in contexts, in which only a part of a journey is used (e.g. in the service Trips), the stop sequence numbers refer to the complete journey and not to the mentioned section.
- Stop sequence numbers are always called as StopSeqNumber.

- The stop sequence numbers are always counted from 1.

There are a few additional conventions with reference to the journey sections which are summarised below:

- Journey sections are always identified with the group *StopSeqIntervalGroup* and their two elements *FromStopSeqNumber* and *ToStopSeqNumber*. The above rules for stop sequence numbers apply to these elements.
- *FromStopSeqNumber* and *ToStopSeqNumber* are optional. If they are specified, they always refer to the complete journey, even in contexts in which only a part of a journey is used (e.g. in Trips).
- If *FromStopSeqNumber* is not specified in the service Trips, the associated journey section begins with the journey section that is actually used. No comment is made about before that. If *ToStopSeqNumber* is not specified in the service Trips, the associated journey section ends with the journey section that is actually used. No comment is made about after that. In this way, specification of an explicit journey section is omitted for journey properties which are applicable to the total journey section used.
- If *FromStopSeqNumber* is not specified in the service *StopsEvents* or *TripInfo*, the associated journey section begins with the journey. If *ToStopSeqNumber* is not specified in the service *StopsEvents* or *TripInfo*, the associated journey section ends with the journey. In this way, specification of an explicit journey section is omitted for journey properties which are applicable to the entire journey.

7 Gemeinsam genutzte XML-Strukturen

In diesem Kapitel werden die XML-Strukturen erläutert, die als Basisobjekte in den dienstübergreifend genutzten XML-Schemadateien definiert werden. Die Gliederung ergibt sich anhand der einzelnen Schemadateien.

7 Common XML structures

This chapter explains the basic XML structures that are defined as basic objects in the XML scheme files used across services. The classification is done with the help of individual schema files.

7.1 Trias, the root element

TRIAS stands for Travellers' Realtime Information and Advisory Standard.

The XML schema file Trias.xsd defines the general root element TRIAS which serves as a common base for all messages of all TRIAS services.

The subsequent sections describe the complex structures defined in Trias.xsd:

7.1.1 ServiceRequestStructure

ServiceRequestStructure			*+Structure*	**Basic structure for every TRIAS request (without subscription)**
ServiceRequestContext	**:::**	**1:1**	*AbstractTriasServiceRequest*	Common request context (see 7.9.2).
	RequestPayload	**1:1**	*RequestPayload*	Service-specific request content (see 7.1.3).

Table 5: Description of structure ServiceRequestStructure

7.1.2 SubscriptionRequestStructure

SubscriptionRequestStructure				*+Structure*	**Basic structure for every TRIAS subscription request**
SubscriptionRequestContext	***:::***		**1:1**	*AbstractTriasSubscription-Request*	Common request context (see 7.9.3).
AlertSettingsGroup	*AlertTimeWindow*		0:*	*WeekdayTimePeriod*	Time window in which messages must be sent (see 7.4.10).
	MaximumAlertFrequency		0:1	*xs:duration*	Maximum frequency for messages for the same reason
	MaximumTimeBeforeEvent		0:1	*xs:duration*	Maximum time before a message calculated from the beginning of the event. Only in the context of events, whose beginning is known in advance.
SubscriptionRequest	***a***	***SituationExchangeSubscriptionRequest***	**-1:1**	*SituationExchangeSubscriptionRequest*	Content of subscription request for general event and error messages (see chapter 20).
	b	***FacilityMonitoringSubscriptionRequest***		*FacilityMonitoringSubscriptionRequest*	Content of subscription request for status messages about the infrastructure of stops and vehicles (see chapter 20).
	c	***TripMonitoringSubscriptionRequest***		*TripMonitoringSubscription Request*	Content of subscription request for messages about a certain trip connection (see 20.3.1).

Table 6: Description of structure SubscriptionRequestStructure

7.1.3 RequestPayloadStructure

RequestPayloadStructure				*+Structure*	**Element for selecting the desired TRIAS service.**
	a	***BookingInfoRequest***		*BookingInfoRequest*	Request for booking information (see 16.2.1).
	b	***ConnectionDemandRequest***		*ConnectionDemandRequest*	Request for pre-announcement of connection (see 13.4.1).
	c	***ConnectionDemandDeleteRequest***		*ConnectionDemandDeleteRequest*	Deletion of a connection pre-announcement (see 13.4.2).
	d	***ConnectionReportRequest***		*ConnectionReportRequest*	Report of the passenger, whether a connection has worked out (see 13.8.1).
	e	***ConnectionStatusRequest***		*ConnectionStatusRequest*	Request for connection status (see 13.6.1).
	f	***FacilityRequest***		*FacilityRequest*	Request for vehicle and infrastructure facilities (see 19.6.1).
	g	***FacilityStatusReport***		*FacilityStatusReport*	Report of status of vehicle and infrastructure facilities in an active subscription (see 19.4.1).
	h	***FaresRequest***		*FaresRequest*	Request for fare calculation service (see 14.2.1).
	i	***GeoCoordinatesRequest***		*GeoCoordinatesRequest*	Request for geo-coordinates (see 18.2.3).
	j	***ImageCoordinatesRequest***		*ImageCoordinatesRequest*	Request for image coordinates (see 18.2.2).
	k	***IndividualRouteRequest***		*IndividualRouteRequest*	Request for an individual route (see 17.2.1).

	l	***LocationInformationRequest***		*LocationInformationRequest*	Request for location information service (see 8.3.1).
	m	***MapServiceRequest***		*MapServiceRequest*	Request for map service (see 18.2.1).
	n	***PersonalisationRequest***		*PersonalisationRequest*	Request for personalisation service (see 21.5.1).
	o	***PositioningRequest***		*PositioningRequest*	Request for positioning service (see 11.2.1).
	p	***RefineRequest***		*RefineRequest*	Request for refining structures (see 15.2.1)
	q	***ServiceRegisterRequest***		*ServiceRegisterRequest*	Request for service registration service (see 24.3.1).
	r	***StopEventRequest***		*StopEventRequest*	Request for stop events (see 10.3.1).
	s	***TripInfoRequest***		*TripInfoRequest*	Request for trip information (see 12.2.1).
	t	***TripRequest***		*TripRequest*	Request for an intermodal route calculation (see 9.2.1).
	u	***VehicleDataRequest***		*VehicleDataRequest*	Request for vehicle information (see 22.2.1).
	v	***VehicleInteractionRequest***		*VehicleInteractionRequest*	Requests, which are directed to a vehicle, in order to trigger an action there (see 23.2.1). Contains the StopRequestRequest among others.

Table 7: Description of structure RequestPayloadStructure

7.1.4 ServiceDeliveryStructure

ServiceDeliveryStructure			*+Structure*	**Basic structure for every TRIAS-specific response.**
	:::	**1:1**	*AbstractTriasResponse*	Common response context (see 7.9.4).
	DeliveryPayload	**1:1**	*DeliveryPayload*	Service-specific response content (see 7.1.5).

Table 8: Description of structure ServiceDeliveryStructure

7.1.5 DeliveryPayloadStructure

DeliveryPayloadStructure				*+Structure*	**Element for selecting suitable TRIAS response.**
	a	***BookingInfoResponse***		*BookingInfoResponse*	Response with booking information (see 16.3.1).
	b	***ConnectionDemandResponse***		*ConnectionDemandResponse*	Response for pre-announcement of connection (see 13.5.1).
	c	***ConnectionDemandDeleteResponse***		*ConnectionDemandDeleteResponse*	Confirmation of deletion of a connection pre-announcement (see 13.5.2).
	d	***ConnectionReportResponse***		*ConnectionReportResponse*	Confirmation of connection report message (see 13.9.1).
	e	***ConnectionStatusNotification***		*ConnectionStatusNotification*	Notification of connection status in an active subscription (see 13.6.2).

	f	***ConnectionStatusResponse***		*ConnectionStatusResponse*	Response to connection status request (see 13.7.1).
	g	***FacilityMonitoringDelivery***		*siri:FacilityMonitoringDelivery*	Update message during subscription for facility features and equipment (see chapter 20).
	h	***FacilityResponse***		*FacilityResponse*	Response to vehicle and infrastructure facilities (see 19.7.1).
	i	***FacilityStatusReportResponse***		*FacilityStatusReportResponse*	Confirmation for reporting the status of vehicle and infrastructure facilities (damage report, see 19.5.1).
	j	***FaresResponse***		*FaresResponse*	Response for fare calculation request (see 14.3.1).
	k	***GeoCoordinatesResponse***		*GeoCoordinatesResponse*	Response to request for geo-coordinates (see 18.3.3).
	l	***ImageCoordinates Response***		*ImageCoordinatesResponse*	Response to request for image coordinates (see 18.3.2).
	m	***IndividualRouteResponse***		*IndividualRouteResponse*	Response with individual routes calculated (see 17.3.1).
	n	***LocationInformation Response***		*LocationInformationResponse*	Response to location information service (see 8.4.1).
	o	***MapServiceResponse***		*MapService Response*	Response to map service (see 18.3.1).
	p	***PersonalisationResponse***		*PersonalisationResponse*	Response to personalisation service (see 19.5.1).
	q	***PositioningResponse***		*Positioning Response*	Response to positioning service (see 11.3.1).
	r	***RefineResponse***		*RefineResponse*	Response to refining service (see 15.3.1).
	s	***ServiceRegisterResponse***		*ServiceRegisterResponse*	Response to service registration service (see 24.4.1).
	t	***SituationExchangeDelivery***		*siri:SituationExchangeDelivery*	Update message during subscription for fault information (see chapter 20).
	u	***StopEventResponse***		*StopEventResponse*	Response with stop events (see 10.3.1).
	v	***TripInfoResponse***		*TripInfoResponse*	Response with trip information (see 12.3.1).
	w	***TripMonitoringDelivery***		*TripMonitoringDelivery*	Update message during subscription for connection statuses (see chapter 20.4.1).
	x	***TripResponse***		*TripResponse*	Response to intermodal route calculation (see 9.3.1).
	y	***VehicleDataResponse***		*VehicleData Response*	Response with vehicle information (see 23.3.1).
	z	***VehicleInteraction Response***		*VehicleInteractionResponse*	Vehicle response to an interaction request (see 23.3.1). Contains the StopRequestResponse among others.

Table 9: Description of structure DeliveryPayloadStructure

7.2 Trias_Utility

The XML schema definition Trias_Utility.xsd contains a range of types and structures that can be reused as basic types in other definitions. The definitions in Trias_Utility are not directly related to the public transport domain.

7.2.1 Simple types

The following simple types are defined:

Type name	***Basic type***	**Description**
PercentType	*xs:nonNegativeInteger*	Percentage as integer value. Maximum value is 100.
OpenPercentType	*xs:nonNegativeInteger*	Percentage as integer value with no upper limit.
BitStringType	*xs:string*	String that can only consist of zeros and ones.
DistanceType	*xs:nonNegativeInteger*	Type for indicating distances (in m).
LengthType	*xs:nonNegativeInteger*	Type for indicating lengths (in m).
SpeedType	*xs:nonNegativeInteger*	Type for indicating speed (in m/s).
PriorityType	*xs:nonNegativeInteger, [1.5]*	Priority values from 1 (highest priority) to 5 (lowest priority).
LongitudeType	*xs:decimal*	Longitude.
LatitudeType	*xs:decimal*	Latitude.
AltitudeType	*xs:decimal*	Height above sea level in meters.
AbsoluteBearingType	*xs:nonNegativeInteger*	Compass direction in degrees. North = 0 degrees; values increasing in clockwise direction.
PhoneNumberType	*xs:normalizedString*	Type for indicating a telephone number.

Table 10: List of simple type definitions in Trias_Utility.xsd

7.2.2 InternationalTextStructure

InternationalTextStructure			*+Structure*	**A text with a text ID and specification of language in which it is written.**
	Text	**1:1**	*xs:normalized String*	Text.
	TextId	0:1	*xs:NMTOKEN*	ID of text.
	Language	0:1	*xs:language*	Language in which the text is written.

Table 11: Description of structure InternationalTextStructure

Elements of type InternationalText are used in order to be able to write texts in different languages. In order to be able to display several languages, for example in multilingual regions, the type “unbound” is provided in the schema.

7.2.3 GeoPositionStructure

GeoPositionStructure			*+Structure*	**Geographic position in WGS84.**
	Longitude	**1:1**	*Longitude*	Longitude with regard to Greenwich meridian. Value range from -180 degrees (west) to +180 degrees (east).
	Latitude	**1:1**	*Latitude*	Latitude with regard to the equator. Value range from -90 degrees (south) to +90 degrees (north).
	Altitude	0:1	*Altitude*	Height above sea level in meters.

Table 12: Description of structure GeoPositionStructure

7.2.4 WebLinkStructure

WebLinkStructure			*+Structure*	**URL with caption text of a resource in the web**
	Label	**1:***	*International-Text*	Caption text of the link (see 7.2.2).
	Url	**1:1**	*xs:anyURI*	URL of web resource.

Table 13: Description of structure WebLinkStructure

In order to be able to access further information about an object, elements of the type WebLinkStructure are added in some places in the TRIAS responses. A client can use the URLs contained in it in order to execute further actions or access information. As the clients can be of different types (e.g. app or web-browser), the providers of the web resources are requested to provide the widest possible support of different client types.

7.3 Trias_ModesSupport

The XML schema definition Trias_ModesSupport.xsd contains a range of basic types and structures that can be used for classifying modes of transport. These definitions strictly follow the TPEG coding which is also used in SIRI.

7.3.1 Simple types

The following simple types are defined:

Type name	***Values***	**Description**
IndividualModesEnumeration	*walk | cycle | taxi | self-drive-car | others-drive-car | motorcycle | truck*	Classification of individual modes of transport.
ContinuousModesEnumeration	*walk | demandResponsive | replacementService*	Classification of continuous modes which can take place at any time (without timetable). *walk*: walk *demandResponsive*: On-demand transport without timetable *replacementService*: e.g. Shuttle-Service in substitute transport
InterchangeModesEnumeration	*walk | parkAndRide | bikeAndRide | carHire | bikeHire | protectedConnection | guaranteedConnection | remainInVehicle | changeWithinVehicle | checkIn | checkOut*	Classification of change processes

PtModesEnumeration	*all | unknown | air | bus | trolleyBus | tram | coach | rail | intercityRail | urbanRail | metro | water | cableway | funicular | taxi*	Classification of modes of public transport. (as per TPEG pti_table 01).
RailSubmodeEnumeration	*unknown | undefined | local | highSpeedRail | suburbanRailway | regionalRail | interregionalRail | longDistance | international | sleeperRailService | nightRail | carTransportRailService | touristRailway | railShuttle | replacementRailService| specialTrain| crossCountryRail | rackAndPinionRailway*	Sub-classification of trains (as per TPEG pti_table 02).
CoachSubmodeEnumeration	*unknown | undefined | internationalCoach | nationalCoach | shuttleCoach | regionalCoach | specialCoach | sightseeingCoach | touristCoach | commuterCoach*	Sub-classification of intercity buses (as per TPEG pti_table 03).
MetroSubmodeEnumeration	*unknown | undefined | metro | tube | urbanRailway*	Sub-classification of underground trains (as per TPEG pti_table 04).
BusSubmodeEnumeration	*unknown | undefined | localBus | regionalBus | expressBus | night- Bus | postBus | specialNeedsBus | mobilityBus|mobilityBusForRegisteredDisabled | sightseeingBus | shuttleBus | schoolBus | schoolAndPublicServiceBus| railReplacementBus|demandAndResponseBus | airportLinkBus*	Sub-classification of buses (as per TPEG pti_table 05).
TramSubmodeEnumeration	*unknown | undefined | cityTram | localTram|regionalTram| sightseeingTram | shuttleTram*	Sub-classification of trams (as per TPEG pti_table 06).
WaterSubmodeEnumeration	*unknown | undefined | internationalCarFerry | nationalCarFerry | regionalCarFerry | localCarFerry | internationalPassengerFerry | nationalPassengerFerry | regionalPassengerFerry | localPassengerFerry | postBoat | trainFerry | roadFerryLink | airportBoatLink | highSpeedVehicleService | highSpeedPassengerService | sightseeingService | schoolBoat | cableFerry | riverBus | scheduled- Ferry | shuttleFerryService*	Sub-classification of water mode of transport (as per TPEG pti_table 07).
AirSubmodeEnumeration	*unknown | undefined | internationalFlight | domesticFlight | intercontinentalFlight | domes- ticScheduledFlight | shuttleFlight | intercontinentalCharterFlight | internationalCharterFlight | roundTripCharterFlight | sightseeing- Flight | helicopterService | domes- ticCharterFlight | SchengenAreaFlight | airshipService | shortHaulInternationalFlight | canalBarge*	Sub-classification of air mode of transport (as per TPEG pti_table 08).
TelecabinSubmodeEnumeration	*unknown | undefined | telecabin | cableCar | lift | chairLift | dragLift | telecabinLink*	Sub-classification of types of lift and cable car (as per TPEG pti_table 09).
FunicularSubmodeEnumeration	*unknown | funicular | allFunicular- Services | undefinedFunicular*	Sub-classification of cableways (as per TPEG pti_table 10).
TaxiSubmodeEnumeration	*unknown | undefined | communal- Taxi | waterTaxi | railTaxi | bikeTaxi | blackCab | miniCab | allTa- xiServices*	Sub-classification of taxi types (nach TPEG pti_table 11).

Table 14: List of simple type definitions in Trias_ModesSupport.xsd

The following sections describe the complex structures which are defined in Trias_ModesSupport.

7.3.2 IndividualTransportOptionsStructure

IndividualTransportOptionsStructure			*+Structure*	**Types of individual transport and their usage limits as specified by the user.**
	Mode	**1:1**	*IndividualModes Enumeration*	Specification of individual transport type. Values for footpath, bicycle, taxi, self-driven car, car driven by someone else, motorcycle and truck are permitted here. The mode "self-driving car" requires a long-term parking space when switching to another mode of transport and is hence a generalised synonym for Park&Ride. As opposed to that, the mode "others-driven car" only requires a place to let the passengers alight.
	MaxDistance	0:1	*Distance*	Maximum distance up to which the use of this individual transport type is permitted.
	MaxDuration	0:1	*xs:duration*	Maximum duration up to which the use of this individual transport type is permitted.
	MinDistance	0:1	*Distance*	Minimum distance from which the use of this individual transport type is permitted.
	MinDuration	0:1	*xs:duration*	Minimum duration from which the use of this individual transport type is permitted.
	Speed	0:1	*OpenPercent*	Relative speed in percentage. The value 100 is the standard speed. Values less than 100 lower the speed and values greater than 100 increase the speed proportionately.

Table 15: Description of structure IndividualTransportOptionsStructure

7.3.3 PtSubmodeChoiceGroup

PtSubmodeChoiceGroup				*+Group*	**Group for selecting sub-types of modes of transport.**
	a	*AirSubmode*	-0:1	*AirSubmodeEnumeration*	Subtypes of air transport mode.
	b	*BusSubmode*		*BusSubmodeEnumeration*	Subtypes of buses.
	c	*CoachSubmode*		*CoachSubmodeEnumeration*	Subtypes of intercity buses.
	d	*FunicularSubmode*		*FunicularSubmodeEnumeration*	Subtypes of cableways.
	e	*MetroSubmode*		*MetroSubmodeEnumeration*	Subtypes of underground trains.
	f	*RailSubmode*		*RailSubmodeEnumeration*	Subtypes of trains.
	g	*TelecabinSubmode*		*TelecabinSubmodeEnumeration*	Subtypes of lift and cable car.
	h	*TramSubmode*		*TramSubmodeEnumeration*	Subtypes of trams.
	i	*WaterSubmode*		*WaterSubmodeEnumeration*	Subtypes of water transport mode.

Table 16: Description of group PtSubmodeChoiceGroup

7.3.4 ModeStructure

ModeStructure			*+Structure*	**Mode of transport with classification and names.**
Mode	***PtMode***	**1:1**	*PtModesEnumeration*	Specification of public transport mode.
PtSubmodeChoice	:::	-0:1	*PtSubmodeChoice*	Subtypes of transport modes (see 7.3.3).
	Name	0:*	*International-Text*	Name of mode of transport.
	ShortName	0:*	*International-Text*	Short name or abbreviation.
	Description	0:*	*International-Text*	Descriptive text.

Table 17: Description of structure ModeStructure

7.3.5 PtModeFilterStructure

PtModeFilterStructure			*+Structure*	**Structure for filtering according to mode of transport**
	Exclude	0:1	*xs:boolean*	Indicator, whether the mode of transport specified in the list is excluded (value *true*) or should be used uniquely (value *false*). Default setting is *true*.
	PtMode	0:*	*PtModesEnumeration*	Types of public transport.
PtSubmodeChoice	*:::*	0:*	*PtSubmodeChoice*	Subtypes of public transport.

Table 18: Description of structure PtModeFilterStructure

The specification of PtModes and PtSubmodeChoice is additive: the specifications are added regardless of the mode. The PtModes and sub-modes are excluded in the mode “exclude”. The PtModes and sub-modes are included in the mode “include”.

7.4 Trias_Common

7.4.1 Simple types

The following simple types are defined:

Type name	***Basic type***	**Description**
ParticipantCodeType	*xs:normalizedString*	ID of a communication partner.
OperatorCodeType	*xs:normalizedString*	ID of a transport company.
LineCodeType	*xs:normalizedString*	ID of a line.
DirectionCodeType	*xs:normalizedString*	ID of a line direction.
JourneyCodeType	*xs:normalizedString*	ID of a journey.
VehicleCodeType	*xs:normalizedString*	ID of a vehicle.
FacilityCodeType	*xs:normalizedString*	ID of a facility.
OwnerCodeType	*xs:normalizedString*	ID of a responsible organisation (owner).
OperatingDayCodeType	*xs:normalizedString*	ID of an operating day.

Table 19: List of simple type definitions in Trias_Common.xsd

In order to be able to understand the codes of lines, transport companies etc. across the system, certain agreements must be made. They are described in chapter 5.

The following sections describe the complex structures which are defined in Trias_Common.

7.4.2 ErrorMessageStructure

ErrorMessageStructure			*+Structure*	**Structure for notifying about error states.**
	Code	**1:1**	*xs:normalizedString*	Code of error state.
	Text	0:*	*+InternationalText*	Description of error state.

Table 20: Description of structure ErrorMessageStructure

7.4.3 PrivateCodeStructure

PrivateCodeStructure			*+Structure*	**Object ID within a private key system (foreign key).**
	System	**1:1**	*xs:NMTOKEN*	Name of key system.
	Value	**1:1**	*xs:NMTOKEN*	Code/object ID.

Table 21: Description of structure PrivateCodeStructure

7.4.4 OperatorFilterStructure

OperatorFilterStructure			*+Structure*	**Structure for filtering according to transport companies**
	Exclude	0:1	*xs:boolean*	Indicator, whether the transport company specified in the list is excluded (value *true*) or should be used uniquely (value *false*). Default setting is *true*.
	OperatorRef	0:*	*→Operator*	Reference to transport companies. See 7.4.1.

Table 22: Description of structure OperatorFilterStructure

7.4.5 LineDirectionStructure

LineDirectionStructure			*+Structure*	**Line-ID, refined in a direction**
	LineRef	**1:1**	*→LineCode*	Reference to a line. See 7.4.1.
	DirectionRef	0:1	*→Direction Code*	Reference to a line direction. See 7.4.1.

Table 23: Description of structure LineDirectionStructure

7.4.6 LineDirectionFilterStructure

LineDirectionFilterStructure			*+Structure*	**Filter structure to include/exclude lines (line directions)**
	Line	**1:***	*+LineDirection*	Reference to the line (see 7.4.5).
	Exclude	0:1	*xs:boolean*	Indicator, whether the lines (line directions) of this list should be included in the search or excluded from it. Default is excluded.

Table 24: Description of structure LineDirectionFilterStructure

7.4.7 SharingServiceStructure

SharingServiceStructure			*+Structure*	**Structure for describing a mobility service with rented vehicles**
	OperatorRef	**1:1**	*→Operator*	Operator-ID. See 7.4.1.
	Name	0:1	*xs:string*	Name of mobility service.
	SharingModel	0:1	*singleStationBased | multipleStationBased | nonStationBased*	Type of rent and return procedure.
Sharing Service Usage	*TimeBufferBefore*	0:1	*xs:duration*	Typical time which a user must plan in order to log into the system and transfer the vehicle into readiness to travel.
	TimeBufferAfter	0:1	*xs:duration*	Typical time which a user must set apart in order to park and lock the vehicle properly and log off the system.
	InfoURL	0:1	*+WebLink*	Link to websites with additional information (see 7.2.4).

Table 25: Description of structure SharingServiceStructure

7.4.8 OperatingDaysStructure

OperatingDaysStructure			*+Structure*	**Structure for defining operating days using bit-chain.**
	From	**1:1**	*xs:date*	Start date of duration.
	To	**1:1**	*xs:date*	End date of duration.
	Pattern	**1:1**	*BitString*	Bit pattern for operating days in the period from the start date (*From*) to end date (*To*). The length of the bit pattern in *Pattern* corresponds to the number of days from *From* to *To*. "1" means that the event in question takes place on the day which corresponds to the position in the bit pattern.

Table 26: Description of structure OperatingDaysStructure

7.4.9 WeekdayTimePeriodStructure

WeekdayTimePeriodStructure			*+Structure*	**Structure for defining time periods on a weekday.**
	Weekday	0:1	*Sunday | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | PublicHoliday*	Weekday type.
	StartTime	**1:1**	*xs:time*	Start time of time period.
	Duration	**1:1**	*xs:duration*	Duration of time period.

Table 27: Description of structure WeekdayTimePeriodStructure

7.4.10 GeneralAttributeStructure

GeneralAttributeStructure			*+Structure*	**Structure for defining attributes/information.**
	Text	**1:***	*+International Text*	Attribute text for passenger information.
	Code	**1:1**	*xs:NMTOKEN*	Internal attribute code. Can be used to detect multiple occurrences of the same attribute.
AllFacilities	:::	0:1	*+AllFacilitiesGroup*	Classification as per TPEG. See 7.7.4
	Mandatory	0:1	*xs:boolean*	Defines whether the attribute must be mandatorily displayed. Default setting is *false*.
	Importance	0:1	*Percent*	Importance for prioritising attributes against each other.
	InfoURL	0:1	*xs:anyURI*	URL for further information about this attribute. If available, the entire text should be marked as link for this URL.
	Status	0:1	*Unknown | Planned | AsPlanned | NotAsPlanned | RealtimeUpdate*	Indicates the status of an attribute, for example in a refining request. • *Planned* means that the specification of the attribute is based on the planning specifications (e.g. a train should have a restaurant car as planned). • *AsPlanned* means that it is already known that the attribute is/will be as planned (e.g. a train has a restaurant car as planned). • *NotAsPlanned* means that it is already known that an attribute is not available as planned (e.g. a train does not have a restaurant car unlike as planned). • RealtimeUpdate is used to notify an about attribute which has become known/resulted only after the planning time (e.g. reference to vehicle facility).

Table 28: Description of structure GeneralAttributeStructure

7.5 Trias_LocationSupport

The XML schema definition Trias_LocationSupport.xsd contains a range of basic types and structures that can be reused in other definitions as location references (stops, stopping points, localities and POIs) and descriptions of stops and stopping points.

7.5.1 Simple types

The following simple types are defined:

Type name	***Basic type***	**Description**
StopPointCodeType	*xs:normalizedString*	Code for a stopping point.
StopPlaceCodeType	*xs:normalizedString*	Code for a stop.
LocalityCodeType	*xs:normalizedString*	Code for a locality.
PointOfInterestCodeType	*xs:normalizedString*	Code for a POI.
AddressCodeType	*xs:normalizedString*	Code for an address.

Table 29: List of simple type definitions in Trias_LocationSupport.xsd

In order to be able to understand the codes of stops, stopping points etc. across the system, certain agreements must be made. They are described in chapter 5. The following sections describe the complex structures which are defined in Trias_LocationSupport.

7.5.2 StopPointStructure

StopPointStructure			*+Structure*	**Modelling a stopping point.**
Stop Point	***StopPointRef***	**1:1**	*→StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+International Text*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+International Text*	Name suffix which can also be left out in case of space shortage, e.g.: "opposite the main entrance".
	PlannedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction.
	PrivateCode	0:*	*+PrivateCode*	Private code for this stopping point in another key system. See 7.4.3.
	ParentRef	0:1	*→StopPlace*	Reference to the stop, to which this stopping point belongs. See 7.5.1.
	LocalityRef	0:1	*→Locality*	Reference to the locality, to which this stopping point belongs. See 7.5.1.
StopAttributes	*WheelchairAccessible*	0:1	*xs:boolean*	Wheelchair accessibility of this stopping point. Default is *false*.
	Lighting	0:1	*xs:boolean*	Specification for lighting this stopping point. Default is *false*.
	Covered	0:1	*xs:boolean*	Specification whether this stopping point offers weather protection (against rain, snow, storm etc.). Default is *false*.

Table 30: Description of structure StopPointStructure

7.5.3 StopPlaceStructure

StopPlaceStructure			*+Structure*	**Modelling of a stop.**
StopPlace	***StopPlaceRef***	**1:1**	*→StopPlace*	Reference to a code for a stop. See 7.5.1.
	StopPlaceName	**1:***	*+International Text*	Name of stop for passenger information.
	NameSuffix	0:*	*+International Text*	Name suffix, which can also be left out in case of space shortage, e.g.: "Exhibition Centre".
	PrivateCode	0:*	*+PrivateCode*	Private code for this stop in another key system.
	LocalityRef	0:1	*→Locality*	Reference to the locality, to which this stop belongs. See 7.5.1.
StopAttributes	*WheelchairAccessible*	0:1	*xs:boolean*	Overall wheelchair accessibility of this stop. Default is *false*.
	Lighting	0:1	*xs:boolean*	Specification for lighting this stop. Default is *false*.
	Covered	0:1	*xs:boolean*	Specification whether this stop offers weather protection (against rain, snow, storm etc.). Default is *false*.

Table 31: Description of structure StopPlaceStructure

7.5.4 LocalityStructure

LocalityStructure			*+Structure*	**Modelling of a locality/city.**
	LocalityCode	**1:1**	*→Locality*	Identifier of locality/city. See 7.5.1.
	LocalityName	**1:***	*+International Text*	Name of locality for passenger information.
	PrivateCode	0:*	*+PrivateCode*	Private code for this stopping point in another key system.
	ParentRef	0:1	*→Locality*	Reference to a parent locality, to which this locality belongs, e.g. relation of district to city. See 7.5.1.
Area	*Points*	3:*	*+GeoPosition*	Traverse line that describes the area of the locality.

Table 32: Description of structure LocalityStructure

7.5.5 PointOfInterestStructure

PointOfInterestStructure			*+Structure*	**Modelling an important point (POI).**
	PointOfInterestCode	**1:1**	*→PointOfInterest*	Identifier of POI.
	PointOfInterestName	**1:***	*+InternationalText*	Name of POI for passenger information.
	NameSuffix	0:*	*+InternationalText*	Name suffix, which can also be left out in case of space shortage, e.g.: "Exhibition Centre".
	PointOfInterestCategory	0:*	*+PointOfInterestCategory*	Categories that are assigned to this POI. See 7.5.6. If several categories are listed, they are sorted according to decreasing relevance.
	PrivateCode	0:*	*+PrivateCode*	Private code for this POI in another key system.
	LocalityRef	0:1	*→Locality*	Reference to the assigned locality, to which this POI belongs. See 7.5.1.

Table 33: Description of structure PointOfInterestStructure

7.5.6 PointOfInterestCategoryStructure

PointOfInterestCategoryStructure			*+Structure*	**Modelling of a POI category list.**
	OsmTag	1:*	*+OsmTag*	List of POI categories, defined by key-value pair as in OpenStreetMap[13]. See 7.5.7

Table 34: Description of structure PointOfInterestCategoryStructure

7.5.7 OsmTagStructure

OsmTagStructure			*+Structure*	**Modelling of a POI category.**
	Tag	**1:1**	*xs:NMTOKEN*	Name of OpenStreetMap tag (e.g. amenity, leisure, tourism, bike, ...)
	Value	**1:1**	*xs:NMTOKEN*	Value of OpenStreetMap tag (e.g. yes, hostel, charging_station, ...)

Table 35: Description of structure OsmTagStructure

7.5.8 PointOfInterestFilterStructure

PointOfInterestFilterStructure			*+Structure*	**Structure for filtering according to POI categories**
	Exclude	0:1	*xs:boolean*	Defines whether the following categories should be included (Exclude=*false*) or excluded (Exclude=*true*) uniquely during the POI search. Default is *false*.
	PointOfInterestCategory	**1:***	*+PointOfInterestCategory*	Identifier for POI categories. See 7.5.6. If several categories are listed, the categories are taken into consideration during the search using a logical "OR" (in case of Exclude=*false*) or using a logical "AND" (in case of Exclude=*true*).

Table 36: Description of structure PointOfInterestFilterStructure

[13] http://wiki.openstreetmap.org/wiki/DE:Map_Features

7.5.9 AddressStructure

AddressStructure			*+Structure*	**Modelling of an address.**
	AddressCode	**1:1**	*→Address*	Identifier of address. See 7.5.1.
	PrivateCode	0:*	*+PrivateCode*	Private code for this address in another key system.
	AddressName	**1:***	*+International Text*	Formatted address text for passenger information contains all the relevant address parts, for example: “Lister Str. 15, 30163 Hannover”.
	NameSuffix	0:*	*+International Text*	Name suffix, which can also be left out in case of space shortage, e.g.: “Exhibition Centre”.
Address Detail	*CountryName*	0:1	*xs:string*	Information about country.
	PostalCode	0:1	*xs:string*	Postal code.
	LocalityName	0:1	*xs:string*	Name of city or locality, in which the address is present.
	LocalityRef	0:1	*→Locality*	Reference to the city or locality, to which this address belongs. See 7.5.1.
	StreetName	0:1	*xs:string*	Name of street, in which the address is present, e.g. “Babarastr.”.
	HouseNumber	0:1	*xs:string*	House number including addition, e.g. “3-9, Block 6”. If empty, a) a crossing can be stated in *CrossingStreet* or b) the street can be meant as a whole.
	CrossingStreet	0:1	*xs:string*	Name of the crossing street.

Table 37: Description of structure AddressStructure

7.5.10 LocationStructure

LocationStructure				*+Structure*	**Basic model of a location (stopping point, stop, coordinate position, locality, POI or address).**
	a	*StopPoint*	-0:1	*+StopPoint*	Information about a stopping point. See 7.5.2.
	b	*StopPlace*		*+StopPlace*	Information about stop. See 7.5.3.
	c	*Locality*		*+Locality*	Information about a city/locality. See 7.5.4.
	d	*PointOfInterest*		*+PointOfInterest*	Information about a POI. See 7.5.5.
	e	*Address*		*+Address*	Information about an address. See 7.5.9.
	LocationName		**1:***	*+International Text*	Name of location.
	GeoPosition		**1:1**	*+GeoPosition*	Coordinate position. See 7.2.3.
	Attribute		*0:**	*+GeneralAttribute*	Attributes which are assigned to the location. See 7.4.10.
	Extension		0:1	*xs:anyType*	Extensions.

Table 38: Description of structure LocationStructure

7.5.11 LocationRefStructure

LocationRefStructure				*+Structure*	**Reference to a general location (stopping point, stop, coordinate position, locality or POI).**
	a	***StopPointRef***	**-1:1**	*→StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	b	***StopPlaceRef***		*→StopPlace*	Reference to a code for a stop. See 7.5.1.
	c	***GeoPosition***		*+GeoPosition*	Coordinate position.
	d	***LocalityRef***		*→Locality*	Reference to a code for a locality. See 7.5.1.
	e	***PointOfInterestRef***		*→PointOfInterest*	Reference to a code for a POI. See 7.5.1.
	f	***AddressRef***		*→Address*	Reference to an address. See 7.5.1.
	LocationName		**1:***	*+InternationalText*	Name of location.

Table 39: Description of structure LocationRefStructure

7.6 Trias_JourneySupport

The XML schema definition Trias_JourneySupport.xsd describes structures which describe the journeys on public transport vehicles. This includes description of a vehicle journey, information about departure and arrival events at stops as well as vehicle movements along the route.

The following sections describe the complex structures which are defined in Trias_JourneySupport.

7.6.1 ServiceViaPointStructure

ServiceViaPointStructure			*+Structure*	**Via point on the route.**
Stop Point	***StopPointRef***	**1:1**	*→StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+InternationalText*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+InternationalText*	Name suffix, which can also be left out in case of space shortage, e.g.: “opposite the main entrance”.
	PlannedBay	0:*	*+InternationalText*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+InternationalText*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction.
	DisplayPriority	0:1	*Priority*	Priority, with which this via-point should be displayed (e.g. if the place is narrow and not all the via points can be displayed).

Table 40: Description of structure ServiceViaPointStructure

7.6.2 ViaStructure

ViaStructure			*+Structure*	**Information about a via condition.**
	ViaPoint	**1:1**	*+LocationRef*	Reference to a via-point. See 7.5.11.
	DwellTime	0:*	*xs:duration*	Minimum dwell time at the via point required for the user.

Table 41: Description of structure ViaStructure

7.6.3 ServiceSectionStructure

ServiceSectionStructure			*+Structure*	**Properties of a journey together with the journey section, in which these properties apply.**
StopSeqInterval	*FromStopSeqNumber*	0:1	*xs:positiveInteger*	Route position number of stopping point, from which the properties are valid. If empty, then valid from the beginning of the route.
	ToStopSeqNumber	0:1	*xs:positiveInteger*	Route position number of stopping point, up to the properties are valid. If empty, then valid up to the end of the route.
LineIdentity	***LineRef***	**1:1**	*→Line*	Line ID. See 7.4.1.
	DirectionRef	**1:1**	*→Direction*	Direction ID See 7.4.1.
Service	***Mode***	**1:1**	*+Mode*	Type of public transport. See 7.3.4.
	PublishedLineName	**1:***	*+InternationalText*	Line number or name, as is known publicly.
	OperatorRef	0:1	*→Operator*	Operator-ID. See 7.4.1.
	RouteDescription	0:*	*+InternationalText*	Description of route, e.g. “right Rheinstrecke”.
	Via	0:*	*+ServiceViaPoint*	Important stops on the route. See 7.6.1.

Table 42: Description of structure ServiceSectionStructure

7.6.4 DatedServiceGroup

DatedServiceGroup			*+Group*	**Group for describing the journey of a line on a specific day.**
	OperatingDayRef	**1:1**	*→Operating-Day*	Operating day of the journey. See 7.4.1.
	VehicleRef	0:1	*→Vehicle*	Vehicle-ID. See 7.4.1.
Service-Journey	***JourneyRef***	**1:1**	*→Journey*	Journey-ID. See 7.4.1.
	ServiceSection	**1:***	*+ServiceSection*	Journey sections with properties. See 6.5.
	Attribute	0:*	*+ServiceAttribute*	Information and attributes (with classifications) about the journey. See 7.6.18.

Table 43: Description of group DatedServiceGroup

7.6.5 DatedJourneyStructure

DatedJourneyStructure			*+Structure*	**Journey on a specific day.**
	:::	1:1	*+DatedServiceGroup*	Journey of a line on the key date (see 7.6.4).
Service Origin	*OriginStopPointRef*	0:1	*→StopPoint*	ID of the first stopping point of the journey; starting stop. See 7.5.1.
	OriginText	0:*	*+InternationalText*	Name of the first stopping point of the journey; starting stop.
Service Destination	*DestinationStopPointRef*	0:1	*→StopPoint*	ID of the last stopping point of the journey; last stop. See 7.5.1.
	DestinationText	**1:***	*+InternationalText*	Name of the last stopping point of the journey; last stop or destination.
Service Status	*Unplanned*	0:1	*xs:boolean*	Specifies, whether the journey is an additional unplanned journey. Default setting is *false*.
	Cancelled	0:1	*xs:boolean*	Specifies whether this journey is completely omitted. Default setting is *false*.
	Deviation	0:1	*xs:boolean*	Specifies whether this journey takes a different route. Default setting is *false*.
	Occupancy	0:1	*manySeatsAvailable | fewSeatsAvailable | noSeatsAvailable | standingAvailable | full*	Occupancy state of the vehicle.
	SituationFullRef	0:*	*+SituationFullRef*	Reference to an error message. This message can be found in the context of the response (ResponseContext) or made known through other channels. See 7.8.2.

Table 44: Description of structure DatedJourneyStructure

7.6.6 ParallelServiceStructure

If a journey is planned on a section that is common with another journey (e.g. ICE between Cologne and Berlin, separation in Hamm), it is helpful to inform the passenger about the risk of boarding in the incorrect train compartment.

ParallelServiceStructure			*+Structure*	**Contains a section, on which another journey is common (e.g. in case of portion working) and the corresponding parallel journey.**
StopSeqInterval	*FromStopSeqNumber*	0:1	*xs:positiveInteger*	Route position number of stopping point, from which the parallel trip begins. If empty, then valid from the beginning of the route.
	ToStopSeqNumber	0:1	*xs:positiveInteger*	Route position number of stopping point, at which parallel journey ends. If empty, then valid up to the end of the route.
	Service	**1:1**	*+DatedJourney*	Parallel journey. See 7.6.5.

Table 45: Description of structure ParallelServiceStructure

7.6.7 TripLocationStructure

TripLocationStructure			*+Structure*	**Journey as current trip location of a passenger**
	OperatingDayRef	**1:1**	→*OperatingDay*	Operating day of the journey. See 7.4.1.
	JourneyRef	**1:1**	→*Journey*	Journey-ID. See 7.4.1.
LineId entity	***LineRef***	**1:1**	→*Line*	Line ID. See 7.4.1.
	DirectionRef	**1:1**	→*Direction*	Direction ID See 7.4.1.

Table 46: Description of structure TripLocationStructure

7.6.8 ServiceCallStructure

ServiceCallStructure			*+Structure*	**Contains information about the arrival or departure of a journey at a point (e.g. time).**
Servic eTime	***TimetabledTime***	**1:1**	*xs:dateTime*	Time as per timetable.
	RecordedAtTime	0:1	*xs:dateTime*	Actual time.
	EstimatedTime	0:1	*xs:dateTime*	Estimated time.
	EstimatedTimeLow	0:1	*xs:dateTime*	Lower limit for estimated time.
	EstimatedTimeHigh	0:1	*xs:dateTime*	Upper limit for estimated time.

Table 47: Description of structure ServiceCallStructure

7.6.9 CallAtStopStructure

CallAtStopStructure			*+Structure*	**Stop of a journey at a stopping point or a stop.**
Stop Point	***StopPointRef***	**1:1**	→*StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+InternationalText*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+InternationalText*	Name suffix, which can also be left out in case of space shortage, e.g.: “opposite the main entrance”.
	PlannedBay	0:*	*+InternationalText*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+InternationalText*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction.
	ServiceArrival	0:1	*+ServiceCall*	Information about arrival. See 7.6.8.
	ServiceDeparture	0:1	*+ServiceCall*	Information about departure. See 7.6.8.
StopCal lStatus	*StopSeqNumber*	0:1	*xs:positiveInteger*	Serial number of the stop on the route of the journey. Counted from the starting stop of the journey (as number 1).
	DemandStop	0:1	*xs:boolean*	Demand stop. Vehicle activates this stop only after advance notification. Default is *false*.
	UnplannedStop	0:1	*xs:boolean*	Stop which was not planned. Default is *false*.
	NotServicedStop	0:1	*xs:boolean*	The vehicle shall not stop contrary to the plan. Default is *false*.
	NoBoardingAtStop	0:1	*xs:boolean*	Passengers must not board at this stop of the journey. Default is *false*.
	NoAlightingAtStop	0:1	*xs:boolean*	Passengers must not alight at this stop of the journey. Default is *false*.
	SituationFullRef	0:*	*+SituationFullRef*	Reference to an error message. This message can be found in the ResponseContext of the response or made known through other channels. See 7.8.2.

Table 48: Description of structure CallAtStopStructure

7.6.10 DatedCallAtLocationStructure

DatedCallAtLocationStructure			*+Structure*	**Vehicle call at a general location on a specific day**
DatedJourneyRef	***JourneyRef***	**1:1**	*→Journey*	Journey-ID. See 7.4.1.
	OperatingDayRef	**1:1**	*→Operating-Day*	Operating day of the journey. See 7.4.1.
LineDirection	***LineRef***	**1:1**	*→LineCode*	Reference to a line. See 7.4.1.
	DirectionRef	0:1	*→DirectionCode*	Reference to a line direction. See 7.4.1.
	OperatorRef	0:1	*→Operator*	Operator-ID. See 7.4.1.
	CallLocation	**1:1**	*+LocationRef*	Generalised departure point. Normally, a departure point is a stop. But it can also be an address or coordinate in case of flexible lines or “Area Dial-A-Ride” services. See 7.5.11.
	ServiceArrival	0:1	*+ServiceCall*	Information about arrival. See 7.6.8.
	ServiceDeparture	0:1	*+ServiceCall*	Information about departure. See 7.6.8.
StopCallStatus	*StopSeqNumber*	0:1	*xs:positiveInteger*	Serial number of the stop on the route of the journey. Counted from the starting stop of the journey (as number 1).
	DemandStop	0:1	*xs:boolean*	Demand stop. Vehicle activates this stop only after advance notification. Default is *false*.
	UnplannedStop	0:1	*xs:boolean*	Stop which was not planned. Default is *false*.
	NotServicedStop	0:1	*xs:boolean*	The vehicle shall not stop contrary to the plan. Default is *false*.
	NoBoardingAtStop	0:1	*xs:boolean*	Passengers must not board at this stop of the journey. Default is *false*.
	NoAlightingAtStop	0:1	*xs:boolean*	Passengers must not alight at this stop of the journey. Default is *false*.

Table 49: Description of structure DatedCallAtLocationStructure

7.6.11 ContinuousServiceStructure

ContinuousServiceStructure				*+Structure*	**A passenger movement using a continuous, non-timetabled service**
	a	***ContinuousMode***	**-1:1**	*walk | demandResponsive | replacementService*	Modality for continuous transport operations.
	b	***IndividualMode***		*walk | cycle | taxi | self-drive-car | others-drive-car | motorcycle |*	Public transport modality for individual transport.
	a	*:::*	-0:1	*+DatedServiceGroup*	Description of a public transport option on key date (see 7.6.4).
	b	*SharingService*		*+SharingService*	Description of a mobility option with rental vehicles (see 7.4.7).
Service Origin	*OriginStopPointRef*		0:1	*→StopPoint*	ID of the first stopping point of the journey; starting stop. See 7.5.1.
	OriginText		0:*	*International-Text*	Name of the first stopping point of the journey; starting stop.
Service Destination	*DestinationStopPointRef*		0:1	*→StopPoint*	ID of the last stopping point of the journey; last stop. See 7.5.1.
	DestinationText		0:*	*International-Text*	Name of the last stopping point of the journey; last stop or destination.
	SituationFullRef		0:*	*+SituationFullRef*	Reference to an error message. This message can be found in the context of the response (ResponseContext) or made known through other channels.

Table 50: Description of structure ContinuousServiceStructure

7.6.12 VehiclePositionStructure

VehiclePositionStructure			*+Structure*	**Geographic and logical position of a vehicle.**
	GeoPosition	0:1	*+GeoPosition*	Geographic position (see 7.2.3).
	Progress	0:1	*Not yet operated | Operation finished | At stop | Between stops*	Logical position based on the stop sequence in the timetable.
	Bearing	0:1	*AbsoluteBearing*	Compass direction in degrees, along which the vehicle moves (see 7.2.1).
	ProgressBetweenStops	0:1	*+ProgressBetweenStops*	Position between the stop last visited and the current position (see 7.6.13).

Table 51: Description of structure VehiclePositionStructure

7.6.13 ProgressBetweenStopsStructure

ProgressBetweenStopsStructure			*+Structure*	**Position between the stop last visited and the current position.**
	LinkDistance	0:1	*Distance*	Total distance in metres between the last and the next stop.
	Percentage	0:1	*Percent*	Percentage that the vehicle has covered along the link distance (LinkDistance)

Table 52: Description of structure ProgressBetweenStopsStructure

7.6.14 LegTrackStructure

LegTrackStructure			*+Structure*	**Container for track sections along a partial link.**
	TrackSection	**1:***	*+TrackSection*	One or more track sections. See 7.6.15.

Table 53: Description of structure LegTrackStructure

7.6.15 TrackSectionStructure

TrackSectionStructure			*+Structure*	**A track section as leg of the journey.**
	TrackStart	0:1	*+LocationRef*	Beginning (location) of the track section. See 7.5.11.
	TrackEnd	0:1	*+LocationRef*	End (location) of the track section. See 7.5.11.
Projection	***Position***	**2:***	*+GeoPosition*	Geographic projection of the track section as polygonal line. See 7.2.3.
	RoadName	0:1	*xs:string*	Name of the road, on which track section lies.
	Duration	0:1	*xs:duration*	Duration the passenger needs to travel along this track section.
	Length	0:1	*LengthType*	Length of the track section.
	Extension	0:1	*xs:anyType*	Extensions.

Table 54: Description of structure TrackSectionStructure

7.6.16 LocationContextStructure

LocationContextStructure				*+Structure*	**Specification of a location and accessibility options for a user via individual routes.**
	a	***LocationRef***	**-1:1**	*+LocationRef*	Specification of spatial situation (see 7.5.11).
	b	***TripLocation***		*+TripLocation*	Stop location in a (moving) vehicle (see 7.6.6).
	a	***DepArrTime***	**-1:1**	*xs:dateTime*	Planned arrival or departure time at the location identified in *Location* or *TripLocation.*
	b	***TimeAllowance***		*xs:duration*	Extra time needed before reaching/after leaving this location. Useful only, if several location contexts are used in parallel, and no explicit date/time at the locations is known.
	IndividualTransportOptions		0:*	*+IndividualTransportOptions*	Options stated by the user, how he/she could access/leave the location by individual transport (see 7.3.2).

Table 55: Description of structure LocationContextStructure

Elements of the type LocationContextStructure are primarily used to describe the start or end context of a passenger. For example, elements of this type define the start and end location within the journey planning service. Here, the implementation of the search algorithm is responsible to map the location details (e.g. a coordinate) to the internal elements (e.g. nodes and edges) of the search network.

For this purpose the IndividualTransportOptions specify the options for the user to access/leave the stop via public transport. Normally, this is a footpath but it can also be a bicycle, car or taxi. When selecting a bicycle, the user must select whether he/she wants to carry the bike on public vehicles or not. This option could lead to different journey planning results in some cases. When selecting a car, the user must choose between self-drive and others-drive. In the first case, journey planning must include a route to a parking place. In the second case, however, a place to stop and get off the car would be sufficient.

7.6.17 AbstractResponseContextStructure

AbstractResponseContextStructure			*+Structure*	**Basic structure for response context. Objects can be saved here which occur multiple times and can be replaced by references to the context.**
Locations	*Location*	0:*	*+Location*	Modelling locations (see 7.5.10).
Situations	*Situation*	0:*	*+siri:PtSituationElement*	SIRI modelling of an event or an error (see 7.8.1).

Table 56: Description of structure AbstractResponseContextStructure

7.6.18 ServiceAttributeStructure

ServiceAttributeStructure			*+Structure (derived from GeneralAttributeStructure, see 7.4.10)*	**Definition of attributes and information which are valid only in parts of a link.**
	Scope	0:1	*onRide* | *atStop* | *atBoardOnly* | *atAlightOnly* | *atBoardAndAlight*	Defines for what an attribute or information is valid. An attribute can be valid during the journey (e.g. “wheelchair space”), at each stop (e.g. “boarding aid available”), only at boarding stops (e.g. “ride only upon booking”), only at alighting stops (e.g. “stops only after button-press”) or at boarding and alighting stops (e.g. “Mind your step”).
StopSeqInterval	*FromStopSeqNumber*	0:1	*xs:positiveInteger*	Route position number of stopping point, from which the attribute is valid. If empty, then valid from the beginning of the route.
	ToStopSeqNumber	0:1	*xs:positiveInteger*	Route position number of stopping point, up to which the attribute is valid. If empty, then valid up to the end of the route.

Table 57: Description of structure ServiceAttributeStructure

7.6.19 PassengerAccessibilityStructure

PassengerAccessibilityStructure			*+Structure*	**Structure for defining special needs and restrictions of passengers.**
Base-TripMobilityFilter	*NoSingleStep*	0:1	*xs.boolean*	Defines whether the user can use steps. Default is *false*.
	NoStairs	0:1	*xs.boolean*	Defines whether the user can use stairs. Default is *false*.
	NoEscalator	0:1	*xs.boolean*	Defines whether the user can use an escalator. Default is *false*.
	NoElevator	0:1	*xs.boolean*	Defines whether the user can use the elevator. Default is *false*.
	NoRamp	0:1	*xs.boolean*	Defines whether the user can use a ramp. Default is *false*.
TripMobilityFilter	*LevelEntrance*	0:1	*xs.boolean*	Defines whether the user needs a level entrance to board and exit vehicles. For this purpose, even a lift to the vehicle or platform is sufficient. If the level entrance is necessary, this parameter is set to *true.* Default is *false*.
	BikeTransport	0:1	*xs.boolean*	Defines whether the user wants to carry a bicycle on public vehicles. If yes, this parameter is set to *true.* Default is *false*.
	WalkSpeed	0:1	*OpenPercent*	Change in standard walking speed in percent. The value 100 is set by default. Values less than 100
Assistance	*BoardingAssistance*	0:1	*xs:boolean*	Specifies whether the user needs help of the driving or station personnel while boarding. Default is *false*.
	AlightingAssistance	0:1	*xs:boolean*	Specifies whether the user needs help of the driving or station personnel while alighting. Default is *false*.
PassengerProfile	*WheelchairUser*	0:1	*xs:boolean*	Passenger uses a wheelchair. Default is *false*.
	WalkingFrame	0:1	*xs:boolean*	Passenger uses a walking frame. Default is *false*.
	WalkingStick	0:1	*xs:boolean*	Passenger uses a walking stick. Default is *false*.
	WalkingImpaired	0:1	*xs:boolean*	Passenger cannot walk. Default is *false*.
	Pram	0:1	*xs:boolean*	Passenger carries a pram. Default is *false*.
	HeavyLuggage	0:1	*xs:boolean*	Passenger carries heavy luggage. Default is *false*.
	VisuallyImpaired	0:1	*xs:boolean*	Passenger is visually impaired. Default is *false*.
	HearingImpaired	0:1	*xs:boolean*	Passenger is hearing-impaired. Default is *false*.
	ReadingImpaired	0:1	*xs:boolean*	Passenger is reading-impaired. Default is *false*.

Table 58: Description of structure PassengerAccessibilityStructure

7.7 Trias_FacilitySupport

The XML schema definition Trias_FacilitySupport.xsd provides structure definitions from the SIRI FM service which can be used for sending messages to infrastructure facilities and vehicle facilities. The structures defined here aim at encapsulating the import of the SIRI schema at one single place within TRIAS and at creating an abstraction level which allows extension without having to change the SIRI definitions.

The following sections describe the complex structures which are defined in Trias_FacilitySupport.

7.7.1 siri:CommonFacilityGroup

The group CommonFacilityGroup is defined in SIRI in schema file siri_facilities-v1.2.xsd. It is mentioned here only for reasons of completeness and ease of comprehension.

siri:CommonFacilityGroup			*+Group*	**Classification of common facility and infrastructure features (as per TPEG pti_table 23).**
	FareClassFacility	0:*	*unknown | firstClass | secondClass | thirdClass | economy- Class | businessClass*	Fare classes.
	TicketingFacility	0:*	*unknown | ticketMachines | ticketOffice | ticketOnDemandMachines | ticketSales | mobileTicketing | ticketCollection | centralReservations | local-Tickets | nationalTickets | international-Tickets*	Facilities for purchasing tickets.
	NuisanceFacility	0:*	*unknown | smoking | noSmoking | mobile- PhoneUse-Zone | mobilePhone-FreeZone*	Common areas.
	MobilityFacility	0:*	*unknown | suitableFor-WheelChairs | lowFloor | boardingAssistance | stepFreeAccess | tactile-PatformEdges | onboardAssistance | unaccompaniedMinorAssistance | audioInformation | visualIinformation | displaysForVisuallyImpaired | audio-ForHea- ringImpaired*	Facility properties for mobility-impaired passengers.
	PassengerInform ation- Facility	0:*	*unknown | nextStopIndicator | stopAnnouncements | passengerInformationDisplay | audioInformation | visualIinformation | tactilePlatformEdges | tactileInformation | walkingGuidance | journeyPlanning | lost-Found | informationDesk | interactiveKiosk-Display | printedPublicNotice*	Facilities for passenger information.
	PassengerComms Facility	0:*	*unknown | faccomms_1 | passengerWifi | telephone | audioServices | videoServices | businessServices | internet | postoffice | letterbox*	Communication facilities for passengers.
	RefreshmentFacility	0:*	*unknown | restaurantService | snacksService | trolley | bar | foodNotAvailable | beveragesNotAvailable |bistro | foodVendingMachine | beverageVendingMachine*	Provision of refreshments, food and drinks.
	AccessFacility	0:*	*unknown | lift | escalator | travellator | ramp | stairs | shuttle | narrowEntrance | barrier | palletAccess_lowFloor | validator*	Access properties of the stops or vehicles.
	SanitaryFacility	0:*	*unknown | toilet | noToilet | shower | wheelchairAcccessToilet | baby-Change*	Sanitary facilities.
	LuggageFacility	0:*	*unknown | bikeCarriage | baggageStorage | leftLuggage | porterage | baggageTrolleys*	Facilities for luggage transport or storage.

Table 59: Description of group siri:CommonFacilityGroup

7.7.2 siri:StopFacilityGroup

The group StopFacilityGroup is defined in SIRI in schema file siri_facilities-v1.2.xsd. It is mentioned here only for reasons of completeness and ease of comprehension.

siri:StopFacilityGroup			*+Group*	**Classification of facility and infrastructure features at stops (as per TPEG pti_table 23).**
CommonFacilityGroup	:::	0:*	*siri:CommonFacilityGroup*	General facility properties. See 7.7.1.
	AssistanceFacility	0:*	*unknown |* *police | firstAid | sosPoint | specificAssis- tance | unaccompaniedMinorAssistance | boardingAssistance*	Facilities for those seeking aid.
	HireFacility	0:*	*unknown | carHire | motorCycleHire | cycleHire | taxiDeviceHire | recreation-*	Hire and rental services.

Table 60: Description of group siri:StopFacilityGroup

7.7.3 siri:ServiceFacilityGroup

The group ServiceFacilityGroup is defined in SIRI in schema file siri_facilities-v1.2.xsd. It is mentioned here only for reasons of completeness and ease of comprehension.

siri:ServiceFacilityGroup			*+Group*	**Classification of facility and infrastructure features in vehicles (as per TPEG pti_table 23).**
CommonFacilityGroup	:::	0:*	*siri:CommonFacilityGroup*	General facility properties. See 7.7.1.
	AccommodationFacility	0:*	*unknown | sleeper | couchette | specialSeating | freeSeating | recliningSeats | babyCompartment | familyCarriage*	Accommodation types.

Table 61: Description of group siri:ServiceFacilityGroup

7.7.4 siri:AllFacilitiesGroup

The group AllFacilitiesGroup is defined in SIRI in schema file siri_facilities-v1.2.xsd. It is mentioned here only for reasons of completeness and ease of comprehension.

siri:AllFacilitiesGroup			*+Group*	**Comprehensive group with all the classifications of facility and infrastructure features (as per TPEG pti_table 23).**
ServiceFacility Group	:::	0:*	*siri:ServiceFacilityGroup*	Facility properties of vehicles. See 7.7.3.
	AssistanceFacility	0:*	*unknown | police | firstAid | sosPoint | specificAssistance | unaccompaniedMinorAssistance | boardingAssistance*	Facilities for those seeking aid.
	HireFacility	0:*	*unknown | carHire | motorCycleHire | cycleHire | taxi | recreationDeviceHire*	Hire and rental services.

Table 62: Description of group siri:AllFacilitiesGroup

7.8 Trias_SituationSupport

The XML schema definition Trias_SituationSupport.xsd provides structure definitions from the SIRI SX service which can be used for sending error and event messages. The structures defined here aim at encapsulating the import of the SIRI schema at one single place within TRIAS and at creating an abstraction level which allows extension without having to change the SIRI definitions.

The following sections describe the complex structures which are defined in Trias_SituationSupport.

7.8.1 SituationsStructure

SituationsStructure			*+Structure*	**Container for structured description of a situation in public transport or road transport, such as an incident in public transport or road transport or an event with effects on traffic.**
	PtSituation	0:*	*siri:PtSituationElement*	Encapsulation of SIRI structure definitions for public transport events, see (CEN, TS 15531 part 5, 2011).
	RoadSituation	0:*	*siri:RoadSitationElement*	Encapsulation of SIRI structure definitions for individual transport events, see (CEN, TS 15531 part 5, 2011).

Table 63: Description of structure SituationsStructure

7.8.2 SituationFullRefStructure

SituationFullRefStructure			*+Structure (derived From siri:SituationFull RefStructure)*	**Reference to a situation description.**
SituationFullIdentity	*VersionCountryRef*	0:1	*ifopt:CountryRef*	References the country, in order to disambiguate *ParticipantRef,* if necessary
	ParticipantRef	**1:1**	*ParticipantRef*	Unique ID of interface partner (see 0). Provides namespace for ID of situation.
	SituationNumber	**1:1**	*EntryQualifier*	Unique ID of situation.
SituationUpdate Identity	*VersionCountryRef*	0:1	*ifopt:CountryRef*	References the country, in order to disambiguate *ParticipantRef,* if necessary
	UpdateParticipantRef	0:1	*ParticipantRef*	Unique ID of interface partner (see 0). Provides namespace for ID of situation.
	Version	0:1	*SituationVersion*	Version number of update regarding situation. Can be omitted during initial notification.

Table 64: Description of structure SituationFullRefStructure

7.9 Trias_RequestSupport

The XML schema definition Trias_RequestSupport.xsd contains a range of basic types and structures that can be used in SIRI message exchange procedures for TRIAS services.

7.9.1 Simple types

The following simple types are defined:

Type name	***Basic type***	**Description**
DataVersionType	*xs:NMTOKEN*	Data type for specifying data version.
CalcTimeType	*xs:integer*	Data type for calculating time in milliseconds.
SignatureType	*xs:string*	Data type for signatures.
CertificateIdType	*xs:NMTOKEN*	Data type for certificate IDs.

Table 65: List of simple type definitions in Trias_RequestSupport.xsd.

The following sections describe the complex structures which are defined in Trias_RequestSupport.

7.9.2 AbstractTriasServiceRequestStructure

AbstractTriasServiceRequestStructure			*+Structure*	**Basic structure for all direct requests (without subscription)**
siri:ContextualisedRequest	*ServiceRequestContext*	0:1	*+siri:ServiceRequestContext*	General message properties which are typically configured and do not have to be exchanged on request. Also see (CEN, TS 15531 Part 2, 2011), section 6.1.2.
	RequestTimestamp	**1:1**	*xs:dateTime*	Timestamp of request.
RequestorEndpoint	*Address*	0:1	*siri:EndpointAddress*	Address to which the response should be sent. Can also be sent via RequestorRef from the configuration.
	RequestorRef	**1:1**	→*siri:ParticipantCode*	ID of requester.
	MessageIdentifier	0:1	*siri:MessageQualifier*	Any unique ID, with which this message can be referenced.
ServiceRequestContext	*DataVersion*	0:1	*DataVersion*	Data version which should be used by the server during processing.
	Language	0:*	*xs:language*	Preferred languages, in which texts in the response should be written.
Message-Integrity-Properties	*Signature*	0:1	*Signature*	Signature of message.
	CertificateId	0:1	*CertificateId*	Certificate ID for checking the message.
ServiceRequestContext	*Extension*	0:1	*xs:anyType*	Extensions.

Table 66: Description of structure AbstractTriasServiceRequestStructure

7.9.3 AbstractTriasSubscriptionRequestStructure

AbstractTriasSubscriptionRequestStructure			*+Structure*	**Basic structure for all requests for a subscription facility.**
siri:AbstractSubscriptionRequest	***RequestTimestamp***	**1:1**	*xs:dateTime*	Timestamp of request.
RequestorEndpoint	*Address*	0:1	*siri:EndpointAddress*	Address to which the response should be sent. Can also be sent via RequestorRef from the configuration.
	RequestorRef	**1:1**	*→siri:ParticipantCode*	ID of requester.
	MessageIdentifier	0:1	*siri:MessageQualifier*	Any unique ID, with which this message can be referenced.
SubscriberEndpoint	*ConsumerAddress*	0:1	*siri:EndpointAddress*	Address, to which the messages arising within the framework of the subscription should be sent. This information can be omitted if the ConsumerAddress is identical to RequestorEndpoint:Address.
	SubscriptionFilterIdentifier	0:1	*xs:NMTOKEN*	ID of a pre-configured filter, to which the messages should be subjected for this subscription.
siri:AbstractSubscriptionRequest	*SubscriptionContext*	0:1	*siri:SubscriptionContext*	General subscription properties which are typically configured and do not have to be explicitly stated. Also see (CEN, TS 15531 Part 2, 2011), section 7.1.1.2.
SubscriptionRequestContext	*DataVersion*	0:1	*DataVersion*	Data version which should be used by the server during processing.
	Language	0:*	*xs:language*	Preferred languages, in which texts in the response should be written.
MessageIntegrityProperties	*Signature*	0:1	*Signature*	Signature of message.
	CertificateId	0:1	*CertificateId*	Certificate ID for checking the message.
SubscriptionRequestContext	*Extension*	0:1	*xs:anyType*	Extensions.

Table 67: Description of structure AbstractTriasSubscriptionRequestStructure

7.9.4 AbstractTriasResponseStructure

AbstractTriasResponseStructure			*+Structure*	**Basic structure for all responses.**
siri:ProducerResponse	***RequestTimestamp***	**1:1**	*xs:dateTime*	Timestamp of response.
siri:ProducerResponseEndpoint	*ProducerRef*	0:1	→*siri:ParticipantCode*	ID of responding participant.
	Address	0:1	*siri:EndpointAddress*	Address to which a message receipt acknowledgement should be sent. Can also be sent via RequestorRef from the configuration.
	ResponseMessageIdentifier	0:1	*siri:MessageQualifier*	Any unique ID, with which this message can be referenced.
	RequestMessageRef	0:1	→*siri:MessageQualifier*	Reference to request message which has triggered this response message.
ResponseStatus	*Status*	0:1	*xs:boolean*	Indicator whether the overall request could be processed completely successfully. Default is *true*.
	ErrorCondition	0:1	*siri:ErrorCondition*	SIRI error states which concern the processing of the request as a whole. Also see (CEN, TS 15531 Part 2, 2011), section 5.7.
	MoreData	0:1	*xs:boolean*	Indicator whether there still are further updates which could be called. Default is *false*.
ServiceResponseContext	*DataVersion*	0:1	*DataVersion*	Data version which has been used by the server during processing.
	Language	**1:1**	*xs:language*	Standard language, in which the text of the response is written, unless specified otherwise per element (see 7.2.2).
	CalcTime	0:1	*CalcTime*	Calculating time for processing the response.
MessageIntegrityProperties	*Signature*	0:1	*Signature*	Signature of message.
	CertificateId	0:1	*CertificateId*	Certificate ID for checking the message.
ServiceResponseContext	*Extension*	0:1	*xs:anyType*	Extensions.

Table 68: Description of structure AbstractTriasResponseStructure

7.10 Trias_FaresSupport

The XML schema definition Trias_FaresSupport.xsd contains a range of basic types and structures that can be used for fare calculation of a journey.

7.10.1 Simple types

The following simple types are defined:

Type name	***Basic type***	**Description**
FaresAuthorityCodeType	*xs:NMTOKEN*	Code for a fare authority or a company fare, e.g. “VVS” or “DBAG”.
FareZoneCodeType	*xs:NMTOKEN*	Code for a tariff zone in a fare authority or a company tariff.
TicketCodeType	*xs:NMTOKEN*	Code for a ticket. Unique within a tariff area or company tariff.
TravellerCardCodeType	*xs:NMTOKEN*	Code for a traveller card, e.g. “Bahn-Card50” or “BahnCard25First”.
TravelClassEnumeration	*all | first | second | third | business | economy*	Travel class
VatRateEnumeration	*no | full | half | mixed | unknown*	Enumeration of possible VAT rates.
PassengerCategoryEnumeration	*Adult | Child | Senior | Youth | Disabled*	Categorisation of passengers with respect to tariff.
TicketUsageValidityTypeEnumeration	*singleTrip | returnTrip | connectingPass | multiRidePass | carnet | dayPass | 24HourPass | weeklyPass | weekendPass | monthlyPass | halfYearPass | annualPass | seasonTicket | profileMembership | openEnded | other*	Enumeration of possible validities of ticket usages (see CEN TS16614-3 [NeTEx]) extended by connectingPass, multiRidePass, 24HourPass, halfYearPass.

Table 69: List of simple type definitions in Trias_FaresSupport.xsd

In order to be able to understand the codes of fare authorities, tariff zones etc. across the system, certain agreements must be made. They are described in chapter 5.14, 5.15, 5.16.

The following sections describe the defined complex structures of the Trias_FaresSupport:

7.10.2 FareZoneStructure

FareZoneStructure			*+Structure*	**Model of a tariff zone with well-known name.**
	FareZoneRef	**1:1**	*→FareZoneCode*	Code for a tariff zone (see 7.10.1).
	FareZoneText	**1:1**	*xs:string*	Name of tariff zone for passengers.

Table 70: Description of structure FareZoneStructure

7.10.3 FareZoneListInAreaStructure

FareZoneListInAreaStructure			*+Structure*	**List of tariff zones based on a tariff authority.**
FaresAuthority	***FaresAuthorityRef***	**1:1**	*→FaresAuthorityCode*	Code for a fare authority or a company fare (see 7.10.1).
	FaresAuthorityText	**1:1**	*xs:string*	Description or name of fare authority.
	FareZone	**1:***	*+FareZone*	One or more tariff zones (see 7.10.2).

Table 71: Description of structure FareZoneListInAreaStructure

7.10.4 BookingInfoStructure

BookingInfoStructure			*+Structure*	**Description of a booking option for the requested object.**
	BookingAgencyName	0:*	*+InternationalText*	Name of booking agency (contractual partner).
	BookingUrl	0:1	*xs:anyURI*	URL for online-booking.
	InfoUrl	0:1	*xs:anyURI*	URL for information pages.
	PhoneNumber	0:1	*PhoneNumber*	Telephone number for booking (see 7.2.1).
	BookingDeadline	0:1	*xs:duration*	Minimum waiting time for booking before the journey begins.
	Extension	0:1	*xs:anyType*	Extensions.

Table 72: Description of structure BookingInfoStructure

7.10.5 TicketStructure

TicketStructure			*+Structure*	**Modelling of a ticket and associated information.**
	TicketId	**1:1**	*→TicketCode*	Unique ticket ID (see 7.10.1).
	TicketName	**1:1**	*xs:string*	Name of ticket.
FaresAuthority	***FaresAuthorityRef***	**1:1**	*→FaresAuthorityCode*	Code for a fare authority or a company fare (see 7.10.1).
	FaresAuthorityText	**1:1**	*xs:string*	Description or name of fare authority.
TicketPrice	*Price*	0:1	*xs:decimal*	Ticket price as decimal number.
	NetPrice	0:1	*xs:decimal*	Net ticket price as decimal number for calculation purposes.
	Currency	0:1	*xs:NMTOKEN*	Currency code as per ISO 4217, e.g. "EUR" or "GBP".
	VatRate	0:1	*VatRateEnumeration*	VAT rate (see 7.10.1). Default setting is *unknown*.
TariffLevel	*TariffLevel*	0:1	*xs:string*	Tariff level (example from Nürnberg "10" or "10+T")
	TariffLevelLabel	0:*	*+InternationalText*	Name for tariff levels in this context (example from Nürnberg "price level", "tariff level")
TicketValidity	*TravelClass*	0:1	*TravelClassEnumeration*	Travel class, for which the ticket is valid (see 7.10.1).
	RequiredCard	0:*	*→TravellerCardCode*	One or more traveller cards which are necessary to be able to purchase or use this ticket (see 7.10.1).
	ValidFor	0:*	*PassengerCategoryEnumeration*	Passenger categories which may use this ticket (see 7.10.1).
	ValidityDuration	0:1	*xs:duration*	Maximum validity duration of the ticket after purchase or validation.
	ValidityDurationText	0:*	*+InternationalText*	Description of validity duration.
	ValidityFareZones	0:1	*+FareZoneListInArea*	Geographic validity of the ticket stated with the help of a list of tariff zones, for which the ticket is valid.
	ValidityAreaText	0:*	*+InternationalText*	Description of validity zone.
	UsageValidityType	0:1	*TicketUsageValidityTypeEnumeration*	Usage validity of a ticket e.g. single trip, day pass, 24 hours pass, carnet ticket, multi pass, …
TicketBooking	*InfoUrl*	0:*	*+WebLink*	URL for information pages for this ticket (see 7.2.4).
	SaleUrl	0:*	*+WebLink*	URL for online purchase options in order to purchase this ticket (see 7.2.4).
	BookingInfo	0:*	*+BookingInfo*	Description of booking options (see 7.10.4).
	Extension	0:1	*xs:anyType*	Extensions.

Table 73: Description of structure TicketStructure

7.10.6 TripFaresResultStructure

TripFaresResultStructure			*+Structure*	**Summarises the result data for tariff information about a connection (or parts of a connection).**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on this tariff information. Refer to the following table for possible values. Also see 7.4.2.
TripLeg Range	*FromTripLegIdRef*	0:1	*xs:NMTOKEN*	Reference to a leg of the trip as beginning of validity of this tariff information.
	ToTripLegIdRef	0:1	*xs:NMTOKEN*	Reference to a leg of the trip as end of validity of this tariff information.
	PassedZones	0:1	*+FareZoneListInArea*	The traversed tariff zones on this section of the trip (see 7.10.3).
	Ticket	0:*	*+Ticket*	Tickets which are valid on this section of the trip (see 7.10.5).
	StaticInfoURL	0:*	*+WebLink*	URL for information pages (see 7.2.4).

Table 74: Description of structure TripFaresResultStructure

In ErrorMessage, the following error states can appear:

Error code	Error description
FARES_OUTOFAREA	The route found in the trip information leaves the tariff area.
FARES_JOURNEYNOTPERMITTED	A mode of transport used in the trip information is not permissible for the tariff.
FARES_ADDITIONALCHARGES	Additional fare must most likely have to be paid (e.g. toll surcharges or reservation fees).
FARES_ADDITIONALTICKETS	Additional tickets are necessary because a suitable ticket could not be ascertained for all modes of transport.
FARES_ROUTENOTFEASIBLE	A ticket cannot be ascertained because the route in the trip information does not comply with the tariff rules (e.g. due to round trips, diversions or exceeding the permissible overall duration).
FARES_ALREADYCOVERED	This connection (or a part thereof stated in *TripLegRange*) can be used with driving authorisation which has been stated in the request.

Table 75: List of error states in TripFaresResult

7.10.7 FaresPassengerStructure

FaresPassengerStructure				*+Structure*	**Profile of a passenger for tariff determination.**
	a	***Age***	**-1:1**	*xs:nonNegativeInteger*	Age of the passenger.
	b	***PassengerCategory***		*PassengerCategoryEnumeration*	Category of the passenger which can be assigned to the passenger (see 7.10.1).
	TravellerCard		0:*	→*TravellerCardCode*	One or more traveller cards which the passenger has purchased and can use (see 7.10.1).
	a	*ZonesAlreadyPaid*	-0:1	*+FareZoneListInArea*	List of tariff zones, for which the passenger already has a valid ticket (see 7.10.3).
	b	*OwnedTicket*	-0:*	→*TicketCode*	One or more IDs of tickets which the passenger already has purchased and which the passenger can use for the journey (or at least for parts thereof).

Table 76: Description of structure FaresPassengerStructure

By stating the element ZonesAlreadyPaid or OwnedTicket, it can be expressed that the passenger already has travel authorisations, e.g. in the form of tickets (such as a monthly pass or a job ticket). In this case, the server should try to determine whether these authorisations are already sufficient to allow the passenger to use this connection or alternatively recommend tickets for the remaining parts for which the existing authorisations are not sufficient. For parts of the connection, for which additional ticket is not needed, error state FARES_ALREADYCOVERED (from Table 75) is stated in TripFaresResult (see 7.10.6).

7.10.8 FaresParamStructure

FaresParamStructure			*+Structure*	**Parameters for tariff determination.**
FaresDataFilter	*FareAuthorityFilter*	0:*	→*FaresAuthorityCode*	Codes for fare authorities or company tariffs which should be taken into account (see 7.10.1).
	PassengerCategory	0:*	*PassengerCategoryEnumeration*	Passenger categories which should be taken into account. (see 7.10.1).
	TravelClass	0:1	*TravelClassEnumeration*	Travel class which should be taken into account (see 7.10.1).
	Traveller	0:*	*+FaresPassenger*	Number of passengers, for which the fare should be determined (see 7.10.7).

Table 77: Description of structure FaresParamStructure

8 Ortsinformationsdienst

8.1 Beschreibung

Der Ortsinformationsdienst umfasst vier Funktionalitäten, die in der VDV-Schrift 431-1 als getrennte Dienste beschrieben werden

- Start-/Ziel-Identifikation bei Eingabe einer Zeichenkette,
- Objektinformationsdienst zum Abrufen aller Ortsobjekte,
- Geografischer Kontextdienst zum Abrufen von Ortsobjekten in einem Kartenausschnitt,
- Koordinaten-zu-Adressdienst zum Abrufen der nächsten Adresse für gegebene Koordinaten.

Diese Funktionalitäten werden durch Abstraktion in einem einzigen Dienst gebündelt. Dadurch entstehen auch weitere Einsatzmöglichkeiten des Dienstes.

Beispielsweise (aber nicht abschließend):

- Abruf der nächsten Haltestelle(n) für gegebene Koordinaten.
- Ortsabhängiges Patternmatching einer Zeichenkette durch Berücksichtigung von gleichzeitig übergebenen Koordinaten.

In der XML-Schema-Definition *Trias_Locations.xsd* werden Datentypen und Strukturen definiert, die für den Ortsinformationsdienst verwendet werden.

8 Location information service

8.1 description

The location information service comprises four functionalities which are described in the VDV guideline 431-1 as a separate service

- Start/destination identification when entering a character string,
- Object information service for calling all location objects,
- Geographic context service for calling location objects in a map section,
- Coordinates-to-address service for calling the next address for the given coordinates.

These functionalities are bundled together into one service through abstraction. This results in more application options of the service.

Examples (but not conclusive):

- Call of the next stop(s) for the given coordinates.
- Location-dependent pattern-matching of a character string by keeping into account the simultaneously transmitted coordinates.

Data types and structures are defined in the XML schema definition Trias_Locations.xsd which are used for the location information service.

8.2 Simple data types

The following simple types are defined:

Type name	***Basic type***	**Description**
LocationTypeEnumeration	*stop | address | poi | coord | locality*	Type of a location object.
LocationUsageEnumeration	*origin | destination | via*	Intended use of a location object.

Table 78: List of simple type definitions in Trias_Locations.xsd

8.3 Request structures

Location objects are requested with the help of an element LocationInformationRequest of the type LocationInformationRequestStructure.

8.3.1 LocationInformationRequestStructure

LocationInformationRequestStructure				*+Structure*	**Summarises the data of location object request.**
	a	***InitialInput***	**-1:1**	*+InitialLocationInput*	Input data for an initial location information request. See 8.3.2.
	b	***LocationRef***		*+LocationRef*	Reference to a location object which should be further refined. In case of hierarchically organised location objects, it can make sense to carry out location identification in several stages. In the process, an initial request to the location information service generates a list of “rough” location objects (e.g. streets) which must be further refined (e.g. to house number ranges, see *Complete* in chapter 0). The “rough” objects are displayed to the user and the user selects one of them. In order to further refine them, its reference is sent to the location information service. See 7.5.11.
	Restrictions		0:1	*+LocationParam*	Additional request parameters. See 8.3.7.
	Extension		0:1	*xs:anyType*	Extensions.

Table 79: Description of structure LocationInformationRequestStructure

8.3.2 InitialLocationInputStructure

InitialLocationInputStructure			*+Structure*	**Summarises the request parameters which require an initial search of location objects.**
	LocationName	0:1	*xs:string*	Input string which should serve as a pattern for the location objects to be found. If specified, the more similar the name of the string is, location objects should be preferred all the more. If *GeoPosition* is specified at the same time, the service must weigh both requests reasonably.
	GeoPosition	0:1	*+GeoPosition*	Geographic position, near which the location objects to be found should lie. If specified, the nearer the geographic position, the more preferred such location objects should be. If *LocationName* is specified at the same time, the service must weigh both requests reasonably. See 7.2.3.
	GeoRestriction	0:1	*+GeoRestrictions*	Geographic filter. If specified, all the location objects found must be subject to this filter. See 8.3.3.

Table 80: Description of structure InitialLocationInputStructure

8.3.3 GeoRestrictionsStructure

GeoRestrictionsStructure				*+Structure*	**Defines a geographic filter.**
	a	***Circle***	**-1:1**	*+GeoCircle*	The filter is defined by a circle. See 8.3.4.
	b	***Rectangle***	**-1:1**	*+GeoRectangle*	The filter is defined by a rectangle. See 8.3.5.
	c	***Area***	**-1:1**	*+GeoArea*	The filter is defined by a polygon. See 8.3.6.

Table 81: Description of structure GeoRestrictionsStructure

8.3.4 GeoCircleStructure

GeoCircleStructure			*+Structure*	**Defines a geographic circle.**
	Centre	**1:1**	*+GeoPosition*	Centre of the circle. See 7.2.3.
	Radius	**1:1**	*Distance*	Radius of the circle in metres.

Table 82: Description of structure GeoCircleStructure

8.3.5 GeoRectangleStructure

GeoRectangleStructure			*+Structure*	**Defines a geographic rectangle.**
	UpperLeft	**1:1**	*+GeoPosition*	Upper left corner of the rectangle. See 7.2.3.
	LowerRight	**1:1**	*+GeoPosition*	Lower right corner of the rectangle. See 7.2.3.

Table 83: Description of structure GeoRectangleStructure

8.3.6 GeoAreaStructure

GeoAreaStructure			*+Structure*	**Defines a geographic polygon.**
	PolylinePoint	**3:***	*+GeoPosition*	Corners of the polygon. See 7.2.3.

Table 84: Description of structure GeoAreaStructure

8.3.7 LocationParamStructure

LocationParamStructure			*+Structure*	**Summarises request parameters which are used in the location information service.**
LocationDataFilter	*Type*	0:*	*stop | address | poi | coord | locality*	Location object types permitted. If types of the location objects are specified, only those must be returned which are of one of the specified types. If not, all object types are permitted.
	Usage	0:1	*origin | destination | via*	Use of a location object. If specified, it informs the service, what the searched location object should be used as. The location information service must then return only those objects which are approved for the stated use.
	PtModes	0:1	*+PtModeFilter*	Modes of transport permitted. If specified, only those location objects must be returned, on which all types of transport can be used which is subject to the filter. This automatically excludes all the non-stops. See 7.3.5.
	OperatorFilter	0:1	*+Operator Filter*	The search is limited to location objects which are operated/not operated by certain companies (see 7.4.4).
	LocalityRef	0:*	→*LocalityCode*	Localities permitted. If specified, only those location objects must be returned which are assigned to at least one of the given localities. See 7.5.1.
	PointOfInterestFil-ter	0:1	*+PointOfInterestFilter*	Facilitates a POI search limited to certain POI categories (see 7.5.6).
LocationPolicy	*NumberOfResults*	0:1	*xs:positive Integer*	Number of maximum location objects that can be returned. The service can return lesser objects if it is practical or if the service is overworked. If more objects fulfil the request (e.g. if all the objects should be called), the maximum number of objects that can be transmitted in a call can be restricted with the help of this parameter. A location information service must be in the position to return at least 500 location objects in a response.
	ContinueAt	0:1	*xs:nonNegativeInteger*	If specified, this parameter instructs the service regarding the number of objects to be skipped in the response. If all the suitable objects could not be sent in a call of location objects, this service notifies this in its response in the field *ContinueAt* (see 8.4.1). In order to call other objects, the request to the location information service is precisely repeated, where this parameter is specified by filling the value from the last service response.
	IncludePtModes	0:1	*xs:boolean*	Informs the service to return the available modes of transport at stops. Default is *false*.

Table 85: Description of structure LocationParamStructure

8.4 Response structures

The result of an object information request is sent via an element LocationInformationResponse of the type LocationInformationResponseStructure.

8.4.1 LocationInformationResponseStructure

LocationInformationResponseStructure			*+Structure*	**Summarises the result data for a location information request.**
	ContinueAt	0:1	*xs:nonNegativeInteger*	In a subsequent call to skip location objects. If set, the service indicates that there are more location objects that match the request which are not included in the response. If the call is repeated and in the process the parameter *ContinueAt* is set to the value sent here (see 8.3.7), the service provides the following location objects.
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	LocationResult	0:*	*+LocationResult*	Location object results found. The location objects must be sorted according to the degree of matching with the input data, i.e. the first is the best suitable object. See 8.4.2.

Table 86: Description of structure LocationInformationResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
LOCATION_NORESULTS	No location objects could be found using the input data.
LOCATION_UNSUPPORTEDTYPE	Only those object types were requested which are not supported by the service.
LOCATION_UNSUPPORTEDCOMBINATION	The combination of the input data used (string, coordinates, geo-restriction) is not supported by the service.
LOCATION_NOREFINEMENT	The location object specified could not be refined.
LOCATION_USAGEIGNORED	The intended use was ignored.
LOCATION_UNSUPPORTEDPTMODES	The service does not support limitation of mode of transport.
LOCATION_UNSUPPORTEDLOCALITY	The service does not support limitation due to localities.

Table 87: List of error states in LocationInformationResponse

8.4.2 LocationResultStructure

LocationResultStructure			*+Structure*	**Result structure for a location object.**
	Location	**1:1**	*+Location*	Actual location object. See 7.5.10.
	Complete	**1:1**	*xs:boolean*	Specifies whether the location object is already completely differentiated or whether it must be further refined so that, for example, it can be used for a TripRequest. Incomplete location objects must be differentiated by another LocationInformationRequest. (See *LocationRef* in chapter 8.3.1)
	Probability	0:1	*xs:float*	Probability that this location object matches the one searched. Is specified using a value between 0 and 1.
	Mode	0:*	*+Mode*	List of modes which call at the location object. Should be filled only in case of stops and only if requested in the request. See 7.3.4.

Table 88: Description of structure LocationResultStructure

9 Dienst Verbindungsauskunft

9.1 Beschreibung

Dieser Dienst berechnet intermodale Verbindungen von einem Startpunkt zu einem Zielpunkt. Dabei werden diverse Benutzerpräferenzen berücksichtigt.

In der XML-Schema-Definition *Trias_Trips.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Verbindungsauskunft verwendet werden.

9 Trip information service

9.1 Description

This service provides intermodal trip information from a starting point to a destination. It takes into consideration diverse user preferences.

Data types and structures are defined in the XML schema definition Trias_Trips.xsd which are used for the trip information service.

9.2 Request structures

An intermodal trip information is requested via an element TripRequest of the type TripRequestStructure.

9.2.1 TripRequestStructure

TripRequestStructure			*+Structure*	**Summarises the request data for trip information.**
	Origin	**1:***	*+LocationContext*	Location data for point of departure. See 7.6.16.
	Destination	**1:***	*+LocationContext*	Location data for destination. See 7.6.16.
	Via	0:*	*+Via*	One or more via-locations. The via-locations specified must be reached in the specified sequence. The server may replace a via-stop by an equivalent stop. See 7.6.2.
	NotVia	0:*	*+NotVia*	Stops or stopping points, via which the trip must not pass. See 9.2.4.
	NoChangeAt	0:*	*+NoChangeAt*	Stops or stopping points, at which the trip must not provide transfer. See 9.2.5.
	Params	0:1	*+TripParam*	Parameters which can affect the search and return values. See 9.2.2.

Table 89: Description of structure TripRequestStructure

The elements Origin and Destination are generally simple. Several Origin or Destination elements should be used only in case several starting points or destinations, which imply separate departure or arrival time, are to be defined. In this case, the server selects the point optimal for the overall trip. The choice of the optimal origin or destination point can depend on the time and hence can change with every trip found.

9.2.2 TripParamStructure

TripParamStructure				*+Structure*	**Summarises the request data for trip information.**
TripDataFilter		*PtModeFilter*	0:1	*+PtModeFilter*	Filter according to modes of transport. See 7.3.5.
		LineFilter	0:1	*+LineDirectionFilter*	Permitted lines (if necessary, refined in directions). See 7.4.6.
		OperatorFilter	0:1	*+OperatorFilter*	Filter according to transport companies. See 7.4.4.
Base-TripMobilityFilter		*NoSingleStep*	0:1	*xs.boolean*	Defines whether the user can use steps. Default is *false*.
		NoStairs	0:1	*xs.boolean*	Defines whether the user can use stairs. Default is *false*.
		NoEscalator	0:1	*xs.boolean*	Defines whether the user can use an escalator. Default is *false*.
		NoElevator	0:1	*xs.boolean*	Defines whether the user can use the elevator. Default is *false*.
		NoRamp	0:1	*xs.boolean*	Defines whether the user can use a ramp. Default is *false*.
TripMobilityFilter		*LevelEntrance*	0:1	*xs.boolean*	Defines whether the user needs a level entrance to board and exit vehicles. For this purpose, even a lift to the vehicle or platform is sufficient. If the level entrance is necessary, this parameter is set to *true.* Default is *false*.
		BikeTransport	0:1	*xs.boolean*	Defines whether the user wants to carry a bicycle on public vehicles. If yes, this parameter is set to *true.* Default is *false*.
		WalkSpeed	0:1	*OpenPercent*	Change in standard walking speed in percent. The value 100 is set by default. Values less than 100 represent slower speed and greater than 100 represent faster speed.
BaseTripPolicy	*a*	*NumberOfResults*	-0:1	*xs:positiveInteger*	Number of trip information results which the user expects as a minimum.
	b	*:::*	-0:1	*NumberOfResultsGroup*	Specification of the desired trips before/after the stated time at start or end (see 9.2.3).
		IgnoreRealtimeData	0:1	*xs:boolean*	If this parameter is set, real-time data or error information should not be considered in the trip search but only target trip data. Default is *false*.
		ImmediateTripStart	0:1	*xs:boolean*	If this parameter is set, the trip to be searched should directly begin at the start situation specified. Optimisation of the departure time at the start is generally not necessary according to the rule “Start as late as possible only if the exact arrival time is ensured at the destination”. Default is *false*.
TripPolicy		*InterchangeLimit*	0:1	*xs:positiveInteger*	Number of maximum permissible interchanges.
		AlgorithmType	0:1	*fastest* | *minChanges* | *leastWalking* | *leastCost*	Type of target function, according to which the algorithm should optimise the trip.
		ItModesToCover	0:*	*IndividualModesEnumeration*	For every individual transport mode (see 7.3.1) in this list, a separate mono-modal trip should be found – in addition to the intermodal trips.
		MultiPointType	0:1	*anyPoint* | *eachOrigin* | *eachDestination*	Whether a solution for any one of multiple origin/destination points is sufficient or a distinct solution for each of the origin/destination points has to be found.

BaseTripContentFilter	*IncludeTrackSections*	0:1	*xs:boolean*	Specifies whether TrackSection element (see 7.6.15) should be output in the result for a detailed geographic description of the route. Default is *false*.
	IncludeLegProjection	0:1	*xs:boolean*	Specifies whether the detailed geographic route should be output in the result as a coordinate sequence. Default is *false*.
	IncludeTurnDescription	0:1	*xs:boolean*	Specifies whether route information should be output in the result with turn recommendations. Default is *false*.
	IncludeAccessibility	0:1	*xs:boolean*	Specifies whether information about barrier freedom should be output in the result. Default is *false*.
	IncludeEstimatedTimes	0:1	*xs:boolean*	Specifies whether information about real-time situation should be output in the result. Default is *false*.
	IncludeSituationInfo	0:1	*xs:boolean*	Specifies whether textual real-time messages should be output in the result. Default is *false*.
TripContentFilter	*IncludeIntermediateStops*	0:1	*xs:boolean*	Specifies whether intermediate stops should be output in the result. Default is *false*.
	IncludeFares	0:1	*xs:boolean*	Specifies whether fare information should be output in the result. Default is *false*.
	IncludeOperatingDays	0:1	*xs:boolean*	Specifies whether information about operating days should be output in the result. Default is *false*.
	FaresParam	0:1	*+FaresParam*	Parameters for tariff determination (see 7.10.8).
	Extension	0:1	*xs:anyType*	Extensions.

Table 90: Description of structure TripParamStructure

9.2.3 NumberOfResultsGroup

NumberOfResultsGroup			*+Group*	**Specification of the number of desired trips before and after the stated time at start or end. This group cannot be used if a time is prescribed at the start AND at the end.**
	NumberOfResultsBefore	**1:1**	*xs:nonNegativeInteger*	Number of desired trips before the specified time.
	NumberOfResultsAfter	**1:1**	*xs:nonNegativeInteger*	Number of desired trips after the specified time.

Table 91: Description of group NumberOfResultsGroup

9.2.4 NotViaStructure

NotViaStructure				*+Structure*	**Information about a not-via condition. This type of condition stops a trip that is not allowed to pass through a specified stop or stopping point.**
	a	***StopPointRef***	**-1:1**	→*StopPoint*	Reference to a not-via stopping point. See 7.5.1.
	b	***StopPlaceRef***		→*StopPlace*	Reference to a not-via stop. See 7.5.1.

Table 92: Description of structure NotViaStructure

9.2.5 NoChangeAtStructure

NoChangeAtStructure				*+Structure*	**Information about a no-change condition. This type of condition prevents that transfer must take place at the specified stop or stopping point in trip information.**
	a	***StopPointRef***	**-1:1**	→*StopPoint*	Reference to a stopping point. See 7.5.1.
	b	***StopPlaceRef***		→*StopPlace*	Reference to a stop. See 7.5.1.

Table 93: Description of structure NoChangeAtStructure

9.3 Response structures

The result of an intermodal trip request is sent via an element TripResponse of the type TripResponseStructure.

9.3.1 TripResponseStructure

TripResponseStructure			*+Structure*	**Summarises the result data for a trip information.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. Also see 7.4.2.
	TripResponseContext	0:1	*+TripResponseContext*	Containers for data, which appear multiple times in the response and are referenced. See 9.3.2.
	TripResult	0:*	*+TripResult*	Container for trip information. See 9.3.3.

Table 94: Description of structure TripResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
TRIP_NOTRIPFOUND	A trip could not be found with regard to the start locations and destinations stated, the desired departure and arrival time and keeping in mind the specified parameters.
TRIP_ORIGINUNKNOWN	The specified location (address, stop etc.) for the start of the trip is unknown.
TRIP_DESTINATIONUNKNOWN	The specified location (address, stop etc.) for the destination of the trip is unknown.
TRIP_VIAUNKNOWN	One of the via-points specified is unknown.
TRIP_NOTVIAUNKNOWN	One of the not-via-stops specified is unknown.
TRIP_NOCHANGEATUNKNOWN	One of the no-change-stops specified is unknown.
TRIP_NOORIGIN	Origin has not been specified.
TRIP_NODESTINATION	A destination has not been specified.
TRIP_ORIGINDESTINATIONIDENTICAL	Start and destination are identical.
TRIP_DATETIMEERROR	Date and/or time are incomprehensible.
TRIP_DEPARTUREAFTERARRIVAL	The desired departure time at all the start points is after the desired arrival time at al destination points.
TRIP_DATEOUTOFRANGE	NO journey data available for the requested date.

Table 95: List of error states in TripResponse

9.3.2 TripResponseContextStructure

TripResponseContextStructure	***+Structure*** *(derived from AbstractResponseContextStructure)*	Containers for data, which appear multiple times in the response and are referenced. See 7.6.17.

Table 96: Description of structure TripResponseContextStructure

9.3.3 TripResultStructure

TripResultStructure			*+Structure*	**Summarises the result data for an individual intermodal trip information.**
	ResultId	**1:1**	*xs:NMTOKEN*	ID of the result for subsequent referencing or debugging purposes.
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on trip result. Refer to the following table for possible values. Also see 7.4.2.
	Trip	**1:1**	*+Trip*	Data about an intermodal trip. See 9.3.4.
	TripFares	0:*	*+TripFaresResult*	Ticket and fare information about the trip as a whole or parts of the trip (see 7.10.6).

Table 97: Description of structure TripResultStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
TRIP_ORIGINEQUIVALENT	The desired starting stop was replaced by an equivalent stop.
TRIP_DESTINATIONEQUIVALENT	The desired destination was replaced by an equivalent stop.
TRIP_VIAEQUIVALENT	A desired via-stop was replaced by an equivalent stop.
TRIP_REALTIMEINCOMPLETE	Real-time data is not available for at least one mode in this trip.
TRIP_ITTIMEEXTENDED	The maximum time specified in individual transport (mostly walking or bicycling) was extended by the system as otherwise no trip can be found.
TRIP_ITMODECHANGED	The maximum time specified in individual transport (mostly footpath or bicycle) was extended by the system as otherwise no trip can be found. Usually, this is a change from walking to taxi.
TRIP_INCONVENIENTWAITING	The trip includes a long waiting time.

Table 98: List of error states in TripResult

9.3.4 TripStructure

TripStructure			*+Structure*	**Data about an individual intermodal trip.**
	TripId	**1:1**	*xs:NMTOKEN*	ID of the trip for subsequent referencing or debugging purposes.
	Duration	**1:1**	*xs:duration*	Total duration of trip.
	StartTime	**1:1**	*xs:dateTime*	Start time of trip.
	EndTime	**1:1**	*xs:dateTime*	End time of trip.
	Interchanges	**1:1**	*xs:nonNegativeInteger*	Number of necessary interchanges.
	Distance	0:1	*Distance*	Total distance of the trip as length of the route to be covered.
	TripLeg	**1:***	*+TripLeg*	Leg or legs of this trip. See 9.3.5.
OperatingDays	*OperatingDays*	0:1	*+OperatingDays*	Operating days for this trip. See 7.4.8.
	OperatingDaysDescription	0:*	*+InternationalText*	Human-readable description of the operating days, e.g. “Monday to Friday” or “Sunday and holidays”.
	SituationFullRef	0:*	*+SituationFullRef*	Reference to an error message. This message can be found in the *TripResponseContext* (see 9.3.2) or made known through other channels. See 7.8.2.
	Extension	0:1	*xs:anyType*	Extensions.

Table 99: Description of structure TripStructure

9.3.5 TripLegStructure

TripLegStructure				*+Structure*	**Trip leg**
		LegId	**1:1**	*xs:NMTOKEN*	Leg ID for later referencing. Unique within *TripResult*.
	a	***TimedLeg***	**-1:1**	*+TimedLeg*	Attribute of trip leg as timetabled trip leg. See 9.3.6.
	b	***InterchangeLeg***		*+InterchangeLeg*	Attribute of trip leg as interchange between modes of transport. See 9.3.7.
	c	***ContinuousLeg***		*+ContinuousLeg*	Attribute of trip leg as movement with a continuously available mode. See 9.3.8.

Table 100: Description of structure TripLegStructure

9.3.6 TimedLegStructure

TimedLegStructure			*+Structure*	**Includes a timetabled trip leg.**
	LegBoard	**1:1**	*+LegBoard*	Beginning (stopping point) of trip leg. See 0.
	LegIntermediates	0:*	*+LegIntermediate*	Intermediate traversed stopping points on the trip leg between *LegBoard* and *LegAlight*. See 0.
	LegAlight	**1:1**	*+LegAlight*	End (stopping point) of trip leg. See 0.
	Service	**1:1**	*+DatedJourney*	Specification of mode of transport such as line, mode type etc. See 7.6.5
OperatingDays	*OperatingDays*	0:1	*+OperatingDays*	Operating days for this trip. See 7.4.8.
	OperatingDaysDescription	0:*	*+InternationalText*	Human-readable description of the operating days, e.g. “Monday to Friday” or “Sunday and holidays”.
	LegTrack	0:1	*+LegTrack*	Detailed geometrical course. See 7.6.14.
	ParallelService	0:*	*+ParallelService*	Parallel trips (e.g. in case of portion working). See 7.6.6.
	Extension	0:1	*xs:anyType*	Extensions.

Table 101: Description of structure TimedLegStructure

9.3.7 InterchangeLegStructure

InterchangeLegStructure				*+Structure*	**Includes a trip leg which is an interchange between two modes.**
	a	***InterchangeMode***	**-1:1**	*walk | parkAndRide | bikeAndRide | carHire | bikeHire | protectedConnection | guaranteedConnection | remainInVehicle | changeWithinVehicle | checkIn | checkOut*	Classification of change processes
	b	***ContinuousMode***		*walk | demandResponsive | replacementService*	Modality for continuous transport operations.
	LegStart		**1:1**	*+LocationRef*	Beginning (location) of this trip leg. See 7.5.11.
	LegEnd		**1:1**	*+LocationRef*	End (location) of this trip leg. See 7.5.11.
TimeWindow	*TimeWindowStart*		0:1	*xs:dateTime*	Earliest time for the start of this trip leg.
	TimeWindowEnd		0:1	*xs:dateTime*	Latest time for the start of this trip leg.
InterchangeDuration	***Duration***		**1:1**	*xs:duration*	Total interchange time necessary.
	WalkDuration		0:1	*xs:duration*	Walking part of the total interchange time.
	BufferTime		0:1	*xs:duration*	Buffer time of the total interchange time. Check-in times are required in many modes, e.g. flight, journeys or even high speed trains.
	LegDescription		0:*	*+International Text*	Description of interchange.
	Length		0:1	*LengthType*	Length of interchange.
	Attribute		0:*	*+GeneralAttribute*	Information and attributes (with classifications) about the interchange. See 7.4.10.
	NavigationPath		0:1	*+NavigationPath*	Detailed information about the geometric course, route and accessibility. See 9.3.12.
	SituationFullRef		0:*	*+SituationFull Ref*	Reference to an error message. This message can be found in the Trip*ResponseContext* (see 9.3.2) or made known through other channels.
	Extension		0:1	*xs:anyType*	Extensions.

Table 102: Description of structure InterchangeLegStructure

9.3.8 ContinuousLegStructure

ContinuousLegStructure			*+Structure*	**Includes a trip leg which is not timetabled (e.g. walking).**
	LegStart	**1:1**	*+LocationRef*	Beginning (location) of this trip leg. See 7.5.11.
	LegEnd	**1:1**	*+LocationRef*	End (location) of this trip leg. See 7.5.11.
	Service	**1:1**	*+ContinuousService*	Information about “mode of transport” (e.g. walking). See 0.
TimeWindow	*TimeWindowStart*	0:1	*xs:dateTime*	Earliest time for the start of this trip leg.
	TimeWindowEnd	0:1	*xs:dateTime*	Latest time for the start of this trip leg.
	Duration	**1:1**	*xs:duration*	Duration of this trip leg.
	LegDescription	0:*	*+InternationalText*	Description of this trip leg.
	Length	0:1	*LengthType*	Length of this trip leg.
	LegTrack	0:1	*+LegTrack*	Detailed (geometrical) course. See 7.6.14
	NavigationPath	0:1	*+NavigationPath*	Detailed information about the geometric course, route and accessibility. See 9.3.12.
	SituationFullRef	0:*	*+SituationFullRef*	Reference to an error message. This message can be found in the Trip*ResponseContext* (see 9.3.2) or made known through other channels. See 7.8.2.
	Extension	0:1	*xs:anyType*	Extensions.

Table 103: Description of structure ContinuousLegStructure

9.3.9 LegBoardStructure

LegBoardStructure			*+Structure*	**Describes the boarding situation in a mode.**
Stop Point	***StopPointRef***	**1:1**	→*StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+International Text*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+International Text*	Name suffix, which can also be left out in case of space shortage, e.g.: “opposite the main entrance”.
	PlannedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction.
	ServiceArrival	0:1	*+ServiceCall*	Information about arrival. See 7.6.8.
	ServiceDeparture	**1:1**	*+ServiceCall*	Information about departure. See 7.6.8.
	MeetsViaRequest	0:1	*xs:boolean*	This stop fulfils one of the via-conditions specified in the request. Default is *false*.
StopCallStatus	*StopSeqNumber*	0:1	*xs:positiveInteger*	Serial number of the stop on the route of the journey. Counted from the starting stop of the journey (as number 1).
	DemandStop	0:1	*xs:boolean*	Demand stop. Vehicle activates this stop only after advance notification. Default is *false*.
	UnplannedStop	0:1	*xs:boolean*	Stop which was not planned. Default is *false*.
	NotServicedStop	0:1	*xs:boolean*	The vehicle shall not stop contrary to the plan. Default is *false*.
	NoBoardingAtStop	0:1	*xs:boolean*	Passengers must not board at this stop of the journey. Default is *false*.
	NoAlightingAtStop	0:1	*xs:boolean*	Passengers must not alight at this stop of the journey. Default is *false*.

Table 104: Description of structure LegBoardStructure

9.3.10 LegAlightStructure

LegAlightStructure			*+Structure*	**Describes the alighting situation from a mode of transport.**
Stop Point	***StopPointRef***	**1:1**	→*StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+International Text*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+International Text*	Name suffix, which can also be left out in case of space shortage, e.g.: “opposite the main entrance”.
	PlannedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction.
	ServiceArrival	**1:1**	*+ServiceCall*	Information about arrival. See 7.6.8.
	ServiceDeparture	0:1	*+ServiceCall*	Information about departure. See 7.6.8.
	MeetsViaRequest	0:1	*xs:boolean*	This stop fulfils one of the via-conditions specified in the request. Default is *false*.
StopCallStatus	*StopSeqNumber*	0:1	*xs:positiveInteger*	Serial number of the stop on the route of the journey. Counted from the starting stop of the journey (as number 1).
	DemandStop	0:1	*xs:boolean*	Demand stop. Vehicle activates this stop only after advance notification. Default is *false*.
	UnplannedStop	0:1	*xs:boolean*	Stop which was not planned. Default is *false*.
	NotServicedStop	0:1	*xs:boolean*	The vehicle shall not stop contrary to the plan. Default is *false*.
	NoBoardingAtStop	0:1	*xs:boolean*	Passengers must not board at this stop of the journey. Default is *false*.
	NoAlightingAtStop	0:1	*xs:boolean*	Passengers must not alight at this stop of the journey. Default is *false*.

Table 105: Description of structure LegAlightStructure

9.3.11 LegIntermediateStructure

LegIntermediateStructure			*+Structure*	**Intermediate stop on a trip leg.**
Stop Point	***StopPointRef***	**1:1**	*→StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+International Text*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+International Text*	Name suffix, which can also be left out in case of space shortage, e.g.: “opposite the main entrance”.
	PlannedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+International Text*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction.
	ServiceArrival	**1:1**	*+ServiceCall*	Information about arrival. See 7.6.8
	ServiceDeparture	**1:1**	*+ServiceCall*	Information about departure. See 7.6.8.
	MeetsViaRequest	0:1	*xs:boolean*	This stop fulfils one of the via-conditions specified in the request. Default is *false*.
StopCallStatus	*StopSeqNumber*	0:1	*xs:positiveInteger*	Serial number of the stop on the route of the journey. Counted from the starting stop of the journey (as number 1).
	DemandStop	0:1	*xs:boolean*	Demand stop. Vehicle activates this stop only after advance notification. Default is *false*.
	UnplannedStop	0:1	*xs:boolean*	Stop which was not planned. Default is *false*.
	NotServicedStop	0:1	*xs:boolean*	The vehicle shall not stop contrary to the plan. Default is *false*.
	NoBoardingAtStop	0:1	*xs:boolean*	Passengers must not board at this stop of the journey. Default is *false*.
	NoAlightingAtStop	0:1	*xs:boolean*	Passengers must not alight at this stop of the journey. Default is *false*.

Table 106: Description of structure LegIntermediateStructure

9.3.12 NavigationPathStructure

NavigationPathStructure			*+Structure*	Container for route descriptions.
	NavigationSection	**1:***	*+NavigationSection*	One or more track sections. See 9.3.13

Table 107: Description of structure NavigationPathStructure

9.3.13 NavigationSectionStructure

NavigationSectionStructure			*+Structure*	Description of a route section, with information of geographic embedding, turn information and route condition (accessibility for disabled persons).
	TrackSection	0:1	*+TrackSection*	Geographic description of track section. See 7.6.15.
	TurnDescription	0:*	*+InternationalText*	Description of manoeuvre to be carried out. The contents of *Manoeuvre*, *TurnAction* and *TrackSection.RoadName* should be described in text form.
	Manoeuvre	0:1	*origin | destination | continue | keep | turn | leave | enter*	Coding of manoeuvre to be carried out.
	TurnAction	0:1	*sharp left | left | half left | straight on | half right | right | sharp right | uturn*	Coding of turning processes.
	DirectionHint	0:*	*+InternationalText*	Textual direction information for better understanding of the following track section, e.g. “Follow the signs to Hamburg”.
	Bearing	0:1	*AbsoluteBearing*	Compass direction which is accepted after the manoeuvre. It does not refer to the entire route section.
	SituationFullRef	0:*	*+SituationFullRef*	References to error messages. These messages can be found in the *TripResponseContext* (see 9.3.2) or made known through other channels. See 7.8.2.
	AccessPath	0:1	*+AccessPath*	Description of accessibility of path. See 9.3.14.

Table 108: Description of structure NavigationSectionStructure

9.3.14 AccessPathStructure

AccessPathStructure			*+Structure*	**Description of accessibility of a path.**
	Transition	0:1	*up | down | level | upAndDown | downAndUp*	Indication whether the path is level or leads upwards/downwards.
	AccessFeatureType	0:1	*lift | stairs | seriesOfStairs | escalator | ramp | foot-path*	Path type.
	Count	0:1	*xs:positiveInteger*	Number of how often the path type occurs.
	FacilityRef	0:*	→*Facility*	Reference to facility used.

Table 109: Description of structure AccessPathStructure

10 Dienst Abfahrtstafeln

10.1 Beschreibung

Dieser Dienst informiert über Ankünfte und Abfahrten von ÖV-Fahrten an Haltestellen für einen bestimmten Zeitpunkt oder Zeitraum. In den Parametern kann eine bestimmte Haltestelle oder Haltestellen im Umkreis eines Ortes angefragt werden, dabei können weitere Einschränkungen vorgegeben werden, die sich als Filter auf die Ergebnisse auswirken.

In der XML-Schema-Definition *Trias_StopEvents.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Abfahrtstafeln verwendet werden.

10 Stop event service

10.1 Description

This service provides information about arrivals and departures of public transport services to stops for a requested time or period of time. A certain stop in the locality can be requested in the parameters. In the process, restrictions can be set that filter the result contents accordingly.

Data types and structures are defined in the XML schema definition Trias_StopEvents.xsd which are used for the stop event service.

10.2 Request structures

A stop event (or arrival board) is requested with the help of an element StopEventRequest of the type StopEventRequestStructure.

10.2.1 StopEventRequestStructure

StopEventRequestStructure			*+Structure*	**Summarises the request data for departure or arrival board.**
	Location	**1:1**	*+LocationContext*	Location data for departure/arrival board. See 7.6.16.
	Params	0:1	*+StopEventParam*	Specific request parameters. See 10.3.2.

Table 110: Description of structure StopEventRequestStructure

10.2.2 StopEventParamStructure

StopEventParamStructure			*+Structure*	**Summarises the request parameters which control the calculation of a departure or arrival board.**
StopEventDataFilter	*PtModeFilter*	0:1	*+PtModeFilter*	Modes of transport permitted. See 7.3.5.
	LineFilter	0:1	*+LineDirectionFilter*	Permitted lines (if necessary, refined in directions) See 7.4.6.
	OperatorFilter	0:1	*+OperatorFilter*	Transport companies permitted. See 7.4.4.
StopEventPolicy	*NumberOfResults*	0:1	*xs:positiveInteger*	Maximum number of departure/arrival results which should be returned in the response.
	TimeWindow	0:1	*xs:duration*	Time window, in which the departure/arrival results should be returned in the response. Is calculated from the time stated in *LocationContext.*
	StopEventType	0:1	*departure* | *arrival* | *both*	Specifies whether the departure or arrival results or both should be returned. Default is *departure*.
StopEventContentFilter	*IncludePreviousCalls*	0:1	*xs:boolean*	Specifies whether the previous stops should be stated for every journey. Default is *false*.
	IncludeOnwardCalls	0:1	*xs:boolean*	Specifies whether the subsequent stops should be stated for every journey. Default is *false*.
	IncludeOperatingDays	0:1	*xs:boolean*	Specifies whether the operating days of the trips should be stated. Default is *false*.
	IncludeRealtimeData	0:1	*xs:boolean*	Controls whether real-time data should be considered and displayed. Default is *false*.

Table 111: Description of structure StopEventParamStructure

10.3 Response structures

The result of a stop event request is sent via an element StopEventResponse of the type StopEventResponseStructure

10.3.1 StopEventResponseStructure

StopEventResponseStructure			*+Structure*	**Summarises the result data for departure or arrival board request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	StopEventResponseContext	0:1	*+StopEventResponseContext*	Containers for data, which appear multiple times in the response and are referenced. See 10.3.2.
	StopEventResult	0:*	*+StopEventResult*	Container for a departure or arrival event. See 10.3.3.

Table 112: Description of structure StopEventResponseStructure

In ErrorMessage, the following error states can appear:

STOPEVENT_DATEOUTOFRANGE	NO journey data available for the requested date.
STOPEVENT_LOCATIONUNKNOWN	The location (address, stop etc.), for which the departure/arrival board has been requested, is unknown.
STOPEVENT_LOCATIONUNSERVED	The location (address, stop etc.), for which the departure/arrival board has been requested, is not serviced by public transport.
STOPEVENT_NOEVENTFOUND	Departure/arrival was not found in compliance with the given options in the period in question.

Table 113: List of error states in StopEventResponse

10.3.2 StopEventResponseContextStructure

StopEventResponseContextStructure	***+ Structure*** *(derived from AbstractResponseContextStructure)*	Containers for data which appear multiple times in the response and are referenced. See 7.6.17.

Table 114: Description of structure StopEventResponseContextStructure

10.3.3 StopEventResultStructure

StopEventResultStructure			*+Structure*	**Summarises the result data for an individual departure or arrival event.**
	ResultId	**1:1**	*xs:NMTOKEN*	ID of the result for subsequent referencing or debugging purposes.
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on this departure/arrival event . Refer to the following table for possible values. Also see 7.4.2.
	StopEvent	**1:1**	*+StopEvent*	Data about a departure or arrival event. See 10.3.4.

Table 115: Description of structure StopEventResultStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
STOPEVENT_LASTSERVICEOFTHISLINE	This departure/arrival is the last of this line at the stop on this operating day.
STOPEVENT_NOREALTIME	Real-time data or prognosis is not available for this departure/arrival.

Table 116: List of error states in StopEventResult

10.3.4 StopEventStructure

StopEventStructure			*+Structure*	**Data about an individual departure or arrival event.**
	PreviousCall	0:*	*+CallAtNearStop*	Departure/arrival events at stops before the stop found. See 10.3.5.
	ThisCall	**1:1**	*+CallAtNearStop*	Departure / arrival event at the stop found. See 10.3.5.
	OnwardCall	0:*	*+CallAtNearStop*	Departure/arrival events at stops after the stop found. See 10.3.5.
	Service	**1:1**	*+DatedJourney*	Specification of mode of transport, line etc. See 7.6.5
OperatingDays	*OperatingDays*	0:1	*+OperatingDays*	Operating days for this departure/arrival event. See 7.4.8.
	OperatingDaysDescription	0:*	*+InternationalText*	Human-readable description of the operating days, e.g. “Monday to Friday” or “Sunday and holidays”.
	ParallelService	0:*	*+ParallelService*	Parallel trips (e.g. in case of portion working). See 7.6.6.
	Extension	0:1	*xs:anyType*	Extensions.

Table 117: Description of structure StopEventStructure

10.3.5 CallAtNearStopStructure

CallAtNearStopStructure			*+Structure*	**Departure or arrival at a nearby stop.**
	CallAtStop	**1:1**	*+CallAtStop*	Departure or arrival at a stopping point. See 7.6.9.
	WalkDistance	0:1	*Distance*	Distance of the stopping point from the requested location in metres. The requested location can be, for example, an address.
	WalkDuration	0:1	*xs:duration*	Duration of the stopping point from the requested location. The requested location can be, for example, an address. The time needed is calculated with the help of individual transport settings in the request: it is taken into consideration whether a bicycle can be used to reach the departure point from the requested location.

Table 118: Description of structure CallAtNearStopStructure

11 Dienst Logische Ortung

11.1 Beschreibung

Der Dienst Logische Ortung hat die Aufgabe, den Aufenthaltsort des Fahrgasts im ÖV-Netz zu bestimmen. Er benutzt dabei das Bewegungsmuster des Fahrgasts, das entweder sein Mobilgerät aufgezeichnet hat oder durch den Fahrplan des Fahrzeugs bestimmt wird, in dem er sich gerade befindet. Als Resultat erhält man mögliche Aufenthaltsorte mit Angabe der jeweiligen Wahrscheinlichkeit.

In der XML-Schema-Definition *Trias_Positioning.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Logische Ortung verwendet werden.

11 Logical positioning service

11.1 Description

The task of the logical positioning service is to determine the stop of the passenger in the public transport network. In the process, it uses movement pattern of the passenger which either tracks the passenger’s mobile device or determines the pattern with the help of the timetable of the vehicle. Possible positions along with the relevant probability is shown as the result.

Data types and structures are defined in the XML schema definition Trias_Positioning.xsd which are used for the logical positioning service.

11.2 Request structures

A logical positioning is requested via an element PositioningRequest of the type PositioningRequestStructure.

11.2.1 PositioningRequestStructure

PositioningRequestStructure				*+Structure*	**Summarises the request data for a logical positioning.**
	a	***LastPositions***	**-1:1**	*+TimedPosition*	Movement pattern of the passenger as a sequence of coordinates with timestamp. See 11.2.3.
	b	***StopSequence***		*+TimedStop*	Movement pattern of the passenger as a sequence of stopping points with times. See 11.2.4.
	Params		0:1	*+PositioningParam*	Specific request parameters. See 11.2.2.

Table 119: Description of structure PositioningRequestStructure

11.2.2 PositioningParamStructure

PositioningParamStructure			*+Structure*	**Summarises the request parameters which control the determination of a logical positioning of the passenger in public transport.**
PositioningData Filter	*PtModeFilter*	0:1	*+PtModeFilter*	Public transport filter. See 7.3.5.
	LineFilter	0:1	*+LineDirectionFilter*	Line filter (if necessary, refined in directions) See 7.4.6.
	OperatorFilter	0:1	*+OperatorFilter*	Transport company filter. See 7.4.4.
Positioning Policy	*NumberOfResults*	0:1	*xs:positiveInteger*	Maximum number of position suggestions which may be returned in the response.

Table 120: Description of structure PositioningParamStructure

11.2.3 TimedPositionStructure

TimedPositionStructure			*+Structure*	**Geographic position with time stamp.**
	Timestamp	**1:1**	*xs:dateTime*	Timestamp, when the passenger has passed this point.
	Position	**1:1**	*+GeoPosition*	Coordinate position. See 7.2.3.
	Speed	0:1	*Speed*	Speed at which the passenger has passed the position. See 7.2.1.
	Direction	0:1	*AbsoluteBearing*	Compass direction, in which the passenger has passed the position. See 7.2.1.

Table 121: Description of structure TimedPositionStructure

11.2.4 TimedStopStructure

TimedStopStructure			*+Structure*	**Describes a stopping point with time, when the passenger reached and/or left.**
Stop Point	***StopPointRef***	**1:1**	*→StopPoint*	Reference to a code for a stopping point. See 7.5.1.
	StopPointName	**1:***	*+InternationalText*	Name of stopping point for passenger information.
	NameSuffix	0:*	*+InternationalText*	Name suffix which can also be left out in case of space shortage, e.g.: “opposite the main entrance”.
	PlannedBay	0:*	*+InternationalText*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to planning state.
	EstimatedBay	0:*	*+InternationalText*	Name of the platform/stopping point, where passengers must board or alight from the vehicle (used in the context of a concrete trip information when a common name is stated in StopPointName, similar to stop name). According to last state of prediction
	ArrivalTime	0:1	*+ServiceCall*	Information about arrival. See 7.6.8.
	DepartureTime	0:1	*+ServiceCall*	Information about departure. See 7.6.8.

Table 122: Description of structure TimedStopStructure

11.3 Response structures

The result of a position request is sent via an element PositioningResponse of the type PositioningResponseStructure.

11.3.1 PositioningResponseStructure

PositioningResponseStructure			*+Structure*	**Summarises the result data for a positioning request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	PositioningResult	0:1	*+PositioningResult*	Structure for a positioning result. See 11.3.2.

Table 123: Description of structure PositioningResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
POSITIONING_NOMATCH	No suitable result found.
POSITIONING_DATEOUTOFRANGE	NO journey data available for the requested date.
POSITIONING_SPEEDTOOFAST	The speed underlying the movement pattern is too fast.
POSITIONING_COORDOUTOFRANGE	The coordinates specifies lie beyond the observed range.
POSITIONING_STOPUNKNOWN	A stop stated is unknown.

Table 124: List of error states in PositioningResponse

11.3.2 PositioningResultStructure

PositioningResultStructure			*+Structure*	**Result structure for positioning result.**
	ResultId	**1:1**	*xs:NMTOKEN*	ID of the result for subsequent referencing or debugging purposes.
	Positioning	**1:1**	*+Positioning*	Container for positioning suggestions. See 11.3.3.
	Extension	0:1	*xs:anyType*	Extensions.

Table 125: Description of structure PositioningResultStructure

11.3.3 PositioningStructure

PositioningStructure			*+Structure*	**Container for positioning suggestions.**
	RankedPosition	**1:***	*+RankedPosition*	One or more positioning suggestions. See 11.3.4.

Table 126: Description of structure PositioningStructure

11.3.4 RankedPositionStructure

RankedPositionStructure				*+Structure*	**Positioning suggestion with probability classification.**
	a	***StationaryLocation***	**-1:1**	*+LocationRef*	Position in public transport outside vehicles. See 7.5.11.
	b	***TripLocation***		*+DatedJourney*	Position in public transport in a trip. See 7.6.5.
	Ranking		**1:1**	*Percent*	Probability ranking in percent. The value 100 means absolute certainty. See 7.2.1.

Table 127: Description of structure RankedPositionStructure

12 Dienst Fahrtinformation (EKAP)

12.1 Beschreibung

Im Dienst Fahrtinformation (EKAP) liefert eine EKAP Informationen zu einer bestimmten Fahrt. Dieser Dienst bezieht die Daten vom Hintergrundsystem EKAP im Unterschied zum Dienst Fahrzeuginformationen (s. Kapitel 22), der die Daten vom Fahrzeug direkt bezieht.

In der XML-Schema-Definition *Trias_TripInfo.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Fahrtinformation (EKAP) verwendet werden.

12 Trip information service (EKAP)

12.1 Beschreibung / Description

In the trip information service (EKAP), EKAP provides information about a certain trip. This service obtains data from background system EKAP in contrast to the service vehicle information (see chapter 22) which obtains data directly from the vehicle.

Data types and structures are defined in the XML schema definition Trias_TripInfo.xsd which are used for the trip information service (EKAP).

12.2 Request structures

Trip information is requested with the help of an element TripInfoRequest of the type TripInfoRequestStructure.

12.2.1 TripInfoRequestStructure

TripInfoRequestStructure				*+Structure*	**Summarises the request data for a trip information request.**
	a	***JourneyRef***	**-1:1**	→*Journey*	Reference to a journey. See 7.4.1.
		OperatingDayRef	**-1:1**	→*OperatingDay*	Reference to an operating day. See 7.4.1.
	b	***VehicleRef***	**-1:1**	→*Vehicle*	Reference to a vehicle. See 7.4.1.
		TimeOfOperation		*xs:dateTime*	Time when the vehicle is in transit. This value matches the time “now” in most use cases.
	Params		0:1	*+TripInfoParam*	Parameters which can affect the search and return values. See 12.2.2.

Table 128: Description of structure TripInfoRequestStructure

The information about a trip can be requested via a journey ID (JourneyRef) or a vehicle ID (VehicleRef).

When using vehicle ID, the trip is uniquely selected from the list of all trips with additional information of time in TimeOfOperation which the vehicle undertakes on the selected day.

12.2.2 TripInfoParamStructure

TripInfoParamStructure			*+Structure*	**Summarises the parameters for a trip information request.**
TripInfo Policy	*UseTimetabledDataOnly*	0:1	*xs:boolean*	Specifies that only target data is used for selecting the trip for the vehicle and the current position in the trip. Default is *false*.
TripInfoContentFilter	*IncludeCalls*	0:1	*xs:boolean*	Specifies whether stops of the trip should be output in the result. Default is *true*.
	IncludeEstimatedTimes	0:1	*xs:boolean*	Specifies whether real-time information (prognosis, cancellations, diversions) should be output in the result. Default is *true*.
	IncludePosition	0:1	*xs:boolean*	Specifies whether the current position of the trip should be output in the result. Default is *true*.
	IncludeService	0:1	*xs:boolean*	Specifies whether information about mode of transport of the trip should be output in the result. Default is *true*.
	IncludeSituationInfo	0:1	*xs:boolean*	Specifies whether textual messages (e.g. messages about faults, events etc.) should be output in the result. Default is *true*.
	IncludeTrackSections	0:1	*xs:boolean*	Specifies whether geographic description of the route of this trip should also be given in the result. Default is *false*.
	IncludeTrackProjection	0:1	*xs:boolean*	Specifies whether geographic projection (coordinate sequence) of the route of this trip should also be given in the result. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 129: Description of structure TripInfoParamStructure

12.3 Response structures

The result of a trip information request is sent via an element TripInfoResponse of the type TripInfoResponseStructure.

12.3.1 TripInfoResponseStructure

TripInfoResponseStructure			*+Structure*	**Summarises the result data for a trip information request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	TripInfoResponse Context	0:1	*+TripInfoResponseContext*	Containers for data which appear multiple times in the response and are referenced. See 12.3.2.
	TripInfoResult	0:1	*+TripInfoResult*	Container for trip information. See 12.3.3.

Table 130: Description of structure TripInfoResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
TRIPINFO_JOURNEYUNKNOWN	The requested journey ID (*JourneyRef*) is unknown.
TRIPINFO_VEHICLEUNKNOWN	The requested vehicle ID (*VehicleRef*) is unknown.
TRIPINFO_NOJOURNEYFOUND	A suitable trip cannot be found for the time for the requested vehicle ID (*VehicleRef*).
TRIPINFO_NOGEOINFO	Geographic information for this trip is not available.

Table 131: List of error states in TripInfoResponse

12.3.2 TripInfoResponseContextStructure

TripInfoResponseContextStructure	***+Structure***(derived from AbstractResponseContext Structur***e)***	Containers for data which appear multiple times in the response and are referenced. See 7.6.17.

Table 132: Description of structure TripInfoResponseContextStructure

12.3.3 TripInfoResultStructure

TripInfoResultStructure			*+Structure*	**Result structure which summarises the trip information.**
	PreviousCall	0:*	*+CallAtStop*	Stops already covered. Also includes the current stop, if the vehicle is at a stop. See 7.6.9.
	CurrentPosition	0:1	*+VehiclePosition*	Current position of vehicle. See 7.6.12.
	OnwardCall	0:*	*+CallAtStop*	The upcoming stops of the trip. See 7.6.9.
	Service	0:1	*+DatedJourney*	Specification of mode of transport, line etc. See 7.6.5
OperatingDays	*OperatingDays*	0:1	*+OperatingDays*	Operating days for this trip. See 7.4.8.
	OperatingDaysDescription	0:*	*+InternationalText*	Human-readable description of the operating days, e.g. “Monday to Friday” or “Sunday and holidays”.
	ParallelService	0:*	*+ParallelService*	Parallel trips (e.g. in case of portion working). See 7.6.6.
	JourneyTrack	0:1	*+LegTrack*	Geographic description of the complete vehicle journey. See 7.6.14.
	Extension	0:1	*xs:anyType*	Extensions.

Table 133: Description of structure TripInfoResultStructure

13 Anschlussdienste

13.1 Beschreibung

Unter dem Begriff „Anschlussdienste“ werden unterschiedliche Dienste des TRIAS-Standards zusammengefasst, die der Kommunikation zu Anschlüssen dienen. Die Anschlussdienste setzen sich aus den Diensten

- Anschlussmeldung,
- Anschlussstatus,
- Info zu Anschlussverlust und
- Anschlussrückmeldung

zusammen. Nachfolgend sind zwei Abläufe dokumentiert, die eine mögliche Nutzung der Dienste darstellen.

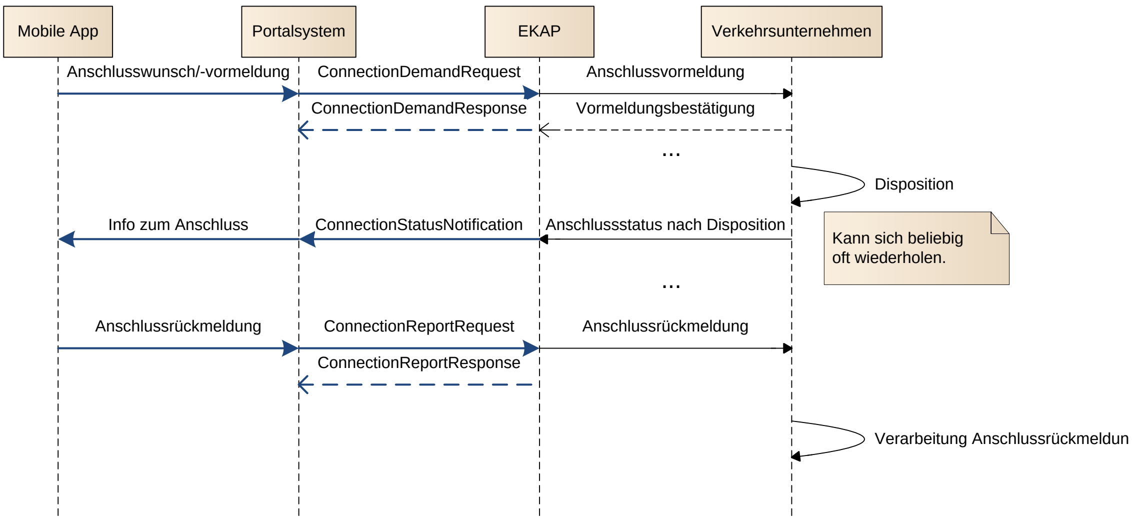

Abbildung 4: Ablauf der Anschlussdienste mit aktiver Benachrichtigung bei Statusänderung

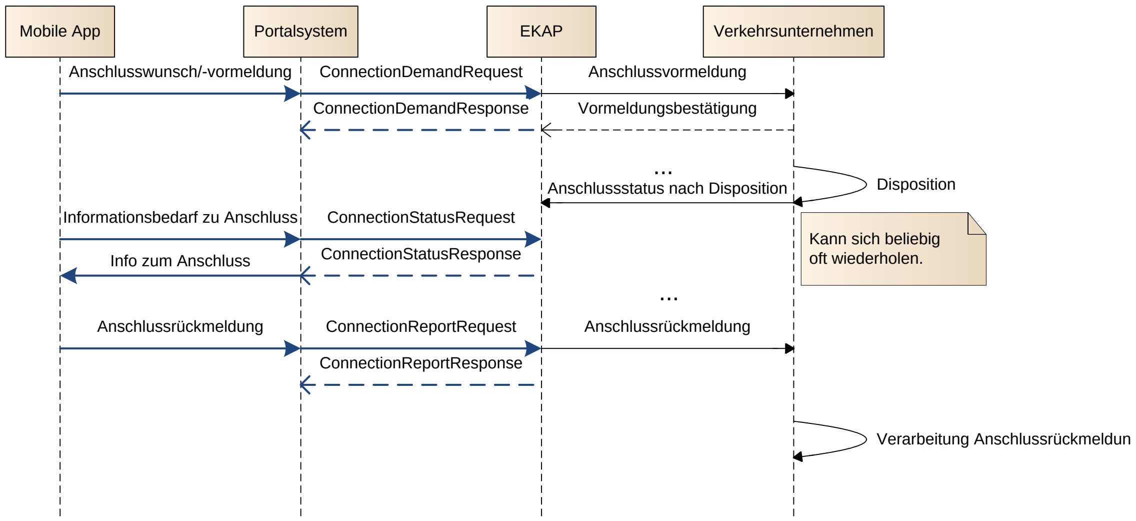

Abbildung 5: Ablauf der Anschlussdienste mit passiver Benachrichtigung bei Statusänderung

Der Ablauf gestaltet sich im Allgemeinen folgendermaßen:

1. Anschlussmeldung durch Reisenden/Applikation, Zugbegleiter oder System über das Portalsystem und die EKAP an das/die beteiligte/n Verkehrsunternehmen
2. Verarbeitung der Anschlussmeldung durch Verkehrsunternehmen und Anschlussdisposition
3. Information über Dispositionsmaßnahme
   a. aktive Information durch EKAP (Abbildung 4)
   b. passive Information mittels Anfrage an die EKAP (Abbildung 5)
4. Rückmeldung des Reisenden oder seiner Applikation zur Anschlusserreichung.

Der Ablauf ist nicht zwingend vorgegeben. Insbesondere kann es sinnvoll sein, einzelne Dienste ohne Bezug zu den anderen zu nutzen. Beispielsweise ist die Abfrage des Anschlussstatus durch die Verbindungsauskunft denkbar. Die Dienste sind genauer in den folgenden Unterkapiteln beschrieben.

13.1.1 Dienst Anschlussvoranmeldung

Mit Hilfe dieses Dienstes können Reisende ihre Anschlusswünsche mitteilen. Auf diese Weise werden Dispositionsverantwortliche und/oder Leitsysteme in die Lage versetzt, Umsteigerzahlen abzuschätzen und in der Entscheidungsfindung einer Anschlussdisposition zu berücksichtigen. Der Mehrwert für die Reisenden ist entsprechend eine verbesserte Anschlussdisposition.

Anschlussbeziehungen umfassen dabei allerdings nicht nur ein Zubringer/Abbringer-Paar, sondern schließen auch Anschlussbeziehungen von einem Startort auf einen Abbringer mit ein. Der Abbringer wiederum kann ein normal verkehrendes Angebot sein, aber auch ein Anrufsammeltaxi (AST) oder Bedarfsverkehr mit fester oder variabler Linienführung und festen oder variablen Halten.

Somit kann und soll die Anschlussmeldung auch als Bestellung für einen Bedarfsverkehr eingesetzt werden.

Der Dienst übermittelt unterschiedliche Grade der Wahrscheinlichkeit, mit der ein Nutzer die gewählte Verbindung nimmt. Auf diese Weise können auch nicht sicher gewählte Verbindungen anhand der Wahrscheinlichkeiten für die Disposition verwendet werden.

Weiterhin besteht die Möglichkeit für die beteiligten Verkehrsunternehmen auf Basis der gemeldeten Umsteiger Rückschlüsse auf die Anzahl der Reisenden im Fahrzeug zu schließen und entsprechende Kapazitäten zu disponieren. Dies gilt insbesondere für die Einstiegs- und Ausstiegsmeldungen.

13.1.2 Dienst Anschlussstatus

Der Dienst Anschlussstatus ermöglicht es den Verkehrsunternehmen, andere Beteiligte über den Status einer Anschlussbeziehung (erwartetes Zustandekommen des Anschlusses) zu informieren.

In erster Linie dient dies der Kundeninformation. Aufgrund der Information über den Anschlussstatus wissen ein Kunde und auch seine Applikation, ob er seine Reisekette (auch bei Verspätung des Zubringers) in der geplanten Weise fortsetzen kann. Seine Applikation kann entsprechend reagieren und Alternativen suchen.

Auch andere Verkehrsunternehmen können Nutzer dieser Information sein. Sie können auf eine Disposition reagieren und von sich aus weitere Maßnahmen im Fall eines abgelehnten Anschlusses einleiten. Ferner lassen sich Prognosen zu Reisendenströmen aufgrund der aktuellen Verkehrslage stellen.

Zur sinnvollen Nutzung des Dienstes ist es erforderlich, dass die Betriebsleitsysteme Anschlussstatusinformationen liefern, sobald sie bekannt werden. Das kann durch eine Dispositionshandlung des Disponenten geschehen oder implizit durch Einflüsse des Betriebsablaufes. Das Betriebsleitsystem meldet die Anschlussinformationen an die Datendrehscheibe einer oder mehrerer EKAPs. Dort kann das Benachrichtigungssystem auf diese Daten zugreifen.

Der Anschlussstatus kann auch im Rahmen einer Verbindungsüberwachung durch den Benachrichtigungsdienst übermittelt werden (siehe Kapitel 20).

13.1.3 Dienst Info bei Anschlussverlust

Durch das Nicht-Zustandekommen eines Anschlusses kann ein Reisender nicht mehr seine ursprünglich geplante Reisekette wahrnehmen. Mit diesem Dienst kann ein Verkehrsunternehmen auf alternative Abbringer, auf die Bestellung von Taxen, Bussen oder Hotelzimmern, die Bereitstellung von Ersatz- oder Sonderfahrten, Umleitungen oder eine Kombination aus unterschiedlichen Maßnahmen verweisen. Die App des Fahrgastes kann anhand der vorgeschlagenen Alternativen prüfen, ob es sich um einen für den Fahrgast sinnvollen Vorschlag handelt und ihn im positiven Fall in die Suche nach Alternativen mit einbeziehen.

Der Dienst ist in die Antworten des Anschlussstatus integriert.

13.1.4 Dienst Anschlussrückmeldung

Mit Hilfe dieses Dienstes können Reisende das Transportunternehmen darüber informieren, ob ein Anschluss aus Sicht des Reisenden erfolgreich disponiert wurde bzw. erfolgreich zustande gekommen ist. Dazu sendet der Reisende eine Nachricht mit einem Anschluss, bestehend aus Zu- und Abbringer und einer Information, ob der Anschluss für ihn zustande kam.

Die Übermittlung der Anschlussrückmeldung kann aber auch automatisch durch die Applikation erfolgen, wenn beispielsweise anhand einer geplanten Verbindung bekannt ist, welche Umstiegsverbindungen genutzt werden sollen. Diese können durch die Applikation überprüft werden, und es kann ein automatisches Feedback bei erkanntem Anschluss oder Anschlussbruch erfolgen. Ein weiteres Beispiel ist mit dem automatischen Erkennen der Fahrzeuge verbunden. Mit Hilfe dieser Funktion kann erkannt werden, wann das Fahrzeug gewechselt wird und dazu eine Anschlusserfolgsmeldung versendet werden.

13 Connection services

13.1 Description

The term "connection services" covers different services of TRIAS standard which help in communicating with connections. The connection services consist of services

- Connection report,
- Connection status,
- Information about missed connection and
- Connection response

Two sequences are documented below which show possible use of the services. The sequence is generally designed as follows:

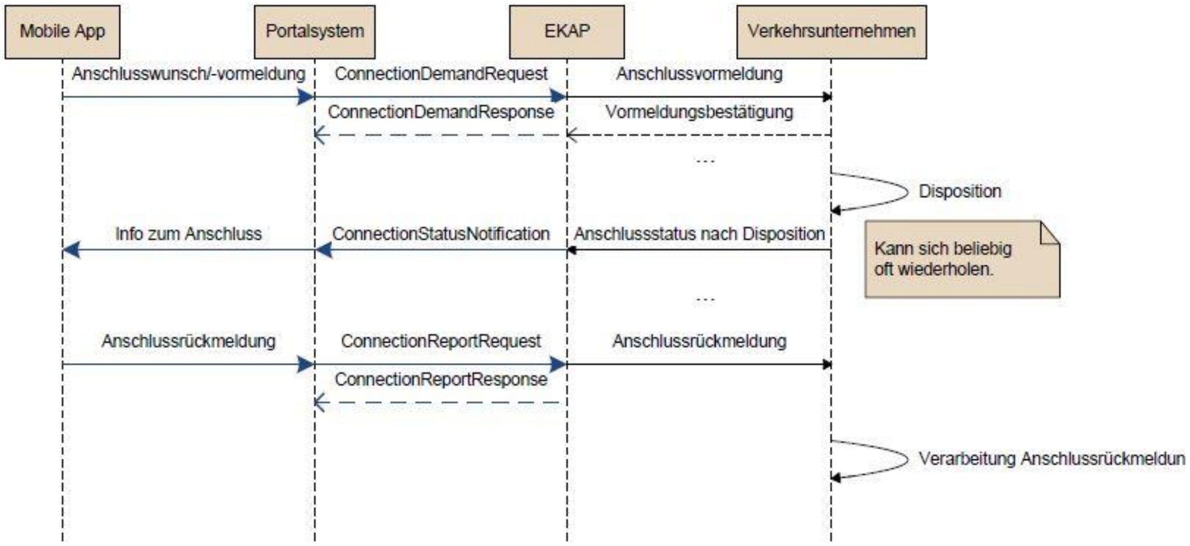

Figure 4: Sequence of connection services with active notification upon status change

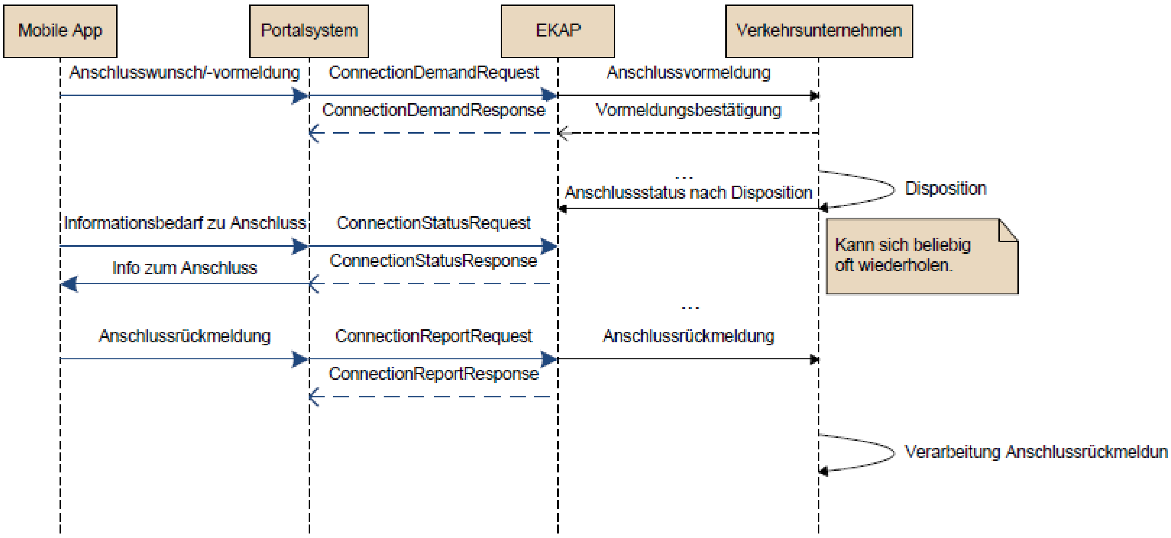

Figure 5: Sequence of connection services with passive notification upon status change

1. Connection report through passengers/application, train attendant or system via the portal system and EKAP to the transport companies involved
2. Processing of connection report by transport companies and connection scheduling
3. Information about scheduling measure
    a. active information via EKAP (Figure 4)
    b. passive information via request to EKAP (Figure 5)
4. Response of passenger or his application for connection establishment

The sequence is not mandatory. In particular, it can be practical to use individual services without reference to other services. For example, it is possible to request connection status via trip information. These services are described in more detail in the following subchapters.

13.1.1 Advance connection report service

Passengers can express their connection requests with the help of this service. In this way, persons in charge of scheduling and/or guidance systems are able to estimate the number of transit passengers and take into consideration a connection schedule during decision-making. An improved connection schedule is an additional benefit for the passengers.

However, connection relations not only cover a feeder / distributor vehicle pair but also include connection relations from a start location to a collecting vehicle. The distributor vehicle can again be a normal operating vehicle but can also be a hailed shared taxi or on-demand transport with fixed or variable line management and fixed or variable stops.

This way, connection report can and should also be used as a message for on-demand transport.

The service transmits different degrees of probability, by means of which a user decides on a connection. In this way, connections that are not selected safely can also be used with the help of probabilities for scheduling.

It is also possible for the transport companies involved to draw conclusions on the number of passengers in the vehicle on the basis of the reported number of transit passengers and plan the corresponding capacities. This is especially applicable for boarding and alighting messages.

13.1.2 Connection status service

The connection status service makes it possible for the transport companies to inform others involved about the status of a connection relation (expected successful occurrence of the connection).

This mainly supports customer information. Based on the information about the connection status, a customer and even his application know whether he can continue his journey chain (even if the feeder vehicle is delayed) as planned. His application can respond accordingly and search for alternatives.

Even other transport companies can use this information. They can respond to a schedule and voluntarily take further actions in case a connection is rejected. Moreover, predictions can be made regarding passenger flows based on the current traffic situation.

For using the service wisely, it is necessary for the control systems to provide connection status information as soon as they are informed. This can be done with the help of a scheduling action of the scheduler or implicitly with the help of influences of operating schedule. The control system reports connection information to the data hub of one or more EKAPs. There, the notification system can access this data.

The connection status can also be sent within the framework of connection monitoring by the notification service (see chapter 20).

13.1.3 Information service in case of missed connection

If a connection does not take place successfully, a passenger can no longer go on his original planned journey chain. Thanks to this service, a transport company can refer to alternative distributor vehicles, ordering taxing, buses or hotel rooms, provision of substitute or special trips, diversions or a combination of different measures. The passenger's App can check with the help of the alternatives suggested, whether the suggestion is practical for him and, if it is, involves him in the search for alternatives.

The service is integrated in the responses of the connection status.

13.1.4 Connection response service

With the help of this service, passengers can inform the transport company whether a connection has been successfully scheduled or successfully occurred from the point of view of the passenger. To this end, the passenger sends a message about the connection, comprising a feeder and distributor vehicle, and informs whether the connection was successful for him.

However, the connection response can also be sent automatically via the application if which transfer connections should be used is known on the basis of the planned connection. This can be checked by the application and an automatic feedback can be sent upon detection of the connection or missed connection. Another example can be associated with automatic detection of vehicles. With the help of this function, it can be detected when the vehicle is changed and for this a connection success message is sent.

13.2 Simple types

The following simple types are defined:

Type name	***Values***	**Description**
ConnectionStatusEnumeration	*unknown | planned | confirmed | broken*	Classification of connection status.
RecommendationTypeEnumeration	*NextService | DifferentRoute | Hotel |Taxi | Bus | Helpdesk | Hotline | Driver | Other*	Classification of detour recommendations in case of missed connection.

Table 134: Description of simple types

13.3 Complex structures

The following sections describe the complex structures which are defined in XML schema Trias_Connections.xsd.

13.3.1 DatedConnectionStructure

DatedConnectionStructure			*+Structure*	**Includes a feeder vehicle and a distributor vehicle for a concrete operating day.**
	ConnectionId	**1:1**	*xs:NMTOKEN*	ID of the connection for subsequent referencing or debugging purposes.
	Feeder	**1:1**	*+FeederDistributor*	Feeder of the reported connection request, see 13.3.2.
	Distributor	**1:1**	*+FeederDistributor*	Distributor of the reported connection request, see 13.3.2.

Table 135: Description of structure DatedConnectionStructure

13.3.2 FeederDistributorStructure

FeederDistributorStructure			*+Structure*	**Includes a feeder or a distributor at a defined location at a defined operating time.**
DatedJourneyRef	***JourneyRef***	**1:1**	*→Journey*	Reference to the journey of the feeder or distributor. See 7.4.1
	OperatingDayRef	**1:1**	*→Operating-Day*	Reference to the operating day. See 7.4.1.
LineDirection	***LineRef***	**1:1**	*→LineCode*	Reference to a line. See 7.4.1.
	DirectionRef	0:1	*→Direction Code*	Reference to a line direction. See 7.4.1.
	OperatorRef	0:1	*→Operator*	Operator-ID. See 7.4.1.
	ConnectionLocation	**1:1**	*+CallAtStop*	Location of connection, see 7.6.9.

Table 136: Description of structure FeederDistributorStructure

13.3.3 GeneralizedConnectionStructure

GeneralizedConnectionStructure				*+Structure*	**Defines a transfer relation. Includes connection types such as boarding (pickup), alighting (set-down) and transfer (DatedConnection)**
	a	***DatedConnection***	**-1:1**	*+DatedConnection*	Transfer connection, for which the status should be requested. See 13.3.1.
	b	***PickUpLocation***		*+DatedCallAtLocation*	Boarding in vehicle, for which the status should be requested. See 7.6.10.
	c	***SetDownLocation***		*+DatedCallAtLocation*	Alighting from vehicle, for which the status should be requested. See 7.6.10.

Table 137: Description of type GeneralizedConnectionStructure

13.3.4 ConnectionStatusStructure

ConnectionStatusStructure			*+Structure*	**Contains the actual connection status. This comprises boarding, alighting or transfer and an associated status.**
	Connection	**1:1**	*+GeneralizedConnection*	Boarding, alighting or transfer See 13.3.3.
	Status	**1:1**	*ConnectionStatusEnumeration*	Connection status. See 13.2.
	Alternative	0:*	*+Recommendation*	Alternatives for missed connection, see 13.3.5.

Table 138: Description of structure ConnectionStatusStructure

13.3.5 RecommendationStructure

This element represents the service “information in case of missed connection”.

RecommendationStructure			*+Structure*	**Contains alternative suggestions in case of missed connection which are more than just a connection alternative. As an alternative, in addition to other existing journeys, substitute transports provided, accommodations or other options to be provided by means of scheduling**
	RecommendationId	**1:1**	*xs:NMTOKEN*	ID of the request for subsequent referencing or debugging purposes.
	Text	**1:***	*+InternationalText*	Description of alternatives.
	Type	**1:1**	*RecommendationTypeEnumeration*	Type of recommendation, see 13.2.

Table 139: Description of structure RecommendationStructure

13.4 Request structures of connection report

13.4.1 ConnectionDemandRequestStructure

ConnectionDemandRequestStructure			*+Structure*	**Includes request data for a message of transfer passengers, boarding passengers or alighting passengers in case of regular or on-demand transport.**
	RequestId	**1:1**	*xs:NMTOKEN*	ID of the request for subsequent referencing or debugging purposes.
	Connection	**1:1**	*+GeneralizedConnection*	Connection relation. See 13.3.3.
	NumberOfPersons	0:1	*xs:positiveInteger*	Number of transfer passengers, boarding passengers or alighting passengers
	TravelProbability	0:1	*Percent*	(Cumulative) travel probability for the transfer passengers, boarding passengers or alighting passengers stated. See 7.2.1.
	RequiredInterchangeDuration	0:1	*xs:duration*	Specifies interchange duration, which is necessary for the passengers, who have initiated this connection request, to reach from the feeder to the distributor vehicle.
	PassengerAccessibilityNeeds	0:*	*+PassengerAccessibility*	For every (anonymous) passenger with special needs, an element can be transmitted which expresses these needs. See 7.6.19.
	Extension	0:1	*xs:anyType*	Extensions.

Table 140: Description of structure ConnectionDemandRequestStructure

13.4.2 ConnectionDemandDeleteRequestStructure

ConnectionDemandDeleteRequestStructure			*+Structure*	Request structure in order to cancel an advance connection notification.
	RequestId	**1:1**	*xs:NMTOKEN*	ID of the request which should be cancelled.

Table 141: Description of structure ConnectionDemandDeleteRequestStructure

13.5 Response structures of connection report

13.5.1 ConnectionDemandResponseStructure

ConnectionDemandResponseStructure			*+Structure*	**Response to a request of the type ConnectionDemandRequest**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Also see 7.4.2.

Table 142: Description of structure ConnectionDemandResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
CONNECTIONDE-MAND_FEEDER_UNKNOWN	The feeder is not known to EKAP.
CONNECTIONDE-MAND_DISTRIBUTOR_UNKNOWN	The distributor is not known to EKAP.
CONNECTIONDE-MAND_DEPARTURE_BEFORE_ARRIVAL	The target departure of the distributor lies before the target arrival of the feeder. Hence, transfer not possible.
CONNECTIONDE-MAND_FEEDER_LOCATION_UNKNOWN	The referenced location of transfer for the feeder is unknown.
CONNECTIONDE-MAND_DISTRIBUTOR_LOCATION_UNKNOWN	The referenced location of transfer for the distributor is unknown.

Table 143: List of error states in ConnectionDemandResponse

13.5.2 ConnectionDemandDeleteResponseStructure

ConnectionDemandDeleteResponseStructure			*+Structure*	**Response to a request of the type ConnectionDemandDeleteRequest.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Also see 7.4.2.

Table 144: Description of structure ConnectionDemandDeleteResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
CONNECTIONDE-MAND_REQUESTID_UNKNOWN	The request ID is not known to EKAP.
CONNECTIONDE-MAND_DELETIONNOTPOSSIBLE	The request message could not be cancelled.

Table 145: List of error states in ConnectionDemandDeleteResponse

13.6 Request structurs of connection status

13.6.1 ConnectionStatusRequestStructure

ConnectionStatusRequestStructure			*+Structure*	**Supports the active request of a connection status.**
	RequestId	**1:1**	*xs:NMTOKEN*	ID of the request for subsequent referencing or debugging purposes.
	Connection	**1:1**	*+Generalized Connection*	Connection relation. See 13.3.3.
	Extension	0:1	*xs:anyType*	Extensions.

Table 146: Description of structure ConnectionStatusRequestStructure

13.6.2 ConnectionStatusNotificationStructure

ConnectionStatusNotificationStructure			+Structure	**Push-information about connection status. Used by TripMonitoring or optionally actively sent by EKAP as per ConnectionDemands.**
	ConnectionStatus	**1:1**	*+ConnectionStatus*	Contains the actual connection status. See 13.3.4.

Table 147: Description of structure ConnectionStatusNotificationStructure

13.7 Response structures of connection status

13.7.1 ConnectionStatusResponseStructure

ConnectionStatusResponseStructure			*+Structure*	**Provides connection status or an error message on request ConnectionStatusRequest.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.
	ConnectionStatus	0:1	*+ConnectionStatus*	Contains the actual connection status. See 13.3.4.

Table 148: Description of structure ConnectionStatusResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
CONNECTIONSTATUS_FEEDER_UNKNOWN	The feeder is not known to EKAP.
CONNECTIONSTATUS _DISTRIBUTOR_UNKNOWN	The distributor is not known to EKAP.
CONNECTIONSTATUS _DEPARTURE_BEFORE_ARRIVAL	The target departure of the distributor lies before the target arrival of the feeder. Hence, transfer not possible.
CONNECTIONSTATUS _FEEDER_LOCATION_UNKNOWN	The referenced location of transfer for the feeder is unknown.
CONNECTIONSTATUS _DISTRIBUTOR_LOCATION_UNKNOWN	The referenced location of transfer for the distributor is unknown.

Table 149: List of error states in ConnectionStatusResponseStructure

13.8 Request structures of connection response

13.8.1 ConnectionReportRequestStructure

ConnectionReportRequestStructure			*+Structure*	**Contains data for a message that indicates whether the connection has successfully occurred for the passenger.**
	RequestId	**1:1**	*xs:NMTOKEN*	ID of the request for subsequent referencing or debugging purposes.
	Connection	**1:1**	*+Generalized Connection*	Connection relation which is reported about. See 13.3.3.
	Succeeded	**1:1**	*xs:boolean*	Specifies whether the connection has succeeded from the point of view of the passenger.
	Reason	0:*	*+International Text*	It is also possible to specify a reason for the missed connection or success of the connection.
	Extension	0:1	*xs:anyType*	Extensions.

Table 150: Description of structure ConnectionReportRequestStructure

13.9 Response structures of connection response

13.9.1 ConnectionReportResponseStructure

ConnectionReportResponseStructure			*+Structure*	**Response to a request of the type ConnectionReportRequest.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.

Table 151: Description of structure ConnectionReportResponseStructure

In ErrorMessage, the following error states can appear:

CONNECTIONRE-PORT_FEEDER_UNKNOWN	The feeder is not known to EKAP.
CONNECTIONREPORT _DISTRIBUTOR_UNKNOWN	The distributor is not known to EKAP.
CONNECTIONREPORT _DEPARTURE_BEFORE_ARRIVAL	The target departure of the distributor lies before the target arrival of the feeder. Hence, transfer not possible.
CONNECTIONREPORT _FEEDER_LOCATION_UNKNOWN	The referenced location of transfer for the feeder is unknown.
CONNECTIONREPORT _DISTRIBUTOR_LOCATION_UNKNOWN	The referenced location of transfer for the distributor is unknown.

Table 152: List of error states in ConnectionReportResponse

14 Dienst Fahrpreis- und Tarifberechnung

14.1 Beschreibung

Dieser Dienst stellt allgemeine, haltestellenbezogene oder verbindungsbezogene Tarifinformationen bereit. In der XML-Schema-Definition *Trias_Fares.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Fahrpreis- und Tarifberechnung verwendet werden.

14 Fare calculation service

14.1 Description

This service provides general fare information related to stops or connections. Data types and structures are defined in the XML schema Trias_Fares.xsd which are used for the fare calculation service.

14.2 Request structures

A request to the fare calculation service is sent via an element FaresRequest of the type FaresRequestStructure.

14.2.1 FaresRequestStructure

FaresRequestStructure				*+Structure*	**Summarises the data for a fare request.**
	a	***StopFaresRequest***	**-1:1**	*+StopFaresRequest*	Fare request related to stops. See 14.2.2.
	b	***StaticFaresRequest***		*+StaticFaresRequest*	General fare request. See 14.2.3.
	c	***TripFaresRequest***		*+TripFaresRequest*	Fare request related to trip. See 14.2.3.
	d	***MultiTripFaresRequest***		*+MultiTripFaresRequest*	Aggregate fare request for multiple trips. See 14.2.5.
	Params		0:1	*+ FaresParam*	Parameters for fare request. See 7.10.8.
	Extension		0:1	*xs:anyType*	Extensions.

Table 153: Description of structure FaresRequestStructure

14.2.2 StopFaresRequestStructure

The fare request related to stops determines fare information which is applicable to a certain stop, e.g. tariff zones in which the stop is located.

StopFaresRequestStructure			*+Structure*	**Summarises the data for a fare request based on a stop.**
	StopPointRef	**1:1**	*→StopPointCode*	References a stopping point. See 7.5.1.
	Date	0:1	*xs:date*	Key date for the validity of fare of fare information.

Table 154: Description of structure StopFaresRequestStructure

14.2.3 StaticFaresRequestStructure

The static fare request ascertains general fare information such as a list of available ticket types or an URL to further fare information (e.g. fare zone plans, fare provisions etc.).

StaticFaresRequestStructure			*+Structure*	**Summarises the data for a static fare request.**
	Date	0:1	*xs:date*	Key date for the validity of fare information.
	TicketRef	0:*	*→TicketCode*	Code of tickets, for which further information is requested. If TicketRef is not specified, the server should provide information about all the available tickets.

Table 155: Description of structure StaticFaresRequestStructure

14.2.4 TripFaresRequestStructure

The trip-related fare request ascertains the tickets and their prices in question for a certain trip.

TripFaresRequestStructure			*+Structure*	**Summarises the data for a fare request related to a trip.**
	Trip	**1:1**	*+Trip*	Contains the trip, for which the fare information must be ascertained. See 9.3.4.

Table 156: Description of structure TripFaresRequestStructure

14.2.5 MultiTripFaresRequestStructure

The difference between MultiTripFaresRequestStructure and TripFaresRequestStructure is that in case of MultiTripFaresRequestStructure the server is requested to find out the most favourable ticket combination which covers the trips, for example, a day ticket, if sufficient trips are supposed to take place on the same day.

MultiTripFaresRequestStructure			*+Structure*	**Summarises the data for a fare request for several trips.**
	Trip	**1:***	*+Trip*	Contains the trips, for which the fare information must be ascertained. See 9.3.4.

Table 157: Description of structure MultiTripFaresRequestStructure

14.3 Response structures

The result of a fare request is sent via an element FaresResponse of the type FaresResponseStructure.

14.3.1 FaresResponseStructure

FaresResponseStructure			*+Structure*	**Summarises the result data for a fare request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	FaresResult	0:*	*+FaresResult*	Structure for a fare result. See 14.3.2.

Table 158: Description of structure FaresResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
FARES_DATEOUTOFRANGE	The fare request cannot be processed because information is not available for the desired date.
FARES_STOPPOINTUNKNOWN	The fare request cannot be processed because the requested stopping point is unknown.

Table 159: List of error states in FaresResponse

14.3.2 FaresResultStructure

FaresResultStructure				*+Structure*	**Result structure for fare information.**
		ResultId	**1:1**	*xs:NMTOKEN*	ID of the result for subsequent referencing.
	a	***StopFaresResult***	**-1:1**	*+StopFaresResult*	Response to fare request related to stops. See 14.3.3.
	b	***StaticFaresResult***		*+StaticFaresResult*	Response to general fare request. See 14.3.4.
	c	***TripFaresResult***		*+TripFaresResult*	Response to fare request related to trip. See 7.10.6.
	d	***MultiTripFaresResult***		*+MultiTripFaresResult*	Response to fare request for multiple trips. See 14.3.6.

Table 160: Description of structure FaresResultStructure

14.3.3 StopFaresResultStructure

StopFaresResultStructure			*+Structure*	**Result structure for fare information related to stops.**
	FareZoneListInArea	**1:***	*+FareZoneListInArea*	List of tariff zones, in which the requested stop lies. See 7.10.3.
	Extension	0:1	*xs:anyType*	Extensions.

Table 161: Description of structure StopFaresResultStructure

14.3.4 StaticFaresResultStructure

StaticFaresResultStructure			*+Structure*	**Result structure for general fare information.**
	Ticket	0:*	*+Ticket*	List of available tickets. See 7.10.5.
	StaticInfoUrl	0:1	*+WebLinkI*	Links to information pages on the web (see 7.2.4).
	Extension	0:1	*xs:anyType*	Extensions.

Table 162: Description of structure StaticFaresResultStructure

14.3.5 TripTicketReferenceStructure

TripTicketReferenceStructure			*+Structure*	**Combination of tickets and links (or parts thereof).**
	TicketRef	**1:1**	→*TicketCode*	Reference to a ticket.
	FromTripIdRef	**1:1**	*xs:NMTOKEN*	Reference to a trip, from which a ticket is valid.
	FromTripLegIdRef	0:1	*xs:NMTOKEN*	Reference to a leg of the trip, from which the ticket is valid.
	ToTripIdRef	**1:1**	*xs:NMTOKEN*	Reference to a trip, up to which a ticket is valid.
	ToTripLegIdRef	0:1	*xs:NMTOKEN*	Reference to a leg of the trip, up to which the ticket is valid.

Table 163: Description of structure TripTicketReferenceStructure

14.3.6 MultiTripFaresResultStructure

MultiTripFaresResultStructure			*+Structure*	**Summarises the result data for a fare information for multiple trips.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on this tariff information. Refer to the following table for possible values. Also see 7.4.2.
	TripTicketReference	**1:***	*+TripTicketReference*	Combination of tickets and links (or parts thereof). See 14.3.5
	Ticket	0:*	*+Ticket*	Tickets which are valid on this section of the trip (see 7.10.5).
	PassedZones	0:1	*+FareZoneListInArea*	List of tariff zones passed, seen across all trips (see 7.10.3).
	StaticInfoURL	0:*	*+WebLink*	URL for information pages (see 7.2.4).

Table 164: Description of structure MultiTripFaresResultStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
FARES_OUTOFAREA	The route found in the trip information leaves the tariff area.
FARES_JOURNEYNOTPERMITTED	A mode of transport used in the trip information is not permissible for the tariff.
FARES_ADDITIONALCHARGES	Additional fare must most likely have to be paid (e.g. toll surcharges or reservation fees).
FARES_ADDITIONALTICKETS	Additional tickets are necessary because a suitable ticket could not be ascertained for all modes of transport or all trips stated.
FARES_ROUTENOTFEASIBLE	A ticket cannot be ascertained because the route in the trip information does not comply with the tariff rules (e.g. due to round trips, diversions or exceeding the permissible overall duration).

Table 165: List of error states in MultiTripFaresResultStructure

15 Dienst Anreicherung

15.1 Beschreibung

Dieser Dienst ist dazu gedacht, für bereits vorher bekannte Objekte zusätzliche (oder aktualisierte) Informationen von einer EKAP holen zu können.

Auf welchem Wege diese Objekte bereits vorab bekannt geworden sind, ist dabei nicht relevant; das kann per TRIAS-Schnittstelle oder auch auf einem anderen Wege geschehen sein.

Die Motivation für den Anreicherungsdienst kam aus den Bedürfnissen des Forschungsprojektes DELFIplus, bei dem die verteilte DELFI-Auskunft in ein hybrides Auskunftssystem überführt wurde, welches auf einem deutschlandweiten Datenbestand Verbindungen berechnet, nicht-integrierte Informationen wie z.B. Tarife oder Echtzeitinformationen von lokalen EKAPs holen möchte, die über ensprechendes Spezialwissen verfügen. Der Anreicherungsdienst wurde aber über die unmittelbaren Bedürfnisse des Forschungsprojektes DELFIplus hinaus spezifiziert, so dass sinnvolle allgemeine Anreicherungsmöglichkeiten im Rahmen der VDV431 geschaffen werden.

Aus diesem Szenario lässt sich leicht ableiten, dass ein anfragendes System Informationen von zwei (oder mehr) verschiedenen EKAPs erhalten möchte. Z.B. könnten für eine Verbindungsauskunft nachträglich Prognoseinformationen von einer anderen EKAP abgefragt werden. In diesem Fall ergibt sich die Schwierigkeit, dass jede EKAP ihr eigenes ID-System mitbringt, innerhalb dessen die Objekte, die sie kennt, referenziert werden können. Wenn nun eine Verbindungsauskunft auf EKAP A gerechnet wird und Prognoseinformationen für ein Verkehrsmittel aus diesem Verbindungsergebnis von EKAP B angereichert werden sollen, so kann man nicht ohne Weiteres davon ausgehen, dass EKAP B die betreffende Fahrt-ID, die von EKAP A verwendet wurde, kennt. Aus diesem Grund wird in den Anreicherungsanfragen die Information mitgegeben, ob in den Anfragedaten „Fremd-IDs“ enthalten sind.

Falls eine EKAP mit fremden IDs konfrontiert wird, muss sie versuchen, die entsprechenden Objekte auf andere Weise zu erkennen und im eigenen Datenbestand wiederzufinden. In der Antwort wird die EKAP notwendigerweise ihre eigenen Objekt-IDs verwenden. Damit das anfragende System die in der Anreicherungsantwort enthaltenen Objekte zweifelsfrei zuordnen kann, ist es unerlässlich, dass die EKAP in ihrer Antwort exakt die Struktur der Anreicherungsanfrage widerspiegelt (Prinzip der Strukturerhaltung). Wenn also z.B. eine Anreicherungsanfrage für den StopEvents-Dienst (*StopEventRefineRequest*) zehn Elemente vom Typ *StopEventResult* enthält und die EKAP für die Elemente 2, 7 und 10 eine Anreicherung bereitstellen kann, ist es erforderlich, dass die EKAP auch die anderen Elemente unverändert und in der ursprünglichen Reihenfolge zurückgibt.

Die Anreicherung ist für folgende Dienste möglich:

- IndividualRoute
  Beispiel: eine vorher ermittelte PKW-Route wird nachträglich mit der geografischen Streckenführung angereichert
- LocationInformation
  Beispiel: zu einer Haltestellen-ID werden die Details der Haltestelle ermittelt (Name, Ort, Koordinaten, …)

- StopEvent
  Beispiel: Zu einer Haltestellenabfahrt (unter vielen) werden nachträglich die vorausgehenden und nachfolgenden Halte der Fahrt ermittelt
- TripInfo
  Beispiel: zu einer Fahrplanfahrt werden nachträglich die Echtzeitinformationen ermittelt (Zuglauf)
- Trip
  Beispiel: zu einer Verbindungsauskunft werden nachträglich die Fahrpreiskosten und mögliche Tickets abgefragt

Um die Kommunikation effizient zu halten, besteht mit einem allgemeinen Refine-Dienst die Möglichkeit, mehrere Anreicherungsanfragen in einem HTTP-Request zu verpacken.

In der XML-Schema-Definition *Trias_Refine.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Anreicherung verwendet werden.

15 Refining service

15.1 Description

This service is intended to be able to obtain additional (or updated) information from an EKAP for objects already previously known.

By which means have these objects become known previously is not relevant here; it can carried out via TRIAS interface or other channels.

The motivation behind the refining service came from the requirements of the research project DELFIplus, in which the distributed information was transferred into a hybrid information system, which develops trips in a database across Germany and would like to obtain non-integrated information such as fares or real-time information from local EKAPs, which have corresponding special knowledge. However, the refining service has been specified beyond the immediate needs of the research project DEL- FIplus so that meaningful general refining options can be provided within the scope of VDV431.

It can be easily inferred from this scenario that a requesting system would like to receive information from two (or more) different EKAPs. For example, additional forecast information could be requested from another EKAP for a trip information. In this case, it is difficult for every EKAP to carry its own ID system, within which the objects, which it knows, can be referenced. If now a trip information is ascertained on EKAP A and forecast information is supposed to be refined by EKAP B for a mode from this trip result, it can be easily assumed that EKAP B knows the trip ID in question which was used by EKAP A. For this reason, the information is provided in the refine requests, as to whether “external IDs” are included in the request data.

If an EKAP is confronted with external IDs, it must try to recognise the relevant objects in another way and retrieve them in a separate database. In the response, EKAP will certainly use its own object IDs. In order that the requesting system can assign the objects included in the refine response unequivocally, it is imperative that the EKAP reflects the structure of the refine request precisely in its response (principle of structure maintenance).

For example, if a refine request for the StopEvents service (StopEventRefineRequest) includes ten elements of the type StopEventResult and EKAP can provide refinement for elements 2, 7 and 10, it is necessary that EKAP also returns the other elements unchanged and in their original sequence.

The refinement is possible for the following services:

- IndividualRoute
  - Example: a car route ascertained in advance is subsequently refined with geographical routing
- LocationInformation
  - Example: the details of a stop are determined for a stop ID (name, location, coordinates, ...)
- StopEvent
  - Example: The previous and upcoming stops of the trip are subsequently determined for a stop departure (among many)
- TripInfo
  - Example: real-time information is subsequently determined for a journey (train course)
- Trip
  - Example: the fare cost and possible tickets are subsequently requested for a trip information

In order to keep the communication efficient, it is possible with the help of a general refine service to pack several refine requests into one HTTP request.

Data types and structures are defined in the XML schema definition Trias_Refine.xsd which are used for the refining service.

15.2 Request structures

An information refinement is requested via an element RefineRequest of the type RefineRequestStructure.

15.2.1 RefineRequestStructure (in Trias_Refine.xsd)

RefineRequestStructure			*+Structure*	**Requests additional information.**
	IndividualRouteRefineRequest	0:*	*+IndividualRouteRefineRequest*	Individual transport routing structures which should be refined. See 15.2.2.
	LocationInformationRefineRequest	0:*	*+LocationInformationRefineRequest*	Location information which should be refined. See 15.2.3.
	StopEventRefineRequest	0:*	*+StopEventRefineRequest*	Stop events which should be refined. See 15.2.4.
	TripInfoRefineRequest	0:*	*+TripInfoRefineRequest*	Trip information which should be refined. See 15.2.5.
	TripRefineRequest	0:*	*+TripRefineRequest*	Trip information which should be refined. See 15.2.6

Table 166: Description of structure RefineRequestStructure

15.2.2 IndividualRouteRefineRequestStructure (in Trias_IndividualTrips.xsd)

IndividualRouteRefineRequestStructure			*+Structure*	**Requests additional information for individual transport routing.**
	RefineParams	0:1	*+IndividualTripRefineParam*	Selects the desired refinements. See 15.2.7.
	IndividualRouteResult	**1:1**	*+RouteResult*	The individual route to be refined. See 17.3.2.
	IndividualRouteResponseContext	0:1	*TripResponseContextStructure*	Context to hold objects, which are referenced within the response. It is used e.g. within stop points to refer to the parent stop.
	Extension	0:1	*xs:anyType*	Extensions.

Table 167: Description of structure IndividualRouteRefineRequestStructure

15.2.3 LocationInformationRefineRequestStructure (in Trias_Locations.xsd)

LocationInformationRefineRequestStructure			*+Structure*	**Requests additional information for location information.**
	RefineParams	0:1	*+LocationInformationRefineParam*	Options for refinement function. See 15.2.8.
	LocationResult	**1:1**	*+LocationResult*	The location information to be refined. See 8.4.2.
	Extension	0:1	*xs:anyType*	Extensions.

Table 168: Description of structure LocationInformationRefineStructure

15.2.4 StopEventRefineRequestStructure (in Trias_StopEvents.xsd)

StopEventRefineRequestStructure			*+Structure*	**Requests additional information for stop events.**
	RefineParams	0:1	*+StopEventRefineParam*	Selects the desired refinements. See 15.2.9.
	StopEventResult	**1:***	*+StopEventResult*	The stop events to be refined. See 10.3.3.
	StopEventResponseContext	0:1	*StopEventResponseContextStructure*	Context to hold objects, which are referenced within the response. It is used e.g. within stop points to refer to the parent stop.
	Extension	0:1	*xs:anyType*	Extensions.

Table 169: Description of structure StopEventRefineRequestStructure

15.2.5 TripInfoRefineRequestStructure (in Trias_TripInfo.xsd)

TripInfoRefineRequestStructure			*+Structure*	**Requests additional information for trip information.**
	RefineParams	0:1	*+TripInfoRefineParam*	Selects the desired refinements. See 15.2.10.
	TripInfoResult	**1:1**	*+TripInfoResult*	The trip information to be refined. See 12.3.3.
	TripInfoResponseContext	0:1	*TripInfoResponseContextStructure*	Context to hold objects, which are referenced within the response. It is used e.g. within stop points to refer to the parent stop.
	Extension	0:1	*xs:anyType*	Extensions.

Table 170: Description of structure TripInfoRefineRequestStructure

15.2.6 TripRefineRequestStructure (in Trias_Trips.xsd)

TripRefineRequestStructure			*+Structure*	**Requests additional information for trip information.**
	RefineParams	0:1	*+TripRefineParam*	Selects the desired refinements. See 15.2.11.
	TripResult	**1:1**	*+TripResult*	The trip information to be refined. See 9.3.3.
	TripResponseContext	0:1	*TripResponseContextStructure*	Context to hold objects, which are referenced within the response. It is used e.g. within stop points to refer to the parent stop.
	Extension	0:1	*xs:anyType*	Extensions.

Table 171: Description of structure TripRefineRequestStructure

15.2.7 IndividualTripRefineParamStructure (in Trias_IndividualTrips.xsd)

IndividualTripRefineParamStructure			*+Structure*	**Parameterises the refine requests of individual routing**
Refine Options	*ForeignObjectRefs*	0:1	*xs:boolean*	If true, this element indicates that object references from other EKAPs can be included in the request data. Default is *false*.
BaseTrip Content Filter	*IncludeTrackSections*	0:1	*xs:boolean*	Specifies whether TrackSection element (see 7.6.15) should be output in the result for a detailed geographic description of the route. Default is *false*.
	IncludeLegProjection	0:1	*xs:boolean*	Specifies whether the detailed geographic route should be output in the result as a coordinate sequence. Default is *false*.
	IncludeTurnDescription	0:1	*xs:boolean*	Specifies whether route information should be output in the result with turn recommendations. Default is *false*.
	IncludeAccessibility	0:1	*xs:boolean*	Specifies whether information about barrier freedom should be output in the result. Default is *false*.
	IncludeEstimatedTimes	0:1	*xs:boolean*	Specifies whether information about real-time situation should be output in the result. Default is *false*.
	IncludeSituationInfo	0:1	*xs:boolean*	Specifies whether error messages should be output in the result. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 172: Description of structure IndividualTripRefineParamStructure

15.2.8 LocationInformationRefineParamStructure (in Trias_Locations.xsd)

LocationInformationRefineParamStructure			*+Structure*	**Parameterises the refine requests of location information.**
Refine Options	*ForeignObjectRefs*	0:1	*xs:boolean*	If true, this element indicates that object references from other EKAPs can be included in the request data. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 173: Description of structure LocationInformationRefineParamStructure

15.2.9 StopEventRefineParamStructure (in Trias_StopEvents.xsd)

StopEventRefineParamStructure			*+Structure*	**Parameterises the refine requests of stop events**
Refine Options	*ForeignObjectRefs*	0:1	*xs:boolean*	If *true*, this element indicates that object references from other EKAPs can be included in the request data. Default is *false*.
StopEventContentFilter	*IncludePreviousCalls*	0:1	*xs:boolean*	Specifies whether the previous stops should be stated for every journey. Default is *false*.
	IncludeOnwardCalls	0:1	*xs:boolean*	Specifies whether the subsequent stops should be stated for every journey. Default is *false*.
	IncludeOperatingDays	0:1	*xs:boolean*	Specifies whether the operating days of the trips should be stated. Default is *false*.
	IncludeRealtimeData	0:1	*xs:boolean*	Controls whether real-time data should be considered and displayed. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 174: Description of structure StopEventRefineParamStructure

15.2.10 TripInfoRefineParamStructure (in Trias_TripInfo.xsd)

TripInfoRefineParamStructure			*+Structure*	**Parameterises the refine requests as per trip information**
Refine Options	*ForeignObjectRefs*	0:1	*xs:boolean*	If *true*, this element indicates that object references from other EKAPs can be included in the request data. Default is *false*.
TripInfoContentFilter	*IncludeCalls*	0:1	*xs:boolean*	Specifies whether stops of the trip should be output in the result. Default is *true*.
	IncludeEstimatedTimes	0:1	*xs:boolean*	Specifies whether real-time information (prognosis, cancellations, diversions) should be output in the result. Default is *true*.
	IncludePosition	0:1	*xs:boolean*	Specifies whether the current position of the trip should be output in the result. Default is *true*.
	IncludeService	0:1	*xs:boolean*	Specifies whether information about mode of transport of the trip should be output in the result. Default is *true*.
	IncludeSituationInfo	0:1	*xs:boolean*	Specifies whether textual messages (e.g. messages about faults, events etc.) should be output in the result. Default is *true*.
	IncludeTrackSection	0:1	*xs:boolean*	Specifies whether geographic description of the route of this trip should also be given in the result. Default is *false*.
	IncludeLegProjection	0:1	*xs:boolean*	Specifies whether geographic projection (coordinate sequence) of the route of this trip should also be given in the result. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 175: Description of structure TripInfoRefineParamStructure

15.2.11 TripRefineParamStructure (in Trias_Trips.xsd)

TripRefineParamStructure			*+Structure*	**Parameterises the refine requests for trip information**
Refine Options	*ForeignObjectRefs*	0:1	*xs:boolean*	If *true*, this element indicates that object references from other EKAPs can be included in the request data. Default is *false*.
	RefineLegRef	1:*	→*xs:NMTOKEN*	Specifies for which legs of the trip the desired refinements should be carried out.
BaseTrip Content Filter	*IncludeTrackSections*	0:1	*xs:boolean*	Specifies whether TrackSection element (see 7.6.15) should be output in the result for a detailed geographic description of the route. Default is *false*.
	IncludeLegProjection	0:1	*xs:boolean*	Specifies whether the detailed geographic route should be output in the result as a coordinate sequence. Default is *false*.
	IncludeTurnDescription	0:1	*xs:boolean*	Specifies whether route information should be output in the result with turn recommendations. Default is *false*.
	IncludeAccessibility	0:1	*xs:boolean*	Specifies whether information about barrier freedom should be output in the result. Default is *false*.
	IncludeEstimatedTimes	0:1	*xs:boolean*	Specifies whether information about real-time situation should be output in the result. Default is *false*.
	IncludeSituationInfo	0:1	*xs:boolean*	Specifies whether error messages should be output in the result. Default is *false*.
TripContentFilter	*IncludeIntermediate-Stops*	0:1	*xs:boolean*	Specifies whether intermediate stops should be output in the result. Default is *false*.
	IncludeFares	0:1	*xs:boolean*	Specifies whether fare information should be output in the result. Default is *false*.
	IncludeOperatingDays	0:1	*xs:boolean*	Specifies whether information about operating days should be output in the result. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 176: Description of structure TripRefineParamStructure

15.3 Response structures

The result of a refine request is sent via an element RefineResponse of the type RefineResponseStructure.

The element ErrorMessage is included in the relevant service-specific response structures (e.g. StopEventRefineResponse or TripRefineResponse) for determining possible error states.

In ErrorMessage, the following error states can appear:

Error code	**Error description**
REFINE_OBJECTNOTFOUND	The object to be refined could not be found in the separate database or could not be found unequivocally.

Table 177: List of error states in response structures of refine requests

15.3.1 RefineResponseStructure (in Trias_Refine.xsd)

RefineResponseStructure			*+Structure*	**Summarises the result data for a refine request.**
	IndividualRouteRefineResponse	0:*	*+IndividualRouteRefineResponse*	The individual routes refined. See 15.3.2.
	LocationInformationRefineResponse	0:*	*+LocationInformationRefineResponse*	The location information refined. See 15.3.3.
	StopEventRefineResponse	0:*	*+StopEventRefineResponse*	The stop events refined. See 15.3.4.
	TripInfoRefineResponse	0:*	*+TripInfoRefineResponse*	The trip information refined. See 15.3.5.
	TripRefineResponse	0:*	*+TripRefineResponse*	The trip information refined. See 15.3.6.

Table 178: Description of structure RefineResponseStructure

15.3.2 IndividualRouteRefineResponseStructure (in Trias_IndividualTrips.xsd)

IndividualRouteRefineResponseStructure			*+Structure*	**Summarises the result data for an individual routing refinement request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.
	IndividualRouteResponseContext	0:1	*+TripResponseContext*	Containers for data which appear multiple times in the response and are referenced. See 9.3.2.
	IndividualRouteResult	0:*	*+RouteResult*	Containers for an individual routing. See 17.3.2.

Table 179: Description of structure IndividualRouteRefineResponseStructure

15.3.3 LocationInformationRefineResponseStructure (in Trias_Locations.xsd)

LocationInformationRefineResponseStructure			*+Structure*	**Summarises the result data for a location information refinement.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.
	LocationResult	0:1	*+LocationResult*	Container for location information. See 8.4.2.

Table 180: Description of structure LocationInformationRefineResponseStructure

15.3.4 StopEventRefineResponseStructure (in Trias_StopEvents.xsd)

StopEventRefineResponseStructure			*+Structure*	**Summarises the result data for a stop event refinement.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.
	StopEventResponseContext	0:1	*+StopEventResponseContext*	Containers for data which appear multiple times in the response and are referenced. See 10.3.2.
	StopEventResult	0:*	*+StopEventResult*	Containers for stop events. See 10.3.3.

Table 181: Description of structure StopEventRefineResponseStructure

15.3.5 TripInfoRefineResponseStructure (in Trias_TripInfo.xsd)

TripInfoRefineResponseStructure			*+Structure*	**Summarises the result data for a trip information refinement.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.
	TripInfoResponseContext	0:1	*+TripInfoResponseContext*	Containers for data which appear multiple times in the response and are referenced. See 12.3.2.
	TripInfoResult	0:1	*+TripInfoResult*	Container for trip information. See 12.3.3.

Table 182: Description of structure TripInfoRefineResponseStructure

15.3.6 TripRefineResponseStructure (in Trias_Trips.xsd)

TripRefineResponseStructure			*+Structure*	**Summarises the result data for an intermodal trip information refinement.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.
	TripResponseContext	0:1	*+TripResponseContext*	Containers for data which appear multiple times in the response and are referenced. See 9.3.2.
	UnknownLegRef	0:*	→*xs:NMTOKEN*	References to the legs which could not be recovered in the local data
	TripResult	0:1	*+TripResult*	Container for trip information. See 9.3.3.

Table 183: Description of structure TripRefineResponseStructure

16 Dienst Buchungsinformation

16.1 Beschreibung

Der Dienst Buchungsinformation stellt Informationen zur Verfügung, mit deren Hilfe Kontakt zu einem Buchungssystem hergestellt werden kann. Das zuständige Buchungssystem kann für ein Verkehrsunternehmen oder für eine einzelne ÖV-Fahrt abgefragt werden. Ein Buchungssystem führt z. B. die Vorbestellung eines Bedarfsverkehrs, eine Sitzplatzreservierung oder auch den Kauf eines Fahrscheins durch.

In der XML-Schema-Definition *Trias_Booking.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Buchungsinformation verwendet werden.

16 Booking information service

16.1 Description

The booking information service provides information, with the help of which contact can be established with a booking system. The competent booking system can be requested for a transport company or individual public transport. For example, a booking system carries out advance booking of an on-demand transport, seat reservation or even purchase of a ticket.

Data types and structures are defined in the XML schema definition Trias_Booking.xsd which are used for the booking information service.

16.2 Request structures

Booking information is requested via an element BookingInfoRequest of the type BookingInfoRequestStructure.

16.2.1 BookingInfoRequestStructure

A certain public transport or a transport company can be optionally stated in a BookingInfoRequestStructure, for which the booking information should be ascertained.

BookingInfoRequestStructure				*+Structure*	**Summarises the request data as per booking information.**
	a	***Service***	**-1:1**	*+DatedJourney*	Definition of a public transport on a certain day. See 7.6.2.
	b	***OperatorRef***		*→Operator*	Reference to a transport company. See 7.4.1.
	Extension		0:1	*xs:anyType*	Extensions.

Table 184: Description of structure BookingInfoRequestStructure

16.3 Response structures

The result of a booking information request is sent via an element BookingInfoResponse of the type BookingInfoResponseStructure.

16.3.1 BookingInfoResponseStructure

BookingInfoResponseStructure			*+Structure*	**Summarises the result data for a booking information request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	BookingInfoResult	0:1	*+BookingInfoResult*	Structure for a booking information result. See 16.3.2.

Table 185: Description of structure ServiceRequestStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
BOOKINGINFO_DATEINVALID	There is no information available about the specified date.
BOOKINGINFO_VEHICLEUNKNOWN	The stated vehicle is unknown.
BOOKINGINFO_OPERATORUNKNOWN	The stated transport company is unknown.
BOOKINGINFO_JOURNEYUNKNOWN	The stated trip is unknown.
BOOKINGINFO_LINEUNKNOWN	The stated line is unknown.
BOOKINGINFO_MODEUNKNOWN	The stated mode is unknown.
BOOKINGINFO_NOINFORMATION	No suitable information is available.

Table 186: List of error states in BookingInfoResponse

16.3.2 BookingInfoResultStructure

BookingInfoResultStructure			*+Structure*	**Result structure for booking information.**
	BookingInfo	**1:***	*+BookingInfo*	Containers for booking information. See 7.10.4.
	Extension	0:1	*xs:anyType*	Extensions.

Table 187: Description of structure BookingInfoResultStructure

17 Dienst IV-Routing

In der XML-Schema-Definition *Trias_IndividualTrips.xsd* werden Datentypen und Strukturen definiert, die für ein IV-Routing verwendet werden.

17 Individual routing service

Data types and structures are defined in the XML schema definition Trias_IndividualTrips.xsd which are used for the individual routing.

17.1 Simple types

The following simple type is defined:

Type name	***Values***	**Description**
IndividualTripsAlgorithmTypeEnumeration	*fastest* | *shortest* | *beautiful* | *optimal* | *economic*	Algorithm type for calculating individual routes.

Table 188: List of simple type definitions in Trias_IndividualTrips.xsd

17.2 Request structures

A route in individual transport is requested via an element IndividualRouteRequest of the type IndividualRouteRequestStructure.

17.2.1 IndividualRouteRequestStructure

IndividualRouteRequestStructure			*+Structure*	**Summarises the request data for an individual routing.**
	Origin	**1:1**	*+IndividualRouteLocationContext*	Location data for point of departure. See 17.2.3.
	Destination	**1:1**	*+IndividualRouteLocationContext*	Location data for destination. See 17.2.3.
	Via	0:*	*+Via*	One or more via-locations. The via-locations specified must be reached in the specified sequence. The server may replace a via-stop by an equivalent stop. See 7.6.2.
	Mode	**1:***	*+IndividualTransportOptions*	Individual transport modes, for which an individual route should be ascertained. Additional controlling parameters can be stated for every individual transport mode. See 7.3.2.
	Params	0:1	*+IndividualTripParam*	Parameters which can affect the search and return values. See 17.2.2.

Table 189: Description of structure IndividualRouteRequestStructure

17.2.2 IndividualTripParamStructure

IndividualTripParamStructure			*+Structure*	**Summarises the parameters which can influence individual route search and return values. These parameters are applicable to all individual transport modes, for which individual routing should be carried out. If different parameter sets are supposed to be used for different individual transport modes, several independent individual route searches must be carried out.**
Base-TripMo-bilityFil-ter	*NoSingleStep*	0:1	*xs.boolean*	Defines whether the user can use steps. Default is *false*.
	NoStairs	0:1	*xs.boolean*	Defines whether the user can use stairs. Default is *false*.
	NoEscalator	0:1	*xs.boolean*	Defines whether the user can use an escalator. Default is *false*.
	NoElevator	0:1	*xs.boolean*	Defines whether the user can use the elevator. Default is *false*.
	NoRamp	0:1	*xs.boolean*	Defines whether the user can use a ramp. Default is *false*.
BaseTripPolicy				
	a *NumberOfResults*	-0:1	*xs:positiveInteger*	Number of trip information results which the user expects as a minimum.
	b *:::*	-0:1	*NumberOfResultsGroup*	Specification of the desired trips before/after the stated time at start or end (see 9.2.3).
	IgnoreRealtimeData	0:1	*xs:boolean*	If this parameter is set, real-time data or error information should not be considered in the trip search but only target trip data. Default is *false*.
	ImmediateTripStart	0:1	*xs:boolean*	If this parameter is set, the trip to be searched should directly begin at the start situation specified. Optimisation of the departure time at the start is generally not necessary according to the rule “Start as late as possible only if the exact arrival time is ensured at the destination”. Default is *false*.
IndividualTripPolicy	*AlgorithmType*	0:1	*fastest* | *shortest* | *beautiful* | *optimal* | *economic*	Type of target function, according to which the routing algorithm should optimise the route. If not specified, the service uses its own default setting.
	BanMotorways	0:1	*xs:boolean*	If set, motorways should not be used in individual transport routing. Default is *false*.
	BanTollRoads	0:1	*xs:boolean*	If set, toll roads should not be used in individual transport routing. Default is *false*.
	BanFerries	0:1	*xs:boolean*	If set, ferries or ships should not be used in individual transport routing. Default is *false*.
	BanTunnels	0:1	*xs:boolean*	If set, tunnels (but not underbridges) should not be used in individual transport routing. The individual transport route service defines the difference between a tunnel and an underbridge. Default is *false*.
	BanBridges	0:1	*xs:boolean*	If set, large bridges (but not underbridges) should not be used in individual transport routing. The individual transport route service defines which bridges are considered “large”. Default is *false*.

	AllowUnpavedRoads	0:1	*xs:boolean*	If set, unpaved roads may be used in individual transport routing. Otherwise, they may not. Default is *false*.
BaseTripContentFilter	*IncludeTrackSections*	0:1	*xs:boolean*	Specifies whether TrackSection element (see 7.6.15) should be output in the result for a detailed geographic description of the route. Default is *false*.
	IncludeLegProjection	0:1	*xs:boolean*	Specifies whether the detailed geographic route should be output in the result as a coordinate sequence. Default is *false*.
	IncludeTurnDescription	0:1	*xs:boolean*	Specifies whether route information should be output in the result with turn recommendations. Default is *false*.
	IncludeAccessibility	0:1	*xs:boolean*	Specifies whether information about barrier freedom should be output in the result. Default is *false*.
	IncludeEstimatedTimes	0:1	*xs:boolean*	Specifies whether information about real-time situation should be output in the result. Default is *false*.
	IncludeSituationInfo	0:1	*xs:boolean*	Specifies whether textual real-time messages should be output in the result. Default is *false*.
	Extension	0:1	*xs:anyType*	Extensions.

Table 190: Description of structure IndividualTripParamStructure

17.2.3 IndividualRouteLocationContextStructure

Elements of the type IndividualRouteLocationContextStructure are used to describe the start or end context which should be assumed for the passenger at the beginning or end of his trip. Elements of this type define the start and end location within the individual transport routing service. Here, the implementation of the search algorithm is responsible to map the location details (e.g. a coordinate) to the internal elements (e.g. nodes and edges) of the search network.

IndividualRouteLocationContextStructure			*+Structure*	**Location specification for start and end locations of individual transport routes.**
	LocationRef	**1:1**	*+LocationRef*	Reference to a location object. See 7.5.11.
	DepArrTime	0:1	*xs:dateTime*	Departure or arrival time.

Table 191: Description of structure IndividualRouteLocationContextStructure

17.3 Response structures

The result of an individual routing request is sent via an element IndividualRouteResponse of the type IndividualRouteResponseStructure.

17.3.1 IndividualRouteResponseStructure

IndividualRouteResponseStructure			*+Structure*	**Summarises the result data for an individual routing information.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. Also see 7.4.2.
	IndividualRouteResponseContext	0:1	*+TripResponseContext*	Containers for data which appear multiple times in the response and are referenced. See 9.3.2.
	RouteResult	0:*	*+RouteResult*	Container for trip information. See 17.3.2.

Table 192: Description of structure IndividualRouteResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
ROUTE_NOROUTEFOUND	Individual transport routes could not be found with regard to the start locations and destinations stated, the desired departure and arrival time and keeping in mind the specified parameters.
ROUTE_ORIGINUNKNOWN	The specified location (address, stop etc.) for the start of the individual route is unknown.
ROUTE_DESTINATIONUNKNOWN	The specified location (address, stop etc.) for the destination of the individual route is unknown.
ROUTE_VIAUNKNOWN	One of the via-points specified is unknown.
ROUTE_ORIGINDESTINATIONIDENTICAL	Start and destination are identical.
ROUTE_UNSUPPORTEDMODE	One of the requested individual modes is not supported.
ROUTE_UNSUPPORTEDMOBILITYFILTER	One of the requested mobility filters is not supported.
ROUTE_UNSUPPORTEDALGORITHM	The requested algorithm type is not supported.
ROUTE_UNSUPPORTEDBAN	One of the requested ban filters (motorways, toll roads, ferries) is not supported.
ROUTE_NODATETIME	Neither the departure time nor the arrival time has been stated.
ROUTE_DATETIMEERROR	Date and/or time are incomprehensible.
ROUTE_DEPARTUREAFTERARRIVAL	The desired departure time at all the start points is after the desired arrival time at al destination points.
ROUTE_DATEOUTOFRANGE	Routing data is not available for the requested date, e.g. because the date is in the past or far into the future.

Table 193: List of error states in IndividualRouteResponse

17.3.2 RouteResultStructure

RouteResultStructure			*+Structure*	**Summarises the result data for an individual transport route.**
	ResultId	**1:1**	*xs:NMTOKEN*	ID of the result for subsequent referencing or debugging purposes.
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on this individual transport route. Refer to the following table for possible values. Also see 7.4.2.
	Route	**1:1**	*+Route*	Data about an individual route. See 17.3.3.

Table 194: Description of structure RouteResultStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
ROUTE_MODEPARAMETERSIGNORED	At least one of the parameters for this individual mode has been ignored for this individual transport route. Also see 7.3.2.
ROUTE_MOBILITYFILTERIGNORED	At least one of the mobility filters has been ignored for this individual transport route.
ROUTE_BANIGNORED	At least one of the ban filters (motorways, toll roads, ferries) has been ignored for this individual transport route.

Table 195: List of error states in RouteResult

17.3.3 RouteStructure

RouteStructure			*+Structure*	**Data about an individual transport route.**
	RouteId	**1:1**	*xs:NMTOKEN*	ID of the trip for subsequent referencing or debugging purposes.
	Duration	**1:1**	*xs:duration*	Total duration of individual transport route.
	StartTime	**1:1**	*xs:dateTime*	Start time of individual transport route.
	EndTime	**1:1**	*xs:dateTime*	End time of individual transport route.
	Distance	0:1	*Distance*	Total distance of the individual transport route as length of the route to be covered.
	RouteLeg	**1:***	*+ContinuousLeg*	Legs of this individual transport route. There must exist exactly one more RouteLeg than what is requested as Vias. Only *IndividualMode* and, if necessary, *SituationFullRef* is filled in *ContinuousLeg.Service* for individual transport routes. See 9.3.8.
	SituationFullRef	0:*	*+SituationFullRef*	Reference to error messages. These messages can be found in the element *IndividualRouteResponseContext* of the type *TripResponseContext* (see 9.3.2) or made known through other channels. See 7.8.2.
	Extension	0:1	*xs:anyType*	Extensions.

Table 196: Description of structure RouteStructure

18 Kartendienst

In der XML-Schema-Definition *Trias_Maps.xsd* werden Datentypen und Strukturen definiert, die für den Kartendienst verwendet werden.

18 Map service

Data types and structures are defined in the XML schema definition Trias_Maps.xsd which are used for the map service.

18.1 Simple data types

The following simple types are defined:

Type name	***Values***	**Description**
MapLayersEnumeration	*physical* | *satellite* | *street* | *rail* | *names* | *stops* | *traffic*	Additional layers in the map.

Table 197: List of simple type definitions in Trias_Maps.xsd

18.2 Request structures

18.2.1 MapServiceRequestStructure

A map is requested via an element MapServiceRequest of the type MapServiceRequestStructure. It returns an image file which contains the requested map. If additional objects are needed to be plotted on the maps (e.g. stops) or active elements integrated (e.g. for Mouse-Over effects, linking etc.), this must be done by the client on the basis of a background map generated by the map service.

MapServiceRequestStructure			*+Structure*	**Summarises the request data for calling a map.**
MapProperties	***Aspect***	**1:1**	*+MapAspect*	Geographic section of the map to be generated. The map generated may cover an additional section other than the one specified. However, the midpoint must remain almost unchanged and the actual map section should be as similar to the requested section as possible. See 18.2.4.
	Size	**1:1**	*+MapSize*	Image size of the map to be generated. The map service must be in the position to generate map sizes up to minimum 1920x1080 pixels ("FullHD"). The map generated must match the specified image size precisely. See 18.2.5.
	ImageType	0:1	*xs:string*	Data format of the map to be generated. It must be specified as media type (formerly called MIME type) of an image data format (subtypes of the type "image"). The list of permitted values is defined by IANA. If not specified, the map service must use "image/png".
	Layer	**1:***	*physical | satellite | street | rail | names | stops | traffic*	Specifies the layers of the map. This includes the background map, as well as additional information which is supposed to be integrated on the map.
	Opaqueness	0:1	*Percent*	Opaqueness of the map background if no background layer has been selected. Between 0 (completely clear) and 100 (completely opaque).
	BackgroundColor	0:1	*xs:string*	Colour of the map background if no background layer has been selected. Permitted values are all the colour values which meet the CSS3 standard of W3C.
	Extension	0:1	*xs:anyType*	Extensions.

Table 198: Description of structure MapServiceRequestStructure

18.2.2 ImageCoordinatesRequestStructure

In a few use cases, geographic objects are supposed to be plotted on a map or made available as an active, and if necessary movable, object. In order to achieve this, the client must subsequently plot the object on the map image. In order to do this, it is helpful to be able to generate image coordinates from the geographic coordinates of an object. Such image coordinates are requested using an element ImageCoordinatesRequest of the type ImageCoordinatesRequestStructure.

ImageCoordinatesRequestStructure			*+Structure*	**Summarises the request data for calling image coordinates.**
MapPro perties	***Aspect***	**1:1**	*+MapAspect*	Actual geographic section of the referenced map. See 18.2.4.
	Size	**1:1**	*+MapSize*	Image size of the referenced map. See 18.2.5.
	Point	**1:***	*+GeoPosition*	Geographic points, for which the image coordinates should be calculated. They may lie outside the specified map section but a map service can reject processing of points that lie too far outside. See 7.2.3.

Table 199: Description of structure ImageCoordinatesRequestStructure

18.2.3 GeoCoordinatesRequestStructure

In order to be able assign a corresponding geographic position to a position on a map image (e.g. after clicking on a map), a suitable conversion function is required. Such geographic positions are requested using an element GeoCoordinatesRequest of the type GeoCoordinatesRequestStructure.

GeoCoordinatesRequestStructure			*+Structure*	**Summarises the request data for calling geographic coordinates.**
MapPro perties	***Aspect***	**1:1**	*+MapAspect*	Actual geographic section of the referenced map. See 18.2.4.
	Size	**1:1**	*+MapSize*	Image size of the referenced map. See 18.2.5.
	ImagePoint	**1:***	*+MapCoordin ate*	Image points, for which the geographic coordinates should be calculated. They can lie outside the map image but a map service can reject processing of points that lie too far outside. See 18.2.6.

Table 200: Description of structure GeoCoordinatesRequestStructure

18.2.4 MapAspectStructure

MapAspectStructure			*+Structure*	**Geographic map section.**
	UpperLeft	**1:1**	*+GeoPosition*	Upper left corner of the geographic map section. See 7.2.3.
	LowerRight	**1:1**	*+GeoPosition*	Lower right corner of the geographic map section. See 7.2.3.

Table 201: Description of structure MapAspectStructure

18.2.5 MapSizeStructure

MapSizeStructure			*+Structure*	**Image size of a map.**
	Width	**1:1**	*xs:nonNegativeInteger*	Width of map in pixels.
	Height	**1:1**	*xs:nonNegativeInteger*	Height of map in pixels.

Table 202: Description of structure MapSizeStructure

18.2.6 MapCoordinateStructure

MapCoordinateStructure			*+Structure*	**Coordinates of an image point.**
	X	**1:1**	*xs:integer*	X-coordinate. Can be negative or greater than the width of the basic image.
	Y	**1:1**	*xs:integer*	Y-coordinate. Can be negative or greater than the width of the basic image.

Table 203: Description of structure MapCoordinateStructure

18.3 Response structures

18.3.1 MapServiceResponseStructure

The result of a map request is sent via an element MapServiceResponse of the type MapServiceResponseStructure.

MapServiceResponseStructure			*+Structure*	**Summarises the result data for a map request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	MapResult	0:1	*+MapResult*	Result of map request.

Table 204: Description of structure MapServiceResponseStructure

In ErrorMessage, the following error states can appear:

MAP_UNSUPPORTEDSIZE	The requested image size is not supported.
MAP_UNSUPPORTEDMEDIATYPE	The requested media type (former MIME type) is not supported.
MAP_UNSUPPORTEDASPECT	The requested map section lies outside the area supported by the map service, is too large or is too small.
MAP_LAYERIGNORED	At least one of the requested layers was ignored.
MAP_UNSUPPORTEDSTYLE	The map service does not support the specification of a background colour or opaqueness.
MAP_NOMAP	A map in accordance with the requested could not be generated.

Table 205: List of error states in MapServiceResponse

18.3.2 ImageCoordinatesResponseStructure

The result of an image coordinate request is sent via an element ImageCoordinatesResponse of the type ImageCoordinatesResponseStructure.

ImageCoordinatesResponseStructure			*+Structure*	**Summarises the result data for an image coordinate request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	Result	0:*	*+ImagePointResult*	Individual results of the conversion from geographic to image coordinates. There can be maximum as many elements as there were geographical points in the request. For identification, the requested point must be included in every individual result. See 18.3.5.

Table 206: Description of structure ImageCoordinatesResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
MAP_UNSUPPORTEDSIZE	The requested image size is not supported.
MAP_UNSUPPORTEDASPECT	The requested map section lies outside the area supported by the map service, is too large or is too small.
MAP_TOOMANYPOINTS	Too many point objects were specified for conversion.
MAP_UNSUPPORTEDPOINT	At least one of the requested points lies outside the convertible area.

Table 207: List of error states in ImageCoordinatesResponse

18.3.3 GeoCoordinatesResponseStructure

The result of a geographic coordinate request is sent via an element GeoCoordinatesResponse of the type GeoCoordinatesResponseStructure.

GeoCoordinatesResponseStructure			*+Structure*	**Summarises the result data for a geographic coordinate request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	Result	0:*	*+GeoCoordinateResult*	Individual results of the conversion from image to geographic coordinates. There can be maximum as many elements as there were image points in the request. For identification, the requested point must be included in every individual result. See 18.3.6.

Table 208: Description of structure GeoCoordinatesResponseStructure

In ErrorMessage, the following error states can appear:

Error code	Error description
MAP_UNSUPPORTEDSIZE	The requested image size is not supported.
MAP_UNSUPPORTEDASPECT	The requested map section lies outside the area supported by the map service, is too large or is too small.
MAP_TOOMANYPOINTS	Too many point objects were specified for conversion.
MAP_UNSUPPORTEDPOINT	At least one of the requested points lies outside the convertible area.

Table 209: List of error states in GeoCoordinatesResponse

18.3.4 MapResultStructure

MapResultStructure			*+Structure*	**Summarises the data of a generated map.**
	File	**1:1**	*xs:base64Binary*	The image data of the map generated.
	ImageType	**1:1**	*xs:string*	Data format of the map generated. It must be specified as media type (formerly called MIME type) of an image data format (subtypes of the type “image”). The list of permitted values is defined by IANA.
	Aspect	**1:1**	*+MapAspect*	Actual geographic section of the map generated. It may differ from the requested map section. However, the midpoint must remain almost unchanged and the actual map section should be as similar to the requested section as possible. See 18.2.4.

Table 210: Description of structure MapResultStructure

18.3.5 ImagePointResultStructure

ImagePointResultStructure			*+Structure*	**Summarises the data of a conversion from geographic to image coordinates.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the conversion of the following geographic coordinate pair. Refer to the following table for possible values. See 7.4.2.
	Point	**1:1**	*+GeoPosition*	Geographic point, for which the image coordinates should be calculated. It must be one of the points from the associated service request. See 7.2.3.
	ImagePoint	0:1	*+MapCoordinate*	Image coordinates for the requested geographic point if the conversion could be carried out. The image point can lie outside the basic map image (including negative coordinate values). If this element is missing, at least one error code must be set. See 18.2.6.

Table 211: Description of structure ImagePointResultStructure

In ErrorMessage, the following error states can appear:

Error code	Error description
MAP_POINTNOTONMAP	The specified point lies outside the map. Nevertheless a conversion could be carried out.
MAP_UNSUPPORTEDPOINT	The requested point lies outside the convertible area.

Table 212: List of error states in ImagePointResultStructure

18.3.6 GeoCoordinateResultStructure

GeoCoordinateResultStructure			*+Structure*	**Summarises the data of a conversion from image to geographic coordinates.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the conversion of the following geographic coordinate pair. Refer to the following table for possible values. See 7.4.2.
	Point	0:1	*+GeoPosition*	Geographic coordinates for the requested image point should be calculated if the conversion could be carried out. If this element is missing, at least one error code must be set. See 7.2.3.
	ImagePoint	**1:1**	*+MapCoordinate*	Image coordinates for which geographical coordinates should be calculated. It must be one of the points from the associated service request. See 18.2.6.

Table 213: Description of structure GeoCoordinateResultStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
MAP_POINTNOTONMAP	The specified point lies outside the map. Nevertheless a conversion could be carried out.
MAP_UNSUPPORTEDPOINT	The requested point lies outside the convertible area.

Table 214: List of error states in GeoCoordinateResultStructure

19 Dienst Schadensmeldung / Zustand von Einrichtungen

19.1 Beschreibung

Dieser Dienst erlaubt es, den Zustand einer Haltestelleneinrichtung oder Fahrzeugausstattung abzufragen oder zu melden (Schadensmeldung).

In der XML-Schema-Definition *Trias_Facilities.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Schadensmeldung / Zustand von Einrichtungen verwendet werden.

19 Damage report / condition of facility service

19.1 Description

This service makes it possible to enquire or report the status of a stop facility or vehicle facility (damage report).

Data types and structures are defined in the XML schema definition Trias_Facilities.xsd which are used for the damage report / condition of facility service.

19.2 Simple types

The following simple types are defined:

Type name	***Values***	**Description**
FacilityStatusTypeEnumeration	*OK | dirty | destroyed | damaged | stolen | out of order*	Status of equipment.
FacilityAvailabilityEnumeration	*unknown | available | notAvailable | partiallyAvailable | added | removed*	Availability of facility.

Table 215: Description of structure simple types

19.3 Complex structures

19.3.1 FacilityStructure

FacilityStructure				*+Structure*	**Description and condition of a facility.**
	a	***VehicleFacility***	**-1:1**	*+VehicleFacility*	Definition of a vehicle facility. See 19.3.2.
	b	***InfrastructureFacility***		*+Infrastructure Facility*	Definition of an infrastructure facility. See 19.3.3.
	Condition		**1:1**	*+FacilityStatus*	Status of equipment. See 19.3.7.
	Extension		0:1	*xs:anyType*	Extension.

Table 216: Description of structure FacilityStructure

19.3.2 VehicleFacilityStructure

VehicleFacilityStructure			*+Structure*	**Description of a vehicle facility.**
Vehicle Facility Ref	***:::***	**1:1**	*+VehicleFacilityRefGroup*	Reference to a facility. See 19.3.4.
Service Facility	***:::***	**1:1**	*+ServiceFacilityGroup*	Classification of facility. See 7.7.3.
	FacilityDescription	0:*	*+International Text*	Name or description of facility. See 7.2.2.
	LocationDescription	0:*	*+International Text*	Description of where the facility can be found. See 7.2.2.

Table 217: Description of structure VehicleFacilityStructure

19.3.3 InfrastructureFacilityStructure

InfrastructureFacilityStructure			*+Structure*	**Description of an infrastructure facility.**
InfrastructureFacilityRef	***:::***	**1:1**	*+InfrastructureFacilityRefGroup*	Reference to a facility. See 19.3.5.
StopFacility	***:::***	**1:1**	*+StopFacility Group*	Classification of facility. See 7.2.2.
	FacilityDescription	0:*	*+International Text*	Name or description of facility. See 7.2.2.
	Location	0:1	*+GeoPosition*	Coordinate position of facility. See 7.2.3.
	LocationDescription	0:*	*+International Text*	Description of where the facility can be found. See 7.2.2.

Table 218: Description of structure InfrastructureFacilityStructure

19.3.4 VehicleFacilityRefGroup

VehicleFacilityRefGroup			*+Group*	**Referencing of a vehicle facility by referencing the facility itself or a parent object.**
	FacilityRef	0:1	*→FacilityCode*	Reference to a facility. See 7.4.1.
	OwnerRef		*→OwnerCode*	Reference to owner. See 7.4.1.
	OperatorRef		*→Operator Code*	Reference to transport companies. See 7.4.1.
	LineRef		*→LineCode*	Reference to a line. See 7.4.1.
	JourneyRef		*→JourneyCode*	Reference to a journey. See 7.4.1.
	VehicleRef		*→VehicleCode*	Reference to a vehicle. See 7.4.1.

Table 219: Description of group VehicleFacilityRefGroup

19.3.5 InfrastructureFacilityRefGroup

InfrastructureFacilityRefGroup			*+Group*	**Referencing of an infrastructure facility by referencing the facility itself or a parent object.**
	FacilityRef	0:1	→*FacilityCode*	Reference to a facility. See 7.4.1.
	OwnerRef		→*OwnerCode*	Reference to owner. See 7.4.1.
	StopPointRef		→*StopPoint*	Reference to a stopping point. See 7.5.1.
	StopPlaceRef		→*StopPlace*	Reference to a stop. See 7.5.1.
	OperatorRef		→*OperatorCode*	Reference to transport companies. See 7.4.1.
	LineRef		→*LineCode*	Reference to a line. See 7.4.1.

Table 220: Description of group InfrastructureFacilityRefGroup

19.3.6 FacilityDataFilterGroup

FacilityDataFilterGroup			*+Group*	**A list of object references as filter for limiting the facilities in question.**
	FacilityRef	0:*	→*FacilityCode*	Reference to a facility. See 7.4.1.
	OwnerRef	0:*	→*OwnerCode*	Reference to owner. See 7.4.1.
	StopPointRef	0:*	→*StopPoint*	Reference to a stopping point. See 7.5.1.
	StopPlaceRef	0:*	→*StopPlace*	Reference to a stop. See 7.5.1.
	OperatorRef	0:*	→*OperatorCode*	Reference to transport companies. See 7.4.1.
	LineRef	0:*	→*LineCode*	Reference to a line. See 7.4.1.
	JourneyRef	0:*	→*JourneyCode*	Reference to a journey. See 7.4.1.
	VehicleRef	0:*	→*VehicleCode*	Reference to a vehicle. See 7.4.1.

Table 221: Description of group FacilityDataFilterGroup

19.3.7 FacilityStatusStructure

FacilityStatusStructure			*+Structure*	**The condition of a facility.**
	Availability	**1:1**	*FacilityAvailabilityEnumeration*	Availability of facility. See 19.2
	Status	**1:1**	*FacilityStatusTypeEnumeration*	Classification of condition. See 19.2
	StatusDescription	0:*	*+InternationalText*	Description of condition. See 7.2.2.

Table 222: Description of structure FacilityStatusStructure

19.4 Damage report request

Report of a damage to a facility is sent by a person via the element FacilityStatusReport of the type FacilityStatusReportStructure.

19.4.1 FacilityStatusReportStructure

FacilityStatusReportStructure				*+Structure*	**Summarises the data for a damage report of a facility.**
	a	***VehicleFacility***	**-1:1**	*+VehicleFacility*	Definition of a vehicle facility. See 19.3.2.
	b	***InfrastructureFacility***		*+Infrastructure Facility*	Definition of an infrastructure facility. See 19.3.3.
	Condition		**1:1**	*+FacilityStatus*	Status of equipment. See 19.3.7.
	Extension		0:1	*xs:anyType*	Extension.

Table 223: Description of structure FacilityStatusReportStructure

19.5 Damage report response

The response to a damage report (FacilityStatusReport, see 19.4) is sent via an element FacilityStatusReportResponse of the type FacilityStatusReportResponseStructure.

19.5.1 FacilityStatusReportResponseStructure

FacilityStatusReportResponseStructure			*+Structure*	**Response to damage report.**
	ErrorMessage	**1:1**	*+ErrorMessage*	Error message. Refer to the following table for possible values. See 7.4.2.

Table 224: Description of structure FacilityStatusReportResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
FACILITYSTATUSREPORT_FACILITYUNKNOWN	The stated facility is unknown.
FACILITYSTATUSREPORT_OWNERUNKNOWN	The stated owner is unknown.
FACILITYSTATUSREPORT_OPERATORUNKNOWN	The stated transport company is unknown.
FACILITYSTATUSREPORT_LINEUNKNOWN	The stated line is unknown.
FACILITYSTATUSREPORT_JOURNEYUNKNOWN	The stated trip is unknown.
FACILITYSTATUSREPORT_VEHICLEUNKNOWN	The stated vehicle is unknown.
FACILITYSTATUSREPORT_STOPPOINTUNKNOWN	The stated stopping point is unknown.
FACILITYSTATUSREPORT_STOPPLACEUNKNOWN	The stated stop is unknown.

Table 225: List of error states in FacilityStatusReportResponse

19.6 Request of condition of facilities

A request of the current status of facilities is sent via an element FacilityRequest of the type FacilityRequestStructure.

19.6.1 FacilityRequestStructure

FacilityRequestStructure			*+Structure*	**Summarises the data for a damage report of a facility.**
Facility DataFilter	***:::***	**1:1**	*+FacilityDataFilterGroup*	Object references as filter. See 19.3.6.
	Extension	0:1	*xs:anyType*	Extension.

Table 226: Description of structure FacilityRequestStructure

19.7 Response to condition of facilities

The response to a condition report is sent via an element FacilityResponse of the type FacilityResponseStructure.

19.7.1 FacilityResponseStructure

FacilityResponseStructure			*+Structure*	**Response to damage report.**
	ErrorMessage	0:*	*+ErrorMessage*	Error message based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	FacilityResult	0:*	*+FacilityResult*	Result structure. See 19.7.2.

Table 227: Description of structure FacilityResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
FACILITYREQUEST_FACILITYUNKNOWN	The stated facility is unknown.
FACILITYREQUEST_OWNERUNKNOWN	The stated owner is unknown.
FACILITYREQUEST_OPERATORUNKNOWN	The stated transport company is unknown.
FACILITYREQUEST_LINEUNKNOWN	The stated line is unknown.
FACILITYREQUEST_JOURNEYUNKNOWN	The stated trip is unknown.
FACILITYREQUEST_VEHICLEUNKNOWN	The stated vehicle is unknown.
FACILITYREQUEST_STOPPOINTUNKNOWN	The stated stopping point is unknown.
FACILITYREQUEST_STOPPLACEUNKNOWN	The stated stop is unknown.

Table 228: List of error states in FacilityResponse

19.7.2 FacilityResultStructure

FacilityResultStructure			*+Structure*	**Result structure for facility status request.**
	ErrorMessage	0:*	*+ErrorMessag e*	Error messages based on the individual FacilityResult. Refer to the following table for possible values. See 7.4.2.
	Facility	**1:1**	*+Facility*	Information about facility. See 19.3.1

Table 229: Description of structure FacilityResultStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
FACILITYREQU-EST_STATUSNOTCONFIRMED	A status report is available for the facility but it is not yet confirmed.

Table 230: List of error states in FacilityResultStructure

20 Benachrichtigungsdienst

20.1 Beschreibung

Der Benachrichtigungsdienst dient der aktiven Benachrichtigung von Benutzern über aktuelle Geschehnisse. Die Benutzer können Abonnements einrichten, um sich bei Auftreten neuer Informationen benachrichtigen zu lassen

Der Dienst informiert unter anderem über

- Geplante Maßnahmen, Störungen oder Ereignisse auf einer Verbindung, einer Strecke oder an einer Haltestelle,
- alternative Verbindungen (aufgrund von Störungen oder der Prozessdatenlage),
- den Status eines Anschlusses und zusätzliche Informationen bei Anschlussverlust.

Diese Aufzählung ist nicht abschließend.

Dabei verfügt der Benachrichtigungsdienst über Funktionalität, die es ihm ermöglicht, betroffene Objekte (Verbindungen, Anschlüsse etc.) zu ermitteln. Teile des Benachrichtigungsdienstes (Ereignismeldungen und Änderungen an der Fahrzeugausstattung bzw. an Haltestelleneinrichtungen) wurden aus dem SIRI-Standard übernommen.

Eine Benachrichtigung besteht aus einer eindeutigen ID, einem Typ und den Nutzdaten. Folgende Arten von Abonnements können eingerichtet werden (vgl. auch die allgemeine Beschreibung von Abonnementanfragen in 7.1.2):

- SituationExchangeSubscriptionRequest (aus SIRI SX):
  Benachrichtigung über Ereignisse und Störungen,
- FacilityMonitoringSubscriptionRequest (aus SIRI FM):
  Änderungen an der Fahrzeugausstattung bzw. an Haltestelleneinrichtungen,
- TripMonitoringSubscriptionRequest (neu in TRIAS):
  Überwachung einer geplanten Verbindung.

Die Funktionsweise des Benachrichtigungsdienstes ist in Abbildung 6 zu sehen. Um Nachrichten zu erhalten, muss ein Portalsystem ein Abonnement beim Benachrichtigungsdienst erstellen (1). Bei der Erstellung des Abonnements (SubscriptionRequest) kann angegeben werden, welche Typen von Nachrichten dem Portalsystem übermittelt werden sollen. Die Erstellung des Abonnements wird vom Benachrichtigungsdienst synchron mit einer siri:SubscriptionResponse beantwortet.

Bei jeder neuen Meldung von einem Datenlieferant (2) ermittelt der Benachrichtigungsdienst die betroffenen Abonnements und leitet die Nachrichten an das zugehörige Portalsystem weiter. (3) Eine Nachricht vom Benachrichtigungsdienst, die eine Datenlieferung (ServiceDelivery) im Rahmen eines bestehenden Abonnements überliefert (z.B. als TripMonitoringDelivery), wird vom Portalsystem synchron mit einer DataReceivedAcknowledgement-Nachricht bestätigt.

Soll das Portalsystem keine neuen Nachrichten mehr empfangen, so muss es das Abonnement am Benachrichtigungsdienst abmelden (4) (oder das Abonnement erlischt von selbst nach Ablauf des Gültigkeitszeitraums).

Eine Aktualisierung des Abonnements durch das Portalsystem (z.B. wegen geänderter TripMonitoringParam) ist nicht vorgesehen, stattdessen muss das Abonnement abgemeldet (4) und neu eingerichtet (1) werden.

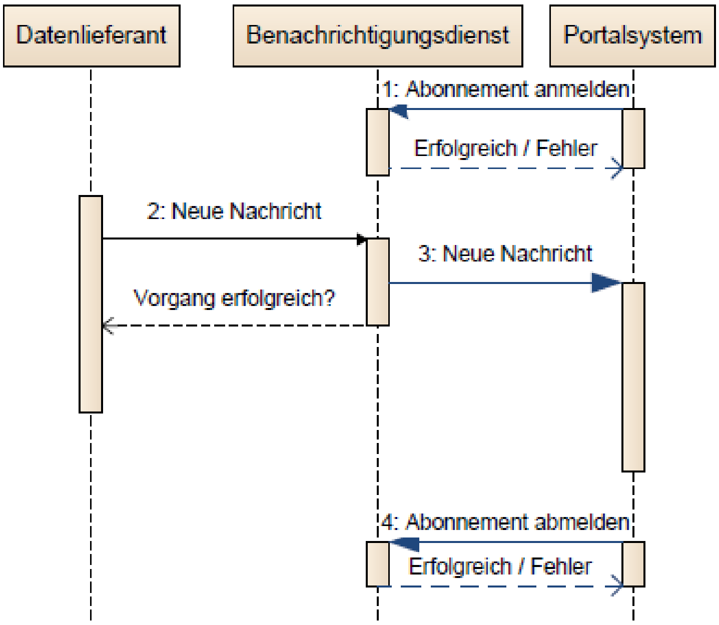

Abbildung 6: Funktionsweise des Benachrichtigungsdienstes

In den XML-Schema-Definitionen *Trias.xsd* und *Trias_Alerts.xsd* werden Datentypen und Strukturen definiert, die für den Benachrichtigungsdienst verwendet werden.

20 Notification service

20.1 Description

The notification service helps in actively notifying users about the current events. The users can set up subscriptions in order get notified of new information

The service informs about

- Planned measures, faults or events on a trip, track or at a stop,
- alternative routes (due to faults or process data situation),
- the status of a connection and additional information in case of a missed connection. This list is not exhaustive.

Here, the notification service has a functionality which enables it to determine relevant objects (links, connections etc.). Parts of the notification service (event messages and changes to vehicle facilities or stop facilities) have been taken from the SIRI standard.

The notification consists of a unique ID, a type and user data. The following types of subscriptions can be set up (also see the general description of subscription requests in 7.1.2).

- SituationExchangeSubscriptionRequest (from SIRI SX): Notification of events and faults,
- FacilityMonitoringSubscriptionRequest (from SIRI FM):
- Changes to vehicle facility or stop facilities,
- TripMonitoringSubscriptionRequest (new in TRIAS): Monitoring a planned trip.

The functioning of the notification service can be seen in figure 6 . In order to receive messages, a portal system must create a subscription for the notification service (1). When creating the subscription (SubscriptionRequest) which types of messages should be sent to the portal system can be specified. The notification service synchronously responds to the subscription creation with a siri:SubscriptionResponse.

With each new notification from a data provider (2), the notification service ascertains the subscriptions concerned and passes the messages on to the associated portal system. (3) A message from the notification service, which the data delivery (ServiceDelivery) passes on within the scope of an existing subscription (e.g. as TripMonitoringDelivery), is confirmed by the portal system synchronously with a DataReceivedAcknowledgement message.

If the portal system does not receive new messages, it must cancel the subscription of the notification service (4) (or the subscription expires on its own after the end of the validity period).

Updating the subscription by the portal system (e.g. due to modified TripMonitoringParam) is not planned. Instead the subscription must be cancelled (4) and set up again (1).

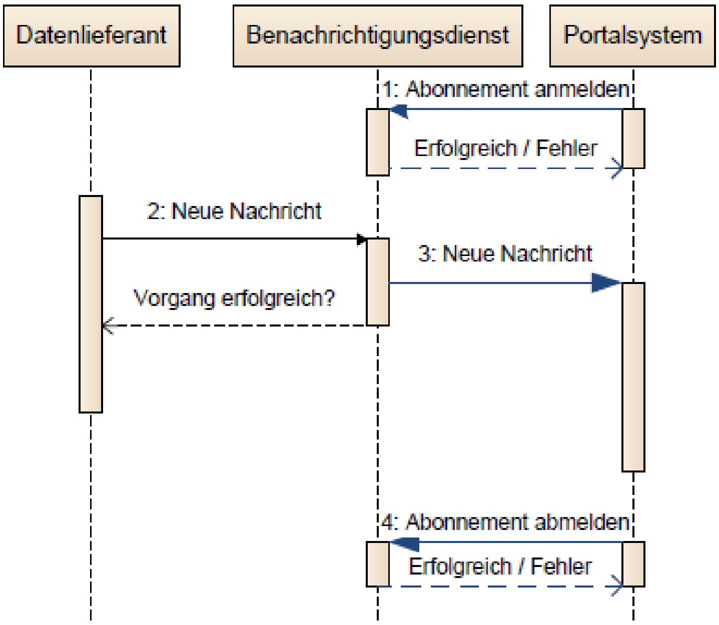

Figure 6: The functioning of the notification service

Data types and structures are defined in the XML schema definitions Trias.xsd and Trias_Alerts.xsd which are used for the notification service.

20.2 Complex structures

20.2.1 TripMonitoringParamStructure

TripMonitoringParamStructure			*+Structure*	**Parameters for trip monitoring.**
TripMonitoringPolicy	*Severity*	0:1	*unknown* | *verySlight* | *slight* | *normal* | *severe* | *verySevere* | *noImpact* | *undefined*	Priorities of events (as per TPEG table 26).
	MinimumDelayChangeThreshold	0:1	*xs:duration*	Delay change, after which notification is sent again.
	AcceptThirdPartyInformation	0:1	*xs:boolean*	Specifies which information should be fetched by other platforms. Default is *false*.
	IncludeAlternatives	0:1	*xs:boolean*	Specifies whether direct alternatives should also be returned. Default is *false*.

Table 231: Description of structure TripMonitoringParamStructure

The parameter AcceptThirdPartyInformation specifies whether the requested EKAP should create the response purely out of its own information sources or whether it must incorporate (missing) information from other data sources such as other EKAPs. This way, the requested platform can control the origin of data. Hence, the requester himself can either request different platforms for the missing information or let this function get executed by the requested system. The parameter is meaningful particularly if several EKAPs are interconnected and have different (geographic) responsibilities.

20.3 Request structures

20.3.1 TripMonitoringSubscriptionRequestStructure

TripMonitoringSubscriptionRequestStructure			*+Structure (derived from siri:AbstractSubscriptionStructure)*	**Setting up trip monitoring.**
	Trip	**1:1**	*+Trip*	Trip to be monitored. See 9.3.4.
	TripRequest	0:1	*+TripRequest*	Original trip request. See 9.2.1.
	MonitoringParameter	0:1	*+TripMonitoringParam*	Other parameters for configuring trip monitoring. See 20.2.1.

Table 232: Description of structure TripMonitoringSubscriptionRequestStructure

20.4 Response structures

20.4.1 TripMonitoringDeliveryStructure

TripMonitoringDeliveryStructure			*+Structure (derived from siri:AbtractServiceDeliveryStructure)*	**Provides information about a monitored trip.**
MonitoringAlertReason	*Situations*	0:1	*Situations*	(Fault) events as reason for notification (see 7.8.1).
	FacilityCondition	0:*	*+siri:FacilityCondition*	One or more statuses of facilities as reason for notification, see 7.7.
	ConnectionStatus	0:*	*+ConnectionStatus*	The status of a connection in the monitored trip. See 13.3.4.
AlternativeTrip	*TripResponse*	0:1	*+TripResponse*	Contains a trip alternative (see 9.3.1).
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. See 7.4.2.

Table 233: Description of structure TripMonitoringDeliveryStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
ALERT_TRIPREQUEST_ORIGIN_UNKNOWN	The departure location (address, stop, etc.) of TripRequest is unknown.
ALERT_TRIPREQUEST_DESTINATION_UNKNOWN	The arrival location (address, stop, etc.) of TripRequest is unknown.
ALERT_TRIP_UNKNOWN	The trip to be monitored is unknown.
ALERT _THRESHOLD_NEGATIVE	Delay change, after which notification is sent again, has a negative value.
ALERT_FACILITY_UNKNOWN	The facility to be monitored is unknown.
ALERT_MONITORED_OBJECT_UNKNOWN	The object about which events and faults should be reported is unknown.

Table 234: List of error states in TripMonitoringDeliveryStructure

21 Personalisierungsdienst

21.1 Beschreibung

Dieser Dienst stellt Funktionen bereit, über die Daten für die personalisierte Konfiguration beliebiger Dienste hinterlegt werden können. Unter „Konfiguration“ sind hier nicht nur explizite Einstellungen im engeren Sinn zu verstehen, sondern allgemein benutzerbezogene Daten. All diese Informationen können das Verhalten derjenigen Dienste, die den Personalisierungsdienst benutzen, beeinflussen und stellen somit eine Konfiguration für den verwendenden Dienst dar.

Es ist wichtig, zu beachten, dass der Dienst keine eigenen personalisierten Funktionen zur Verfügung stellt. Seine Aufgabe besteht in der Verwaltung von Benutzereinstellungen. Andere Dienste können auf den Personalisierungsdienst zurückgreifen, um ihre Funktionen personalisiert zur Verfügung zu stellen. Der Zugriff auf die personalisierten Daten erfolgt mit Hilfe des Authentifizierungsdienstes, um einen Schutz der Daten, entsprechend der rechtlichen Vorgaben zum Datenschutz, zu ermöglichen. Diese Vorgaben sind bei der Umsetzung des Personalisierungsdienstes zu berücksichtigen, werden in der Dienstbeschreibung des Personalisierungsdienstes jedoch nicht behandelt. Generell gilt, dass Benutzer nur auf von ihnen gespeicherte Daten zugreifen können. Daten von anderen Benutzern bleiben stets vollkommen unsichtbar und unerreichbar. Wie diese Trennung der benutzerbezogenen Daten erreicht wird, ist herstellerabhängig und wird hier nicht vorgeschrieben.

Der Dienst speichert beliebige Datenwerte, wobei jedem Datenwert ein – für den aktuellen Benutzer – eindeutiger Schlüssel zugeordnet wird. Über diesen Schlüssel kann der Datenwert wieder abgerufen werden. Bei den Werten handelt es sich um Zeichenketten mit beliebigem Format, sodass prinzipiell jegliche Datenstrukturen abgelegt werden können.

Der Dienst bietet keine Zuordnung von Werten zu einem oder mehreren anderen Diensten an. Diese kann über herstellerspezifische Schnittstellen unterstützt werden.

In der XML-Schema-Definition *Trias_Personalisation.xsd* werden Datentypen und Strukturen definiert, die für den Personalisierungsdienst verwendet werden.

21.2 Speicherung von Verbindungen

Der Personalisierungsdienst kann neben allgemeinen Inhalten auch Verbindungen speichern. Dafür wird die Struktur *TripStructure* (vgl. 9.3.4) verwendet. Als Schlüssel wird entsprechend die *TripId* der Verbindung verwendet. Dieser Teil des Dienstes funktioniert genauso, wie in den restlichen Unterkapiteln beschrieben.

21.3 Interaktionen

Die Funktionen dieses Dienstes stehen oft für sich alleine und sind sehr generisch. Daher werden im Folgenden drei beispielhafte Abläufe im Zusammenhang mit dem Dienst beschrieben, bei denen die Funktionsaufrufe in einen größeren Zusammenhang eingebettet sind.

Auf den Personalisierungsdienst kann von verschiedenen Komponenten aus zugegriffen werden, zum Beispiel von der EKAP oder von Mehrwertdiensten aus. Bei den im Folgenden gezeigten Zugriffen auf den Personalisierungsdienst handelt es sich um beispielhafte Abläufe. Dabei sollen die anderen beteiligten Komponenten nicht konkret festgelegt werden. Aus diesem Grund wird in den untenstehenden Diagrammen von beliebigen Mehrwertdiensten - welche in beliebigen Ausprägungen vorkommen können und im Rahmen dieser Schrift ebenfalls nicht näher spezifiziert werden - gesprochen. Zugriffe von anderen Komponenten aus finden nach demselben Muster statt.

21.3.1 Lebenszyklus eines Wertes

Das erste Beispiel zeigt auf, wie ein Wert über den Personalisierungsdienst abgelegt und wieder gelöscht werden kann.

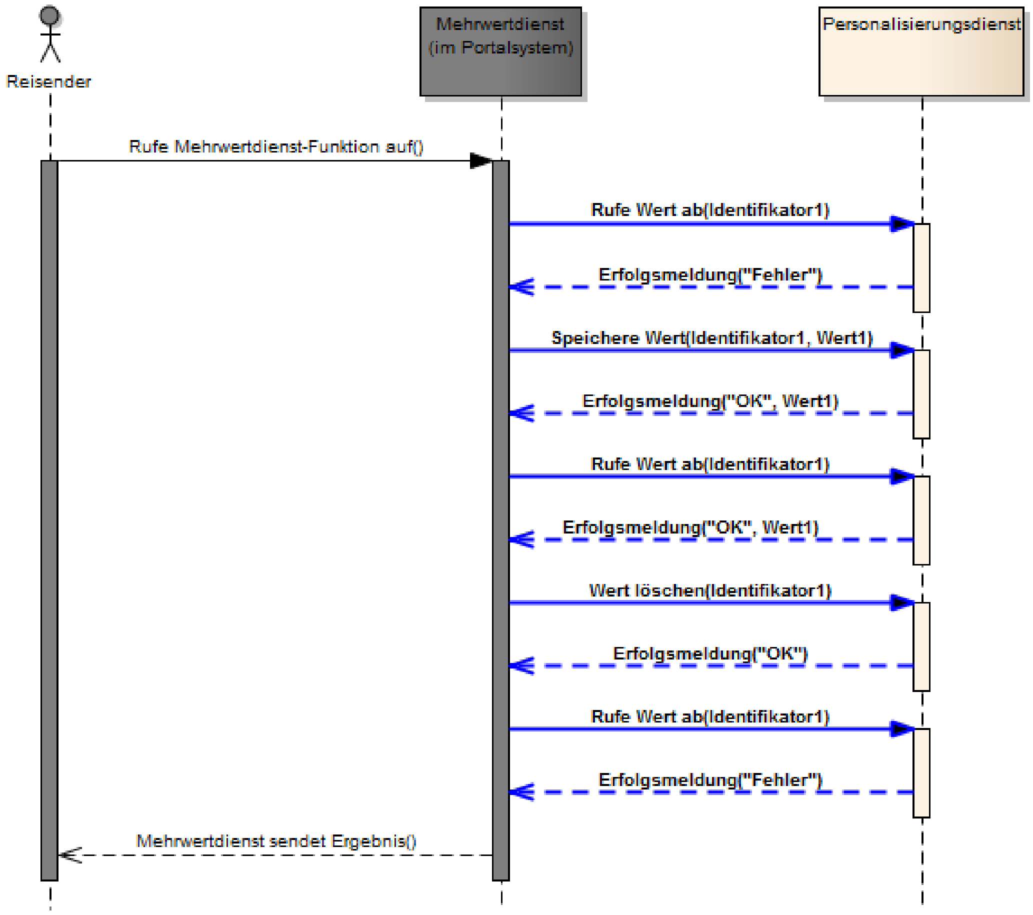
Reisender
Mehrwertdienst (im Portalsystem)
Personalisierungsdienst
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("Fehler")
Speichere Wert(Identifikator1, Wert1)
Erfolgsmeldung("OK", Wert1)
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("OK", Wert1)
Wert löschen(Identifikator1)
Erfolgsmeldung("OK")
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("Fehler")
Mehrwertdienst sendet Ergebnis()

Abbildung 7: Sequenzdiagramm Lebenszyklus eines Wertes

21.3.2 Werteliste ermitteln

In diesem Beispiel ist dargestellt, wie die Funktion zum Auflisten der verfügbaren Werte funktioniert.

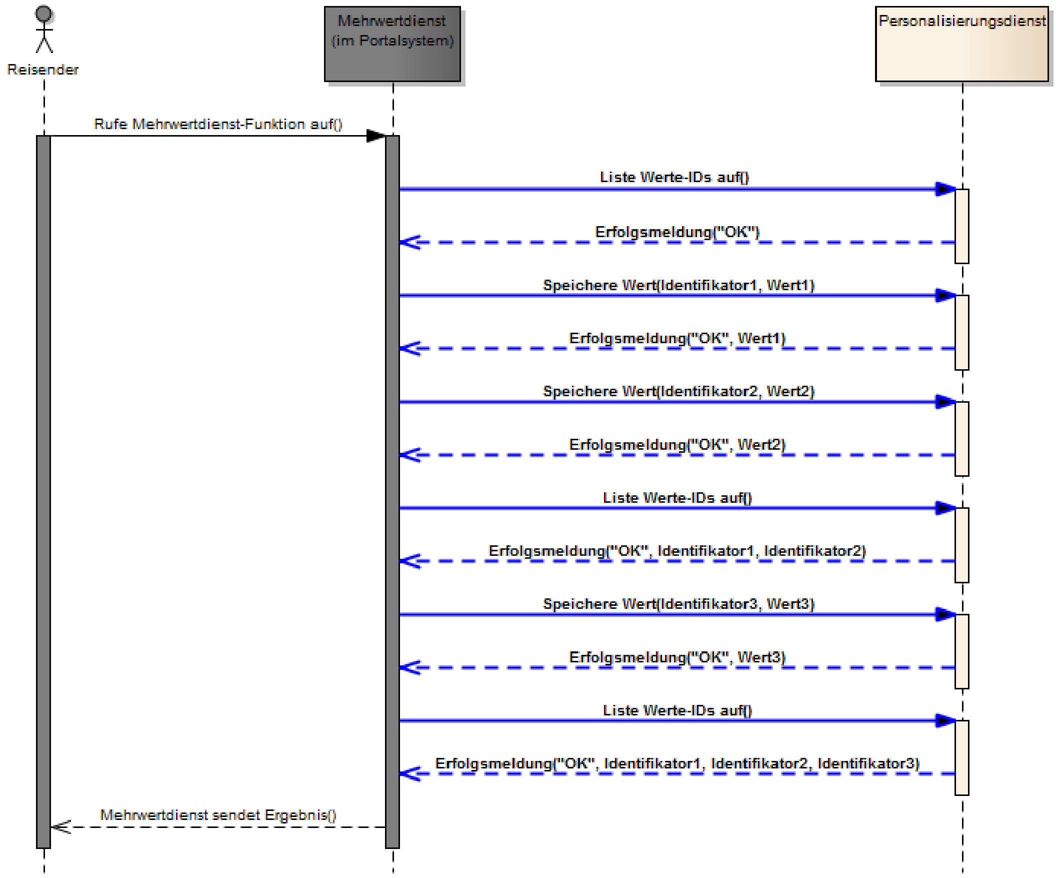
Reisender
Mehrwertdienst (im Portalsystem)
Personalisierungsdienst
Rufe Mehrwertdienst-Funktion auf()
Liste Werte-IDs auf()
Erfolgsmeldung("OK")
Speichere Wert(Identifikator1, Wert1)
Erfolgsmeldung("OK", Wert1)
Speichere Wert(Identifikator2, Wert2)
Erfolgsmeldung("OK", Wert2)
Liste Werte-IDs auf()
Erfolgsmeldung("OK", Identifikator1, Identifikator2)
Speichere Wert(Identifikator3, Wert3)
Erfolgsmeldung("OK", Wert3)
Liste Werte-IDs auf()
Erfolgsmeldung("OK", Identifikator1, Identifikator2, Identifikator3)
Mehrwertdienst sendet Ergebnis()

Abbildung 8: Sequenzdiagramm Werteliste ermitteln

21.3.3 Werte speichern und abrufen

Im folgenden Beispiel wird gezeigt, wie ein Reisender auf unterschiedliche Mehrwertdienste zugreift, welche den Personalisierungsdienst nutzen, um Konfigurationseinstellungen abzulegen und wieder abzurufen.

Dabei wird illustriert, dass zu jedem gespeicherten Wert ein für den Benutzer eindeutiger Identifikator gehört. Erstmaliges Speichern eines Wertes für einen Identifikator legt einen Wert im Speicher an, erneutes Speichern unter demselben Identifikator überschreibt den ursprünglichen Wert.

Des Weiteren wird im Beispiel gezeigt, dass die Werte vom Personalisierungsdienst prinzipiell dauerhaft gespeichert werden, auch wenn beispielsweise inzwischen ein anderer Mehrwertdienst genutzt wird. Hierbei ist zu beachten, dass Anbieter gespeicherte Werte mit einem Löschdatum versehen können, um ungenutzte Daten nicht unbegrenzt vorhalten zu müssen. Die genaue Umsetzung entsprechender Löschungen ist anbieterspezifisch und wird in dieser Schrift nicht festgelegt.

Um klarzustellen, dass in der standardisierten Form keine Zuordnung zwischen Werten und Diensten stattfindet, wird ferner vorgeführt, dass ein Mehrwertdienst (MWD 2 in der Abbildung) einen ursprünglich von einem anderen Mehrwertdienst (MWD 1 in der Abbildung) abgelegten Wert überschreiben kann. Die Voraussetzung dazu ist lediglich, dass MWD 2 den Identifikator des

Werts kennt, sei es, weil MWD 1 und 2 vom selben Hersteller stammen, oder weil der Entwickler von MWD 1 die in MWD 1 verwendeten Identifikatoren öffentlich bekanntgegeben hat.

Der Dienst kann bei Bedarf durch herstellerspezifische Schnittstellen um die Funktionalität, den Zugriff auf bestimmte Werte dienstspezifisch einzuschränken, erweitert werden und somit den Zugriff ausgewählter Dienste auf einzelne Werte einschränken. Hierzu kann, je nach Implementierung und Konfiguration, auch der Authentifizierungsdienst genutzt werden.

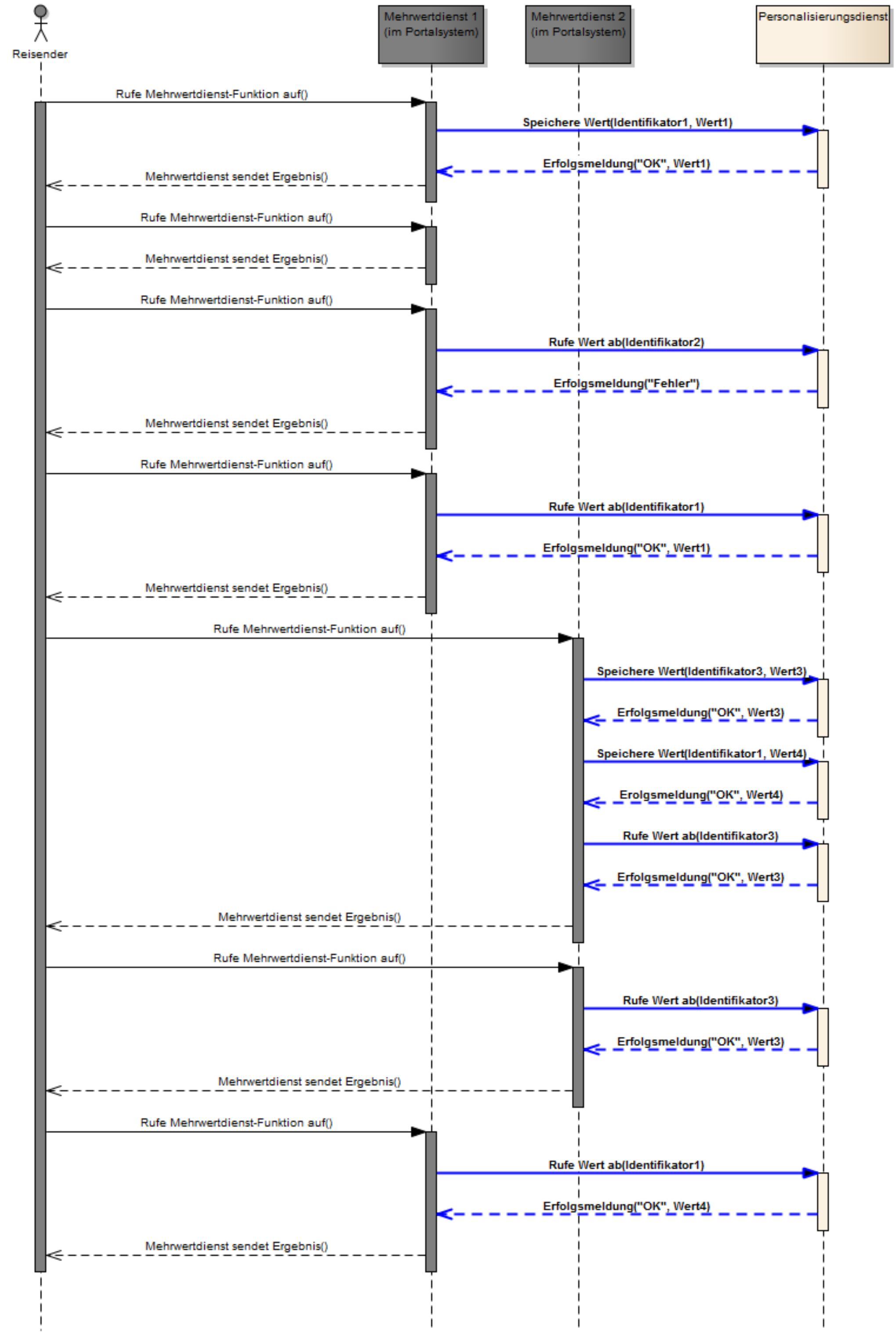
Reisender
Mehrwertdienst 1 (im Portalsystem)
Mehrwertdienst 2 (im Portalsystem)
Personalisierungsdienst
Rufe Mehrwertdienst-Funktion auf()
Speichere Wert(Identifikator1, Wert1)
Erfolgsmeldung("OK", Wert1)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator2)
Erfolgsmeldung("Fehler")
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("OK", Wert1)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Speichere Wert(Identifikator3, Wert3)
Erfolgsmeldung("OK", Wert3)
Speichere Wert(Identifikator1, Wert4)
Erolgsmeldung("OK", Wert4)
Rufe Wert ab(Identifikator3)
Erfolgsmeldung("OK", Wert3)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator3)
Erfolgsmeldung("OK", Wert3)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("OK", Wert4)
Mehrwertdienst sendet Ergebnis()

Abbildung 9: Sequenzdiagramm Personalisierung

21 Personalisation service

21.1 Description

This service provides functions, by means of which data can be saved for personalised configuration of any services. Here, “configuration” means not only explicit setting in a narrow sense but also general user-related data. All this information can influence the behaviour of those services that use personalisation service, and represent a configuration for the used service.

It is important is ensure that the service does not provide its own personalised functions. Its task is management of user settings. Other services can fall back on the personalisation service in order to provide its functions in a personalised way. Personalised data is accessed with the help of authentication service in order to enable protection of data in accordance with the legal regulations on data protection. These regulations must be taken into account when implementing the personalisation service but must not be used in describing the personalisation service. Usually users can only access data saved by them. Data of other users always remains completely invisible and inaccessible. How this partition of user-related data is achieved is manufacturer-specific and is not stipulated here.

The service saves arbitrary data values, whereby a unique key for the current user is assigned to every data value. The data value can be accessed again using this key. These values are strings with an arbitrary format so that in principle any data structures can be saved.

The service does not offer assignment of values to one or more other services. It can be supported via manufacturer-specific interfaces.

Data types and structures which are used for the personalisation service are defined in the XML schema definition *Trias_Personalisation.xsd*.

21.2 Storing trips

Additional to general content, the Personalisation service is able to save trips. For that purpose the structure *TripStructure* (vgl. 9.3.4) is used. Correspondingly the *TripId* is used as a key for a trip. This part of the service works exactly as the general part.

21.3 Interactions

The functions of this service are often independent and very generic. Hence, three exemplary sequences are described below with regard to the service, in which the function calls are embedded in a broader context.

Different components can access the personalisation service, for example, EKAP or value-added services. The exemplary sequences are accesses to personalisation service shown below. In the process, the other components involved should not be concretely defined. For this reason, value-added services are discussed in the diagrams below - which can occur in any instances and within the scope of this document if not specified in detail. Access from other components takes place as per the same pattern.

21.3.1 Lifecycle of a value

The first example shows how a value can be saved and deleted using the personalisation service.

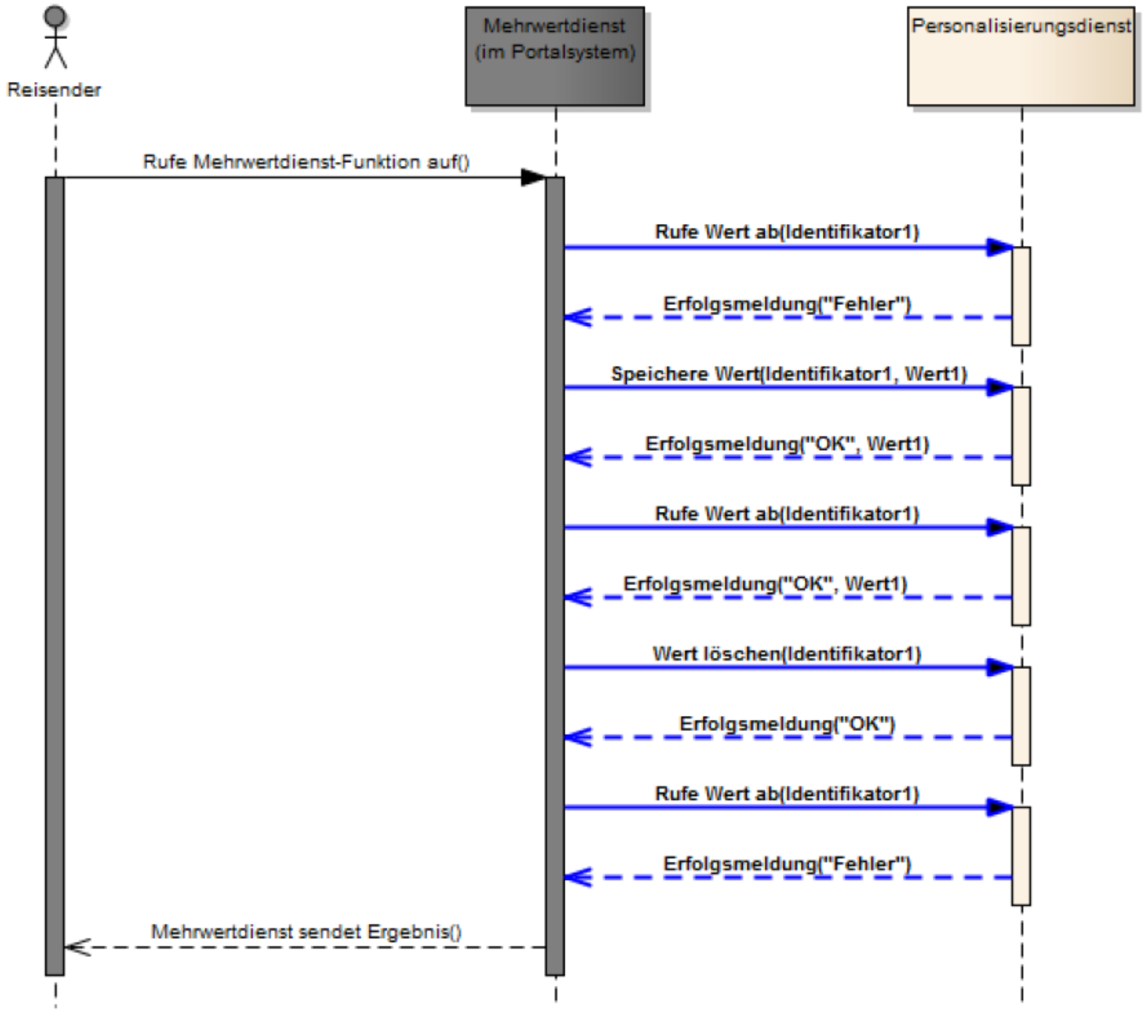
Reisender
Mehrwertdienst (im Portalsystem)
Personalisierungsdienst
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("Fehler")
Speichere Wert(Identifikator1, Wert1)
Erfolgsmeldung("OK", Wert1)
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("OK", Wert1)
Wert löschen(Identifikator1)
Erfolgsmeldung("OK")
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("Fehler")
Mehrwertdienst sendet Ergebnis()

Figure 7: Sequence diagram of the lifecycle of a value

21.3.2 Determining list of values

This example shows how the function to list available values works.

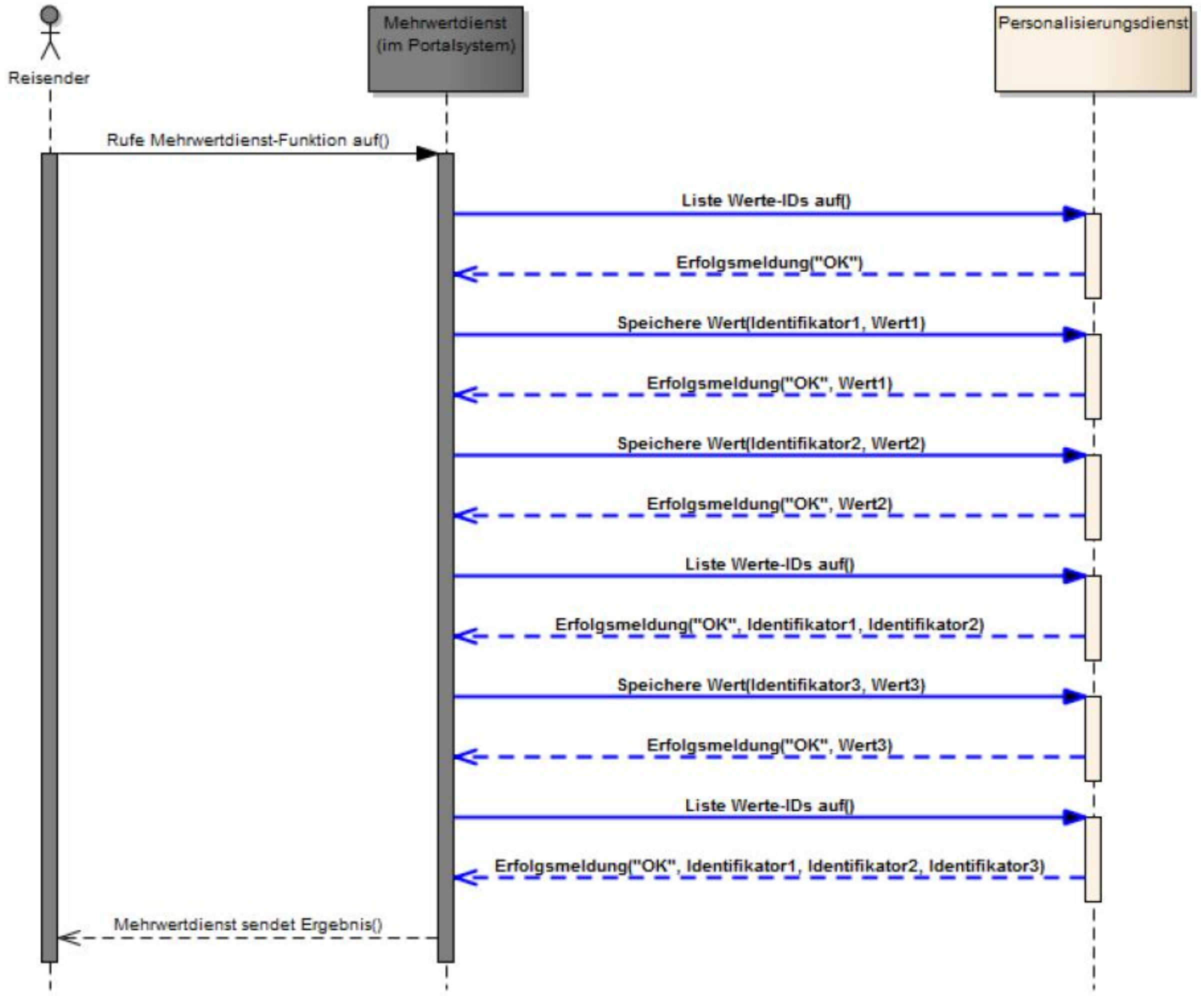
Reisender
Mehrwertdienst (im Portalsystem)
Personalisierungsdienst
Rufe Mehrwertdienst-Funktion auf()
Liste Werte-IDs auf()
Erfolgsmeldung("OK")
Speichere Wert(Identifikator1, Wert1)
Erfolgsmeldung("OK", Wert1)
Speichere Wert(Identifikator2, Wert2)
Erfolgsmeldung("OK", Wert2)
Liste Werte-IDs auf()
Erfolgsmeldung("OK", Identifikator1, Identifikator2)
Speichere Wert(Identifikator3, Wert3)
Erfolgsmeldung("OK", Wert3)
Liste Werte-IDs auf()
Erfolgsmeldung("OK", Identifikator1, Identifikator2, Identifikator3)
Mehrwertdienst sendet Ergebnis()

Figure 8: Sequence diagram for determining list of values

21.3.4 Saving and calling values

The following example shows how a passenger access different value-added services which use the personalisation service, in order to save and re-access the configuration settings.

Here, it is also illustrated that an identifier unique for the user belongs to each value saved. First-time saving of a value for an identifier creates a value in the memory. Next saving under the same identifier overwrites the original value.

Moreover, the example shows that the values from the personalisation service are permanently saved, even if, for example, another value-added service is used in the meanwhile. Here, it must be kept in mind that the providers can provide saved values with a deletion date so that the data need not be saved indefinitely. The exact implementation of the corresponding deletions is provider-specific and is not defined in this document.

In order to clarify that an assignment between values and services is not carried out in the standardised form, it is shown that a value-added service (MWD 2 in the figure) can overwrite an original value saved by another value-added service (MWD 1 in the figure). The only prerequisite for this is that MWD 2 recognises the identifier of the value unless MWD 1 and 2 originate from the same manufacturer or because the developer of MWD 1 has publicly disclosed the identifiers used in MWD 1.

If necessary, the service can be extended by the function to limit access to certain values depending on the service via manufacturer-specific interfaces and thus limit access of selected services to individual values. For this purpose, even the authentication service can be used depending on the implementation and configuration.

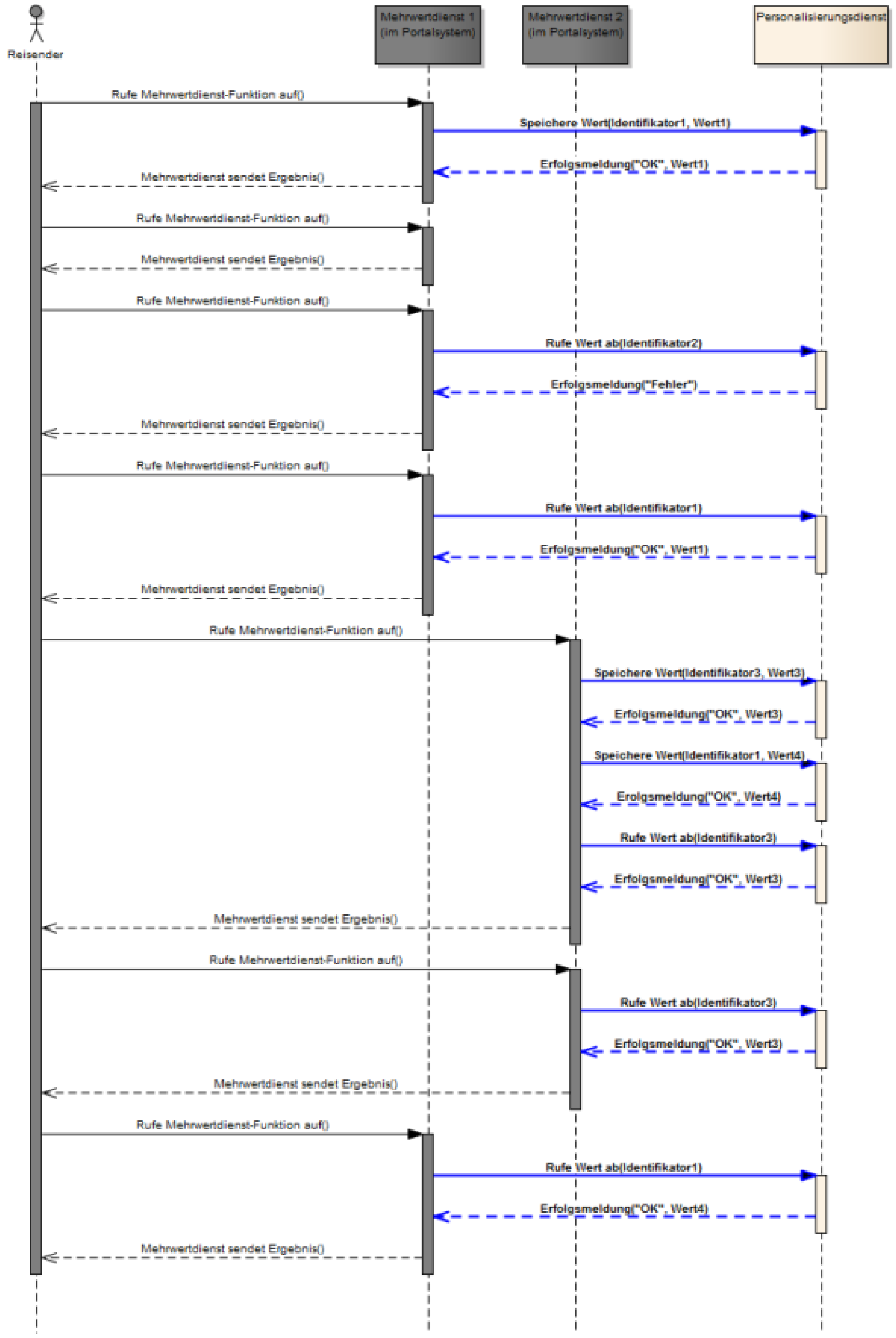
Reisender
Mehrwertdienst 1 (im Portalsystem)
Mehrwertdienst 2 (im Portalsystem)
Personalisierungsdienst
Rufe Mehrwertdienst-Funktion auf()
Speichere Wert(Identifikator1, Wert1)
Erfolgsmeldung("OK", Wert1)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator2)
Erfolgsmeldung("Fehler")
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("OK", Wert1)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Speichere Wert(Identifikator3, Wert3)
Erfolgsmeldung("OK", Wert3)
Speichere Wert(Identifikator1, Wert4)
Erolgsmeldung("OK", Wert4)
Rufe Wert ab(Identifikator3)
Erfolgsmeldung("OK", Wert3)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator3)
Erfolgsmeldung("OK", Wert3)
Mehrwertdienst sendet Ergebnis()
Rufe Mehrwertdienst-Funktion auf()
Rufe Wert ab(Identifikator1)
Erfolgsmeldung("OK", Wert4)
Mehrwertdienst sendet Ergebnis()

Figure 9: Sequence diagram for personalisation

21.4 Simple types

The following simple types are defined:

Type name	*Basic type*	Description
ValueIdType	*xs:string*	Identifier of a value.

Table 235: List of simple type definitions in Trias_Personalisation.xsd

21.5 Request structures

21.5.1 PersonalisationRequestStructure

PersonalisationRequest				*+Structure*	**Represents a request to the personalisation service.**
	a	***SaveValue***	**–1:n**	*+PersonalisationSaveValueRequest*	If this is available, personalised values should be saved by the request. Each value identifier is allowed only once. Detailed information about this is included in this structure (see 21.5.2).
	b	***RetrieveValue***		*+PersonalisationRetrieveValueRequest*	If this is available, saved personalised values should be retrieved by the request. Each value identifier is allowed only once. Detailed information about this is included in this structure (see 21.5.3).
	c	***DeleteValue***		*+PersonalisationDeleteValueRequest*	If this is available, saved personalised values should be deleted by the request.Each value identifier is allowed only once. Detailed information about this is included in this structure (see 21.5.4).
	d	***EnumerateValues***		*+PersonalisationEnumerateValuesRequest*	If this is available, the saved personalised values available should be retrieved by the request. Detailed information about this is included in this structure (see 21.5.5).
	e	***SaveTrip***		*+PersonalisationSaveTripRequest*	If this is available, trips should be saved by the request. Each trip identifier is allowed only once. Detailed information about this is included in this structure (see 21.5.6).
	f	***RetrieveTrip***		*+PersonalisationRetrieveTripRequest*	If this is available, saved trips should be retrieved by the request. Each trip identifier is allowed only once. Detailed information about this is included in this structure (see 21.5.7).
	g	***DeleteTrip***		*+PersonalisationDeleteTripRequest*	If this is available, saved trips should be deleted by the request. Each trip identifier is allowed only once. Detailed information about this is included in this structure (see 21.5.8).
	h	***EnumerateTrips***		*+PersonalisationEnumerateTripsRequest*	If this is available, the saved trips available should be retrieved by the request. Detailed information about this is included in this structure (see 21.5.9).

Table 236: Description of structure PersonalisationRequestStructure

21.5.2 PersonalisationSaveValueRequestStructure

PersonalisationSaveValueRequest			*+Structure*	**Contains detailed information for saving personalised values.**
	ValueId	**1:1**	→*ValueId*	The identifier of the value to be saved. Already stored values with identical identifier will be overwritten. See 21.4.
	Value	**1:1**	*xs:string*	The value to be saved.

Table 237: Description of structure PersonalisationSaveValueRequestStructure

21.5.3 PersonalisationRetrieveValueRequestStructure

PersonalisationRetrieveValueRequest			*+Structure*	**Contains detailed information for retrieving saved personalised values.**
	ValueId	**1:1**	→*ValueId*	The identifier of the value to be retrieved. See 21.4.

Table 238: Description of structure PersonalisationRetrieveValueRequestStructure

21.5.4 PersonalisationDeleteValueRequestStructure

PersonalisationDeleteValueRequest			*+Structure*	**Contains detailed information for deleting saved personalised values.**
	ValueId	**1:1**	→*ValueId*	The identifier of the value to be deleted. See 21.4.

Table 239: Description of structure PersonalisationDeleteValueRequestStructure

21.5.5 PersonalisationEnumerateValuesRequestStructure

PersonalisationEnumerateValuesRequest	*+Structure*	**Expresses that a list of the values saved should be retrieved.**

Table 240: Description of structure PersonalisationEnumerateValuesRequestStructure

21.5.6 PersonalisationSaveTripRequestStructure

PersonalisationSaveTripRequest			*+Structure*	**Contains detailed information for saving trips.**
	Trip	**1:1**	*+Trip*	The trip to be saved. Including its identifier *TripId*. Already stored trips with identical identifier will be overwritten. See 9.3.4.

Table 241: Description of structure PersonalisationSaveTripRequestStructure

21.5.7 PersonalisationRetrieveTripRequestStructure

PersonalisationRetrieveTripRequest			*+Structure*	**Contains detailed information for retrieving saved trips.**
	TripId	**1:1**	*xs:NMTOKEN*	The identifier of the trip to be retrieved.

Table 242: Description of structure PersonalisationRetrieveTripRequestStructure

21.5.8 PersonalisationDeleteTripRequestStructure

PersonalisationDeleteTripRequest			*+Structure*	**Contains detailed information for deleting saved trips.**
	TripId	**1:1**	*xs:NMTOKEN*	The identifier of the trip to be deleted.

Table 243: Description of structure PersonalisationDeleteTripRequestStructure

21.5.9 PersonalisationEnumerateTripsRequestStructure

PersonalisationEnumerateTripsRequest	*+Structure*	**Expresses that a list of the saved trips should be retrieved.**

Table 244: Description of structure PersonalisationEnumerateTripsRequestStructure

21.6 Response structures

21.6.1 PersonalisationResponseStructure

PersonalisationResponse				*+Structure*	**Represents the response to a request to the personalisation service.**
	ErrorMessage		0:*	*+ErrorMessage*	Contains possible error messages which are about the general processing of the message by a personalisation service.
	a	***SaveValue***	**–1:n**	*+PersonalisationSaveValueResponse*	If this is available, the response expresses whether saving personalised values was successful. Detailed information about this is included in this structure (see 21.6.2).
	b	***RetrieveValue***		*+PersonalisationRetrieveValueResponse*	If this is available, the response expresses whether retrieving personalised values was successful. Detailed information about this is included in this structure (see 21.6.3).
	c	***DeleteValue***		*+PersonalisationDeleteValueResponse*	If this is available, the response expresses whether deleting personalised values was successful. Detailed information about this is included in this structure (see 21.6.4).
	d	***EnumerateValues***		*+PersonalisationEnumerateValuesResponse*	If this is available, the response expresses whether listing all the personalised values available was successful. Detailed information about this is included in this structure (see 21.6.5).
	e	***SaveTrip***		*+PersonalisationSaveTripResponse*	If this is available, the response expresses whether saving trips was successful. Detailed information about this is included in this structure (see 21.6.6).
	f	***RetrieveTrip***		*+PersonalisationRetrieveTripResponse*	If this is available, the response expresses whether retrieving trips was successful. Detailed information about this is included in this structure (see 21.6.7).
	g	***DeleteTrip***		*+PersonalisationDeleteTripResponse*	If this is available, the response expresses whether deleting trips was successful. Detailed information about this is included in this structure (see 21.6.8).
	h	***EnumerateTrips***		*+PersonalisationEnumerateTripsResponse*	If this is available, the response expresses whether listing all the trips available was successful. Detailed information about this is included in this structure (see 21.6.9).

Table 245: Description of structure PersonalisationResponseStructure

21.6.2 PersonalisationSaveValueResponseStructure

PersonalisationSaveValueResponse			*+Structure*	**Contains detailed information about the completed saving of a personalised value.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the saving process was successful.
	ValueId	**1:1**	→*ValueId*	The identifier of the value just saved. See 21.4.

Table 246: Description of structure PersonalisationSaveValueResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
PERSONALISATIONSAVEVALUEREQU-EST_INVALID_ID	The specified identifier has an invalid format.

Table 247: List of error states in PersonalisationSaveValueResponseStructure

21.6.3 PersonalisationRetrieveValueResponseStructure

PersonalisationRetrieveValueResponse			*+Structure*	**Contains detailed information about the completed retrieving of a personalised value.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the retrieving process was successful.
	ValueId	**1:1**	→ValueId	The identifier of the value retrieved. See 21.4.
	Value	0:1	*xs:string*	If the retrieving process was successful, the retrieved value.

Table 248: Description of structure PersonalisationRetrieveValueResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
PERSONALISATIONRETRIEVEVALUERE-QUEST_UNKNOWN_ID	A value is not saved for the specified identifier in the personalisation service.

Table 249: List of error states in PersonalisationRetrieveValueResponseStructure

21.6.4 PersonalisationDeleteValueResponseStructure

PersonalisationDeleteValueResponse			*+Structure*	**Contains detailed information about the completed deletion of a personalised value.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the deletion process was successful.
	ValueId	**1:1**	→ValueId	The identifier of the value retrieved. See 21.4.

Table 250: Description of structure PersonalisationDeleteValueResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
PERSONALISATIONDELETEVALUEREQUEST_UNKNOWN_ID	A value is not saved for the specified identifier in the personalisation service.

Table 251: List of error states in PersonalisationDeleteValueResponseStructure

21.6.5 PersonalisationEnumerateValuesResponseStructure

PersonalisationEnumerateValuesResponse			*+Structure*	**Contains detailed information about the listing of all the personalised values saved.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the retrieving process was successful.
	ValueId	0:*	→*ValueId*	If the retrieving process was successful, an element, which contains an identifier of a saved value, is available for every saved value. See 21.4.

Table 252: Description of structure PersonalisationEnumerateValuesResponseStructure

21.6.6 PersonalisationSaveTripResponseStructure

PersonalisationSaveTripResponse			*+Structure*	**Contains detailed information about the completed saving of a trip.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the saving process was successful.
	TripId	**1:1**	*xs:NMTOKEN*	The identifier of the trip just saved.

Table 253: Description of structure PersonalisationSaveTripResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
PERSONALISATIONSAVETRIPREQUEST_INVALID_ID	The specified identifier has an invalid format.

Table 254: List of error states in PersonalisationSaveTripResponseStructure

21.6.7 PersonalisationRetrieveTripResponseStructure

PersonalisationRetrieveTripResponse			*+Structure*	**Contains detailed information about the completed retrieving of a trip.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the retrieving process was successful.
	TripId	**1:1**	*xs:NMTOKEN*	The identifier of the value retrieved.
	Trip	0:1	*+Trip*	If the retrieving process was successful, the retrieved trip. See 9.3.4.

Table 255: Description of structure PersonalisationRetrieveTripResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
PERSONALISATIONRETRIEVETRIPREQUEST_UNKNOWN_ID	A trip is not saved for the specified identifier in the personalisation service.

Table 256: List of error states in PersonalisationRetrieveTripResponseStructure

21.6.8 PersonalisationDeleteTripResponseStructure

PersonalisationDeleteTripResponse			*+Structure*	**Contains detailed information about the completed deletion of a trip.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the deletion process was successful.
	TripId	**1:1**	*xs:NMTOKEN*	The identifier of the value retrieved.

Table 257: Description of structure PersonalisationDeleteTripResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
PERSONALISATIONDELETETRIPREQUEST_UNKNOWN_ID	A trip is not saved for the specified identifier in the personalisation service.

Table 258: List of error states in PersonalisationDeleteTripResponseStructure

21.6.9 PersonalisationEnumerateTripsResponseStructure

PersonalisationEnumerateTripsResponse			*+Structure*	**Contains detailed information about the listing of all the saved trips.**
	ErrorMessage	0:*	*+ErrorMessage*	States whether the retrieving process was successful.
	TripId	0:*	*xs:NMTOKEN*	If the retrieving process was successful, an element, which contains an identifier of a saved trip, is available for every saved trip.

Table 259: Description of structure PersonalisationEnumerateTripsResponseStructure

22 Dienst Fahrzeuginformationen

22.1 Beschreibung

Dieser Dienst dient dazu, dass zwischen einem Fahrzeug und einer mobilen Applikation, die von einem Fahrgast benutzt wird, Informationen, die das Fahrzeug betreffen, ausgetauscht werden können.

In der XML-Schema-Definition *Trias_VehicleInterface.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Fahrzeuginformationen verwendet werden.

22 Vehicle information service

22.1 Description

This service helps in exchanging information related to the vehicle between a vehicle and a mobile application which is used by a passenger.

Data types and structures are defined in the XML schema definition Trias_VehicleInterface.xsd which are used for the vehicle information service.

22.2 Request structures

22.2.1 VehicleDataRequestStructure

VehicleDataRequestStructure			*+Structure*	**Summarises the request data for a request of vehicle data.**
	VehicleStatus	0:1	*xs:boolean*	Vehicle-status information should be sent by the vehicle. Default is *true*.
	VehicleActivity	0:1	*xs:boolean*	Vehicle-activity information should be sent by the vehicle. Default is *false*.

Table 260: Description of structure VehicleDataRequestStructure

22.3 Response structures

22.3.1 VehicleDataResponseStructure

VehicleDataResponseStructure			*+Structure*	**Summarises the result data for a request of vehicle data.**
	VehicleCode	**1:1**	*→VehicleCode*	Unique vehicle ID, with which referencing to EKAP data should be enabled; see 7.4.1.
	VehicleStatus	0:1	*+VehicleStatus*	Information related to the status of the vehicle; see 22.3.2.
	VehicleActivity	0:1	*+VehicleActivity*	Information related to the activity of the vehicle; see 22.3.3.

Table 261: Description of structure VehicleDataResponseStructure

22.3.2 VehicleStatusStructure

VehicleStatusStructure			*+Structure*	**Information related to the status of the individual vehicle.**
	DoorState	0:1	*DoorsOpen | All-DoorsClosed*	Information about the door state.
	VehicleStopRequested	0:1	*xs:boolean*	Information, whether a stop request for the upcoming stop is already known to the vehicle system. Default is *false*.
	InPanic	0:1	*xs:boolean*	Information whether a safety alarm has been triggered. Default is *false*.
	VehicleTypeRef	0:1	→*VehicleType*	Information about the vehicle type and hence the vehicle facility, see 7.4.1.
Service Facility	:::	0:1	*+siri:ServiceFacilityGroup*	Classification of facility properties. See 7.7.3.

Table 262: Description of structure VehicleStatusStructure

22.3.3 VehicleActivityStructure

VehicleActivityStructure			*+Structure*	**Information related to the activity of the individual vehicle.**
	TimetableDelay	0:1	*xs:int*	Deviation from the timetable in seconds, early arrivals are shown as negative values.
	RouteDeviation	0:1	*onroute | offroute | unknown*	Information, whether the vehicle is on the planned line route or not.
	JourneyMode	0:1	*NoTrip | AdditionalTrip | ServiceTrip*	Information about the type of journey which the vehicle has undertaken (planned journey, repeater journey etc.).
Service Pattern Position	*StopSequence*	0:*	*+StopSequence*	Information about the sequence of stopping points which is necessary to represent the stops in the form of a string of pearls; see 22.3.4.
	CurrentStopIndex	0:1	*xs:int*	Index about the stopping point reached next in the stopping point sequence.
	LocationState	0:1	*AfterStop | AtStop | BeforeStop | BetweenStop*	Information whether the vehicle is still at the stop, is just before it or passed it or is between two stops.
	NextExitSide	0:1	*both |left |right | unknown*	Information about the exit side at the next stopping point.

Table 263: Description of structure VehicleActivityStructure

22.3.4 StopSequenceStructure

StopSequenceStructure			*+Structure*	**Information about stopping point sequence.**
	StopPoint	**2:***	*+StopInformation*	Information about the stopping point, see 22.3.5.

Table 264: Description of structure StopSequenceStructure

22.3.5 StopInformationStructure

StopInformationStructure			*+Structure*	**Information about the individual stopping point.**
	StopIndex	**1:1**	*xs:int*	Index of the current stopping point in the stopping point sequence.
	StopRef	0:1	*→StopPoint*	Reference to the stopping point, see 7.5.1.
	StopName	**1:***	*+InternationalText*	Name of the stopping point.
	StopAlternativeName	0:*	*+InternationalText*	Alternative name of the stopping point.
	Platform	0:*	*xs:string*	Name of stop platform.
	DisplayContent	0:*	*+DisplayContent*	Information about the composition of the target text display content; see 22.3.6.
	ArrivalScheduled	0:1	*xs:dateTime*	Display of scheduled arrival time.
	DepartureScheduled	0:1	*xs:dateTime*	Display of scheduled departure time.
	RecordedArrivalTime	0:1	*xs:dateTime*	Information about the actual arrival time (is necessary for the field test in Stuttgart and during the migration period).
	DistanceToNextStop	0:1	*xs:double*	Distance from the next stop in [m].
	AnnouncementNext Stop	0:*	*+Announcement*	Information about the stop announcement, see 22.3.10.
	Farezone	0:*	*xs:NMTOKEN*	Information about the tariff zones, in which this stopping point lies.
	Connection	0:*	*+Connection*	Information about connections, see 22.3.11.

Table 265: Description of structure StopInformationStructure

22.3.6 DisplayContentStructure

DisplayContentStructure			*+Structure*	**Information about the composition of the individual target text display content.**
	Line	0:1	*+LineInformati on*	Information about the name of the line; see 22.3.7.
	Destination	**1:1**	*+Destination*	Information about the content of the target text; see 22.3.8.
	Via	0:*	*+ViaPoint*	Information about intermediate stops, see 22.3.9.
	AdditionalInformation	0:*	*+International Text*	Additional information such as “fast-track”, “relief bus” etc.
Display Policy	***PeriodDuration***	**1:1**	*xs:duration*	Information about the period duration when changing between different display contents.
	Duration	**1:1**	*xs:duration*	Information about the display duration of a display content within a display period (when changing between different display contents).

Table 266: Description of structure DisplayContentStructure

22.3.7 LineInformationStructure

LineInformationStructure			*+Structure*	**Information about the name of the line.**
	LineRef	**1:1**	→*Line*	Reference to a line; see 7.4.1.
	LineName	**1:***	*+International Text*	Passenger-relevant name of the line
	LineShortName	0:*	*+International Text*	Short name of the line.
	LineNumber	**1:1**	*xs:int*	Number of the line.

Table 267: Description of structure LineInformationStructure

22.3.8 DestinationStructure

DestinationStructure			*+Structure*	**Information about the content of the target text.**
	DestinationRef	**1:1**	*xs:NMTOKEN*	Index of the target text.
	DestinationName	0:*	*+International Text*	Target text.
	DestinationShortName	0:*	*+International Text*	Destination short name.

Table 268: Description of structure DestinationStructure

22.3.9 ViaPointStructure

ViaPointStructure			*+Structure*	**Information about intermediate stops.**
	ViaPointRef	**1:1**	*xs:int*	Index of the stopping point within the list of intermediate stopping points.
	PlaceRef	0:1	→*StopPoint*	Reference to the stopping point, see 7.5.1.
	PlaceName	0:*	*+International Text*	Name of the intermediate stopping point.
	PlaceShortName	0:*	*+International Text*	Short name of the intermediate stopping point.
	ViaPointDisplayPriority	0:1	*xs:nonNegativeInteger*	Display priority of the intermediate stopping point.

Table 269: Description of structure ViaPointStructure

22.3.10 AnnouncementStructure

AnnouncementStructure			*+Structure*	**Information about individual stop announcement**
	AnnouncementRef	**1:1**	*xs:NMTOKEN*	Index of the announcement.
	AnnouncementText	0:*	*+International Text*	Announcement text as information legible for the passenger.
	AnnouncementTTSText	0:*	*+International Text*	Announcement text, for a TextToSpeech system.

Table 270: Description of structure AnnouncementStructure

22.3.11 ConnectionStructure

ConnectionStructure			*+Structure*	**Information about individual connections to a stop including neighbouring stops.**
	ConnectionRef	**1:1**	*xs:NMTOKEN*	Index about stopping points across all available connections.
	ConnectionType	**1:1**	*Interchange | ProtectedConnection*	Type of connection (saved connection or simple interchange relation).
	ConnectionStop	0:1	→*StopPoint*	Reference to a neighbouring stopping point; see 7.5.1. If not available, the connection takes place at the same stopping point.
	DisplayContent	**1:1**	*+DisplayContent*	Information about the composition of the target text display content; see 22.3.6.
	Platform	**1:1**	*xs:string*	Departure point of the distributor vehicle.
	WalkDuration	0:1	*xs:duration*	Average walking time to the departure stop of the connection.
	ConnectionState	**1:1**	*ConnectionOK | Connection-Broken | NoInformationAvailable*	Information whether the connection can be established or not.
	Transportmode	0:1	*+VehicleType*	Information about the type of distributor vehicle; see 22.3.12.
	ExpectedDepartureTime	0:1	*xs:dateTime*	Expected departure time of the distributor vehicle.

Table 271: Description of structure ConnectionStructure

22.3.12 VehicleTypeStructure

VehicleTypeStructure			*+Structure*	**Information about the type of a vehicle.**
	VehicleTypeRef	**1:1**	→*VehicleType*	Information about the vehicle type and hence the vehicle facility, see 7.4.1.
	VehicleName	0:*	*+InternationalText*	Name of vehicle type.

Table 272: Description of structure VehicleTypeStructure

23 Dienst Fahrzeugaktionen

23.1 Beschreibung

Dieser Dienst dient der Übertragung eines Interaktionswunsches (z.B. eines Haltewunsches) von einer mobilen Applikation eines Fahrgasts an das Fahrzeug.

In der XML-Schema-Definition *Trias_VehicleInterface.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Fahrzeugaktionen verwendet werden.

23 Vehicle interaction service

23.1 Description

This service helps in transmitting an interaction wish (e.g. a stop request) of a mobile application of a passenger to the vehicle.

Data types and structures are defined in the XML schema definition Trias_VehicleInterface.xsd which are used for the vehicle interaction service.

23.2 Request structures

23.2.1 VehicleInteractionRequestStructure

VehicleInteractionRequestStructure				*+Structure*	**Request which should trigger an interaction with a vehicle.**
	a	***ActivateOutsideSpeakerRequest***	**-1:1**	*+ActivateOutsideSpeakerRequest*	Request in order to activate the outside speaker of a vehicle. See 23.2.2.
	b	***StopRequestRequest***		*+StopRequestRequest*	Request to transmit a stop request to the vehicle. See 23.2.3.

Table 273: Description of structure VehicleInteractionRequestStructure

23.2.2 ActivateOutsideSpeakerRequestStructure

ActivateOutsideSpeakerRequestStructure			*+Structure*	**Request to activate vehicle outside speaker.**
	ActivateOutsideSpeaker	0:1	*xs:boolean*	States whether the outside speaker should be activated.

Table 274: Description of structure ActivateOutsideSpeakerRequestStructure

23.2.3 StopRequestRequestStructure

StopRequestRequestStructure			*+Structure*	**Summarises the information about the request structure in case of a stop request to the vehicle.**
	StopRef	**1:1**	*→StopPoint*	Reference to desired alighting stops, see 7.5.1.
	StopName	0:*	*+International Text*	Identifier of alighting stop.
	Intention	0:1	*Boarding | Alighting*	Specifies the reason why the stop request was triggered. Default is *Alighting*.
	AssistanceRequired	0:1	*xs:boolean*	Passenger needs assistance to board/alight. Default is *false*.
PassengerProfile	*WheelchairUser*	0:1	*xs:boolean*	Passenger uses a wheelchair. Default is *false*.
	WalkingFrame	0:1	*xs:boolean*	Passenger uses a walking frame. Default is *false*.
	WalkingStick	0:1	*xs:boolean*	Passenger uses a walking stick. Default is *false*.
	WalkingImpaired	0:1	*xs:boolean*	Passenger cannot walk. Default is *false*.
	Pram	0:1	*xs:boolean*	Passenger carries a pram. Default is *false*.
	HeavyLuggage	0:1	*xs:boolean*	Passenger carries heavy luggage. Default is *false*.
	VisuallyImpaired	0:1	*xs:boolean*	Passenger is visually impaired. Default is *false*.
	HearingImpaired	0:1	*xs:boolean*	Passenger is hearing-impaired. Default is *false*.
	ReadingImpaired	0:1	*xs:boolean*	Passenger is reading-impaired. Default is *false*.

Table 275: Description of structure StopRequestRequestStructure

23.3 Response structures

23.3.1 VehicleInteractionResponseStructure

VehicleInteractionResponseStructure				*+Structure*	**Response of the vehicle to an interaction wish.**
	a	***ActivateOutsideSpeakerResponse***	**-1:1**	*+ActivateOutsideSpeakerResponse*	Vehicle response to the activation of outside speaker. See 23.3.2.
	b	***StopRequestResponse***		*+StopRequest Response*	Vehicle response to a stop request. See 23.3.3.

Table 276: Description of structure VehicleInteractionResponseStructure

23.3.2 ActivateOutsideSpeakerResponseStructure

ActivateOutsideSpeakerResponseStructure			*+Structure*	**Request to activate vehicle outside speaker.**
	OutsideSpeakerActivated	**1:1**	*xs:boolean*	States whether the outside speaker was activated.

Table 277: Description of structure ActivateOutsideSpeakerResponseStructure

23.3.3 StopRequestResponseStructure

StopRequestResponseStructure			*+Structure*	**Summarises the information about the response structure in case of a stop request to the vehicle.**
	StopRequestReceived	**1:1**	*xs:boolean*	Information that the stop request has arrived.
	RequestedStop	0:1	*xs:NMTOKEN*	Reference to desired alighting stops, see 7.5.1.
	EstimatedArrivalTime	0:1	*xs:dateTime*	Estimated arrival time at the alighting stop.

Table 278: Description of structure StopRequestResponseStructure

24 Dienst Diensteregister

24.1 Beschreibung

Der Dienst Diensteregister führt Buch über verfügbare TRIAS-Dienste.

In der XML-Schema-Definition *Trias_ServiceRegister.xsd* werden Datentypen und Strukturen definiert, die für den Dienst Diensteregister verwendet werden.

24 Service register service

24.1 Description

The service register service keeps accounts of available TRIAS services.

Data types and structures are defined in the XML schema definition Trias_ServiceRegister.xsd which are used for the service register service.

24.2 Simple types

The following simple types are defined:

Type name	***Basic types and values***	**Type description**
TriasServiceIdType	*xs:NMTOKEN*	**ID of an instance of a TRIAS service.**
InterfaceVersionType	*xs:NMTOKEN*	Version number of interface definition service.
TriasServiceTypeEnumeration	*Alerts* | *BookingInfo* | *ConnectionDemand* | *Facilities* | *Fares* | *IndividualRoutes* | *IndividualRoutesRefine* | *Locations* | *LocationsRefine* | *Maps* | *Positioning* | *ServiceRegister* | *StopEvents* | *StopEventsRefine* | *TripInfo* | *TripInfoRefine* | *Trips* | *TripsRefine*	Type of service.
ServiceAddressType	*xs:anyURI*	Address (URL) of an online service.
ServiceUsageEnumeration	*Consumer* | *Provider*	Use of service as provider or client.

Table 279: Description of simple types

24.3 Request structures

A request to the service register is sent via an element ServiceRegisterRequest of the type ServiceRegisterRequestStructure.

24.3.1 ServiceRegisterRequestStructure

With a request of the type ServiceRegisterRequestStructure, a TRIAS service can be included, deleted or updated in the service register or all the registered services can be searched which fulfil the filter criteria stated.

ServiceRegisterRequestStructure				*+Structure*	**Summarises the request data to the service register.**
	a	***RegisterRequest***	**-1:1**	*+ServiceRegisterRegisterRequest*	Request in order to register a service instance in the service register. See 24.3.3.
	b	***UpdateRequest***		*+ServiceRegisterUpdateRequest*	Request in order to update a service instance in the service register. See 24.3.4.
	c	***LookupRequest***		*+ServiceRegisterLookupRequest*	Request to look up for suitable services in the service register. 24.3.5.
	d	***UnregisterRequest***		*+ServiceRegisterUnregisterRequest*	Request in order to delete a service instance in the service register. See 24.3.6.
	Params		0:1	*+ServiceRegisterParam*	Request parameters. See 24.3.2.

Table 280: Description of structure ServiceRegisterRequestStructure

24.3.2 ServiceRegisterParamStructure

ServiceRegisterParamStructure			*+Structure*	**Summarises the parameters for a request to the service register.**
	Extension	0:1	*xs:anyType*	Extensions.

Table 281: Description of structure ServiceRegisterParamStructure

24.3.3 ServiceRegisterRegisterRequestStructure

ServiceRegisterRegisterRequestStructure			*+Structure*	**Request to the service register in order to register a service.**
TriasServiceProperties	*ServiceType*	0:1	*TriasServiceTypeEnumeration*	Type of service.
	Version	0:1	*InterfaceVersion*	Version number of interface definition service.
	ServiceAddress	0:1	*ServiceAddress*	Address (URL) of an online service.
	ParticipantRef	0:1	*→ParticipantCode*	ID of a communication partner. See 7.4.1.
	ServiceUsage	0:1	*Consumer | Provider*	Use of service as provider or client.

Table 282: Description of structure ServiceRegisterRegisterRequestStructure

24.3.4 ServiceRegisterUpdateRequestStructure

ServiceRegisterUpdateRequestStructure			*+Structure*	**Request to the service register in order to update the entries for a service.**
	ServiceId	**1:1**	*TriasServiceId*	ID of the service which should be updated.
TriasServiceProperties	*ServiceType*	0:1	*TriasServiceTypeEnumeration*	Type of service.
	Version	0:1	*InterfaceVersion*	Version number of interface definition service.
	ServiceAddress	0:1	*ServiceAddress*	Address (URL) of an online service.
	ParticipantRef	0:1	*→ParticipantCode*	ID of a communication partner. See 7.4.1.
	ServiceUsage	0:1	*Consumer | Provider*	Use of service as provider or client.

Table 283: Description of structure ServiceRegisterUpdateRequestStructure

24.3.5 ServiceRegisterLookupRequestStructure

ServiceRegisterLookupRequestStructure			*+Structure*	**Request to the service register in order to find suitable services.**
	ServiceId	0:1	*TriasServiceId*	ID of the service which is searched.
TriasServiceProperties	*ServiceType*	0:1	*TriasServiceTypeEnumeration*	Type of service.
	Version	0:1	*InterfaceVersion*	Version number of interface definition service.
	ServiceAddress	0:1	*ServiceAddress*	Address (URL) of an online service.
	ParticipantRef	0:1	*→ParticipantCode*	ID of a communication partner. See 7.4.1.
	ServiceUsage	0:1	*Consumer | Provider*	Use of service as provider or client.

Table 284: Description of structure ServiceRegisterLookupRequestStructure

24.3.6 ServiceRegisterUnregisterRequestStructure

ServiceRegisterUnregisterRequestStructure			*+Structure*	Request to the service register in order to delete a service.
	ServiceId	0:1	*TriasServiceId*	ID of the service which should be deleted.

Table 285: Description of structure ServiceRegisterUnregisterRequestStructure

24.4 Response structures

The result of a service register request is sent via an element ServiceRegisterResponse of the type ServiceRegisterResponseStructure.

24.4.1 ServiceRegisterResponseStructure

ServiceRegisterResponseStructure			*+Structure*	**Summarises the result data for a service register request.**
	ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the overall response of the request. Refer to the following table for possible values. See 7.4.2.
	ServiceRegisterResult	0:1	*+ServiceRegisterResult*	Structure for a service register result. See 24.4.2.

Table 286: Description of structure ServiceRegisterResponseStructure

In ErrorMessage, the following error states can appear:

Error code	**Error description**
SERVICEREGISTER_SERVICEIDUNKNOWN	The request to the service register contains an unknown service ID.
SERVICEREGISTER_NOMATCH	The search request to the service register provides no match.
SERVICEREGISTER_TOOMANYMATCHES	The search request to the service register provides too many matches.

Table 287: List of error states in ServiceRegisterResponse

24.4.2 ServiceRegisterResultStructure

ServiceRegisterResultStructure				*+Structure*	**Result structure for service register request.**
		ResultId	**1:1**	*xs:NMTOKEN*	ID of the result for subsequent referencing.
		ErrorMessage	0:*	*+ErrorMessage*	Error messages based on the request to service register. Refer to the following table for possible values. See 7.4.2.
	a	***RegisterResponse***	**-1:1**	*+ServiceRegisterRegisterResponse*	Response to register service. See 24.4.3.
	b	***UpdateResponse***		*+ServiceRegisterUpdateResponse*	Response to update service. See 24.4.4.
	c	***LookupResponse***		*+ServiceRegisterLookupResponse*	Response for searching for suitable services. See 24.4.5.
	d	***UnregisterResponse***		*+ServiceRegisterUnregisterResponse*	Response to deletion service. See 24.4.6.

Table 288: Description of structure ServiceRegisterResultStructure

The use of ErrorMessage in ServiceRegisterResultStructure is reserved for subsequent extensions.

24.4.3 ServiceRegisterRegisterResponseStructure

ServiceRegisterRegisterResponseStructure			*+Structure*	**Response to register service.**
	ServiceId	**1:1**	*TriasServiceId*	ID of the service as is entered in the register. This ID must be used for subsequent requests to the service register.

Table 289: Description of structure ServiceRegisterRegisterResponseStructure

24.4.4 ServiceRegisterUpdateResponseStructure

ServiceRegisteUpdateResponseStructure			*+Structure*	**Response to update service.**
	ServiceId	**1:1**	*TriasServiceId*	ID of the service as is entered in the register. This ID must be used for subsequent requests to the service register.

Table 290: Description of structure ServiceRegisterUpdateResponseStructure

24.4.5 ServiceRegisterLookupResponseStructure

ServiceRegisteLookupResponseStructure			*+Structure*	**Response to search service.**
	Service	**1:***	*TriasService*	One or more services which fulfil the request criteria. See 24.4.7.

Table 291: Description of structure ServiceRegisterLookupResponseStructure

24.4.6 ServiceRegisterUnregisterResponseStructure

ServiceRegisteUnregisterResponseStructure			*+Structure*	**Response to deletion service from the register.**
				The response element remains empty for the time being.

Table 292: Description of structure ServiceRegisterUnregisterResponseStructure

24.4.7 TriasServiceStructure

TriasServiceStructure			*+Structure*	**Definition of an instance of a TRIAS service.**
	ServiceId	**1:1**	*TriasServiceId*	ID of the instance.
TriasServiceProperties	*ServiceType*	0:1	*TriasServiceTypeEnumeration*	Type of service.
	Version	0:1	*InterfaceVersion*	Version number of interface definition service.
	ServiceAddress	0:1	*ServiceAddress*	Address (URL) of an online service.
	ParticipantRef	0:1	*→Participant Code*	ID of a communication partner. See 7.4.1.
	ServiceUsage	0:1	*Consumer | Provider*	Use of service as provider or client.

Table 293: Description of structure TriasServiceStructure

25 Dienst Authentifizierung

Einige Teile der TRIAS-Schnittstelle werden zum Austausch von wichtigen betrieblichen Daten verwendet. Um die Unverfälschtheit dieser Daten sicherzustellen, sind Mechanismen zur Authentifizierung und zur Autorisierung der Schnittstellenpartner notwendig. In diesem Kapitel soll daher ein Mechanismus beschrieben werden, um die Authentizität von Schnittstellenpartnern überprüfen zu können.

Zur Authentifizierung der Schnittstellenpartner kommt ein PKI-Verfahren zum Einsatz. Es wird die Verwendung des DSA-Verfahrens[14] empfohlen. Jedoch können auch andere Verfahren zum Einsatz kommen. Die Kommunikationspartner müssen sich dazu bilateral abstimmen.

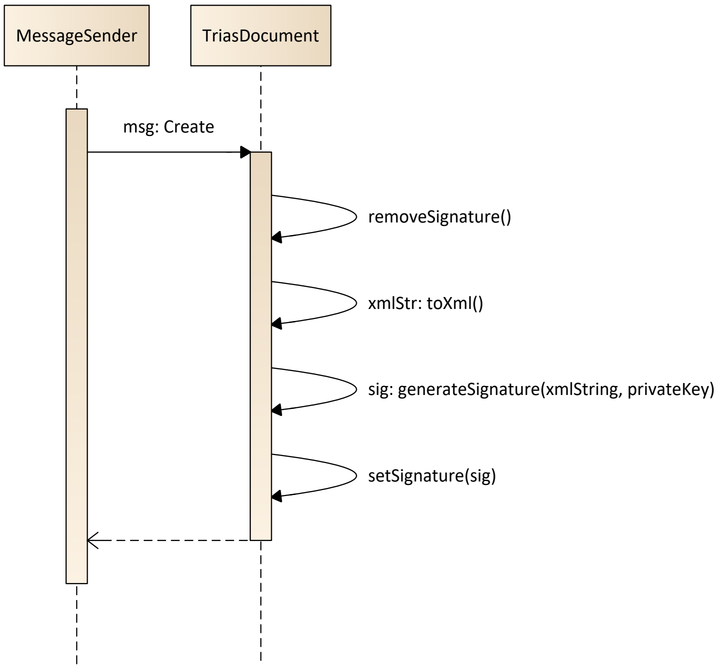

Abbildung 10: Erzeugung einer Nachrichtensignatur

[14] Digital Signature Standard: http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

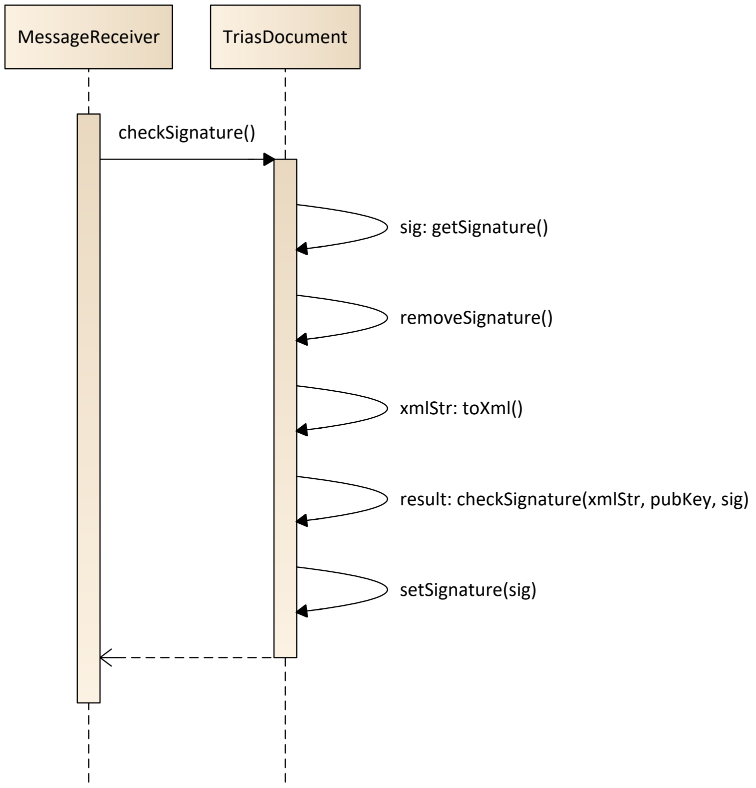

Abbildung 11: Überprüfung einer Signatur

Der Authentifizierungsdienst des Empfängers überprüft die CertificateId und die SignatureId aus dem ServiceRequest (vgl. die Definition der Nachrichteneigenschaften in 7.9.2). Das Feld CertificateId enthält eine Referenz auf diesen Schlüssel. Der Schlüssel muss also vorab ausgetauscht werden. Das Feld SignatureId enthält die Signatur der versendeten TRIAS-Nachricht. Zur Berechnung der Signatur wird die XML-Nachricht in der kanonischen Normalform[15] verwendet. Bei der Berechnung der Signatur darf das Feld SignatureId in der XML-Nachricht nicht vorhanden sein, es muss also vom Empfänger wieder entfernt werden. Dabei muss die kanonische Normalform erhalten bleiben.

Der Ablauf der Signierung einer Nachricht ist schematisch in Abbildung 10 dargestellt. Zunächst wird das Feld SignatureId komplett aus der Nachricht entfernt. Dann wird die Nachricht in ihre XML-Repräsentation in der kanonischen Normalform exportiert. Mithilfe der XML-Repräsentation und dem privaten Schlüssel kann nun die korrekte Signatur der Nachricht berechnet und im Feld SignatureId gespeichert werden. Das Ergebnis ist die korrekt signierte, und vor Modifikation durch Dritte geschützte Nachricht.

Der schematische Ablauf der Signaturprüfung ist in Abbildung 11 dargestellt. Zunächst wird das Feld SignatureId aus der Nachricht entfernt und die Nachricht in ihre kanonische XML-Darstellung überführt. Anschließend kann mithilfe des öffentlichen Schlüssels des Nachrichtensenders, der Nachricht in XML-Darstellung und der Signatur überprüft werden, ob diese Signatur gültig ist. Anschließend muss, um die Nachricht nicht zu verfälschen, die Signatur wieder auf ihren ursprünglichen Wert gesetzt werden.

[15] RFC 3076: http://www.ietf.org/rfc/rfc3076.txt

25 Authentication service

A few parts of the TRIAS interface are used for exchanging important operating data. In order to ensure the authenticity of this data, mechanisms are necessary for authenticating and authorising interface partners. Hence, a mechanism should be described in this chapter which can check the authenticity of interface partners.

A PKI procedure is used for authenticating the interface partners. The use of a DSA procedure[16] is recommended. However, other procedures can also be used. The communication partners must mutually agree.

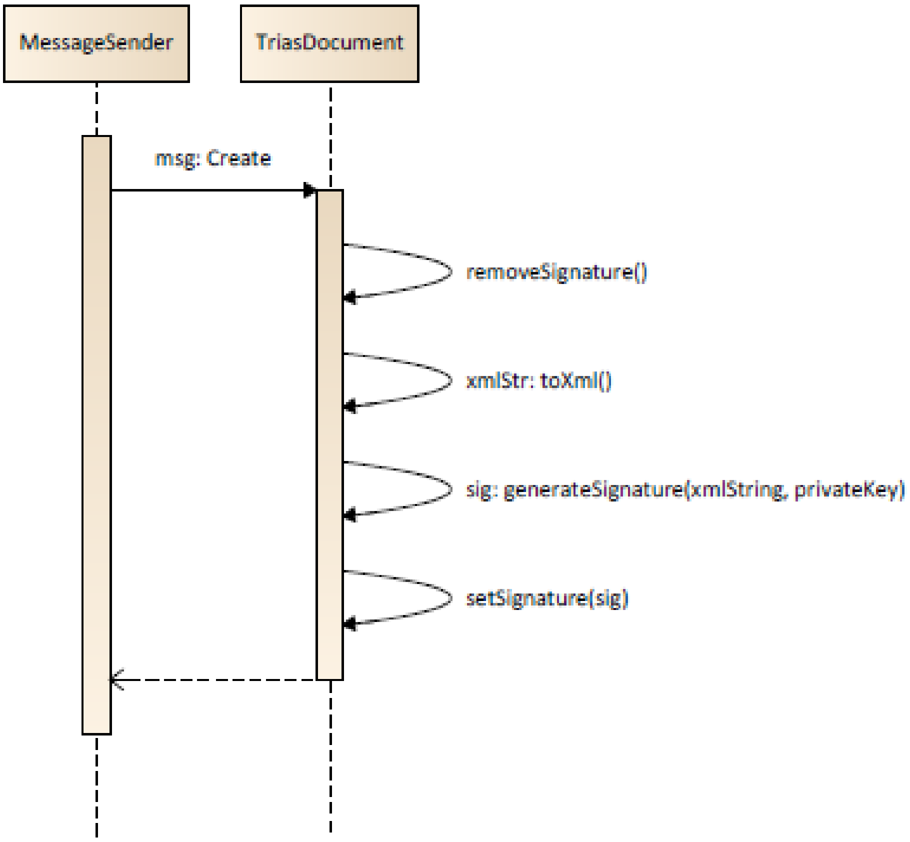

Figure 10: Addition of a message signature

[16] Digital Signature Standard: http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

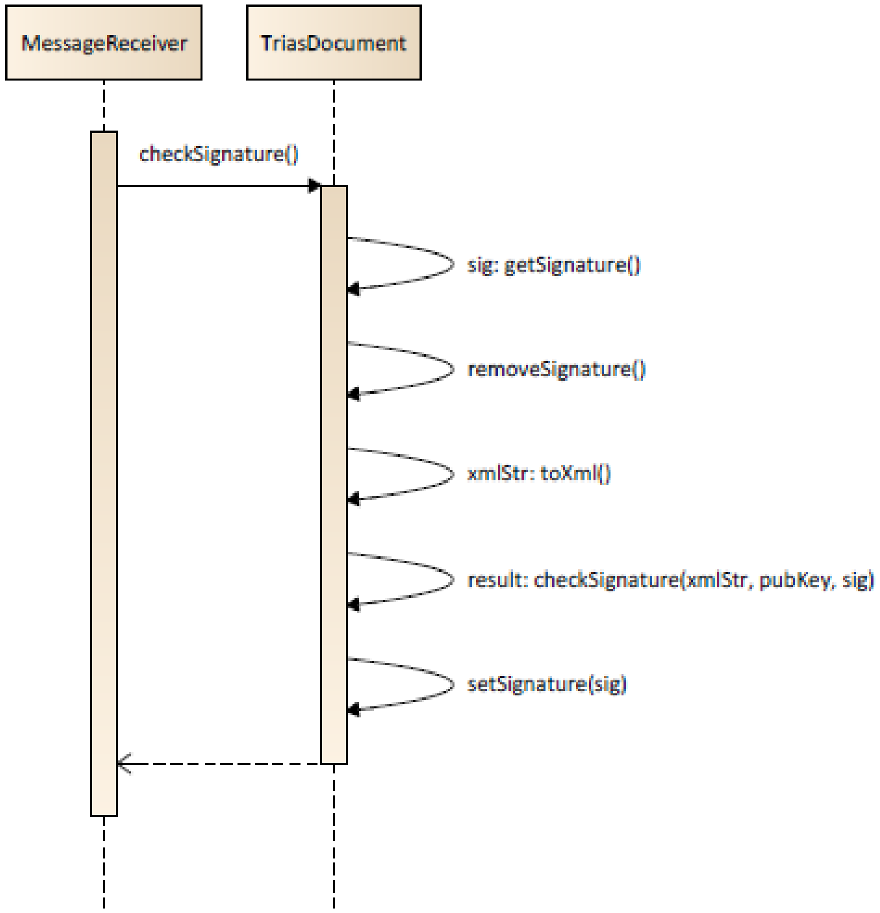

Figure 11: Checking a signature

The authentication service of the receiver checks the CertificateId and SignatureId from ServiceRequest (refer to the definition of message properties in 7.9.2). The field CertificateId contains a reference to this key. The key must also be exchanged in advance. The field SignatureId contains the signature of the TRIAS message sent. The XML message is used in canonical normal form 9 for calculating the signature. When calculating the signature, the field SignatureId must not be present in the XML message. It must be removed by the receiver again. In the process, a canonical normal form must be retained.

The process of signing a message is schematically shown in Figure 10. At first, the field SignatureId is completely removed from the message. Then the message is exported in its XML representation in canonical normal form. With the help of the XML representation and the private key, the correct signature of the message can now be calculated and saved in the field SignatureId. The result is a correctly signed message protected against modification by third parties.

The schematic sequence of signature check is shown in Figure 11. At first, the field SignatureId is completely removed from the message and the message is converted into its canonical XML representation. It can then be checked whether this signature is valid with the help of the public key of the message sender, the message in XML representation and the signature. In order to ensure that the message is not distorted, the signature must again be set to its original value.

26 Version history

26.1 Version 1.1 (regulation and schema)

26.1.1 Functional extensions

- Schema version in root element increased to “1.1”.
- New structure WebLinkStructure (link + description) defined as substitute for the element of type xs:anyUri which represents only one link without description.
- A new structure FareZoneStructure is included in Trias_FaresSupport.xsd in order to be able to specify tariff zone names for the references to tariff zone objects. Is used, for example in PassedZones or ZonesAlreadyPaid.
- A new element FaresAuthorityText is set aside for the element FaresAuthorityRef, in order to be able to appoint tariff authorities..
- Additional elements ValidityDuration, ValidityDurationText, ValidityFareZones and ValidityAreaText in TicketStructure.
- StaticFaresRequestStructure now offers the option to specify a TicketID in order to retrieve information especially about this ticket.
- FaresPassengerStructure extended by ZonesAlreadyPaid and OwnedTickets, in order to be able to model the already existing driving authorisations.
- POI categories added as a list of Key-Value pairs (attributes from Open Street Map). A POI can be assigned to one or more POI categories. The search for POIs can be filtered according to these POI categories (in LocationParamStructure).
- New element OperatorFilter included in LocationParamStructure.
- New element SharingService for modelling rental vehicle suppliers (Car-Sharing, Bike-Sharing) included in ContinuousServiceStructure.
- The elements Origin and Destination are now also permitted shared in TripRequestStructure.

26.1.2 Technical additions/corrections

- Version numbers removed from the file names of schema files.
- Multiple import instructions removed from siri namespace per schema file. Instead, import of SIRI main schema file.
- Uniform assignment of TRIAS namespace “www.vdv.de/trias”.
- Correction of spelling of Manoeuvre in NavigationSectionStructure.
- In the notification service (TripMonitoringPolicyGroup), the default value false included for the elements AcceptThirdPartyInformation and IncludeAlternatives.
- Both the unused schema files Trias_Authorisation.xsd and Trias_PushToDevice.xsd removed. These were remnants from the research project IP-KOM-ÖV which are no longer required.
- Typing error in the element IndvidualTransportOptions (sic!) corrected in LocationContextStructure.

- OriginStopPointRef or DestinationStopPointRef are now optional in DatedJourneyStructure and ContinuousServiceStructure if ServiceOriginGroup or ServiceDestinationGroup is requested to be used.
- Additional error codes TRIASGENERIC_ERROR, TRIASGENERIC_SERVICENOTSUPPORTED, TRIASGENERIC_REQUESTNOTSUPPORTED and TRIASGENERIC_FEATURENOTSUPPORTED defined.
- Trias_JourneySupport.xsd:DatedCallAtLocationStructure->ServiceDeparture is
- <xs:documentation>Arrival times of the service at this stop.</xs:documentation> As it is the ServiceDeparture element, now it is "departure times" instead.

26.1.3 Documental corrections

- In TicketStructure the documentation of the elements InfoUrl and SaleUrl are added.
- In AddressStructure the element PrivateCode is moved to the correct position.
- In AddressStructure das element CityName is renamed to the correct name LocalityName.
- In AddressStructure the documentation of the element LocalityRef is added.
- References to location objects in the child element present in the schema altered in LocationStructure.
- Diverse typing errors, formatting errors and incorrect/missing cross references corrected.
- Explanation about the operating day and operating day code added in section 5.9
- Section 5.3 Addition with reference to OSM
- Addition 7.5.5. PointOfInterestCategoryStructure and 7.5.6. OsmTagStructure
- Additions and explanations in chapter 19 to the sequence (DataReceivedAcknowledgement)

26.2 Version 1.2 (regulation and schema)

26.2.1 Functional extensions

- Refine service (see RefineRequest in chap. 15) added as new service.
- Two new optional elements included in StopCallStatusGroup: NoBoardingAtStop and NoAlightingAtStop.
- New optional element JourneyTrack in TripInfoResultStructure.
- In TripInfoParamStructure two additional filter elements IncludeTrackSections and IncludeTrackProjection.
- BaseTripContentFilterGroup was extended by IncludeEstimatedTimes and IncludeSituationInfo.
- An option was created to illustrate the change in trip properties (e.g. train number, type). For this purpose, the new ServiceSectionStructure was created and ServiceJourneyGroup and ServiceGroup remodelled.

- ServiceAttributeStructure was extended by an element Scope in order to be able to state the reference of the attribute.
- Parallel trips can now be stated in sections (e.g. portion working).
- For this purpose, a new ParallelServiceStructure was defined and integrated in TimedLegStructure, TripInfoResultStructure as well as StopEventStructure.
- It is now possible to request boarding or alighting help; the connection service and service for vehicle information have been correspondingly extended to this end.
- The vehicle service was extended by the option of making announcements on the outside speaker. In this context, the functionality was allocated within the vehicle service in interaction and information.
- Extension of request structures for connections and stop request by a PassengerProfileGroup, in which generic mobility limitations are shown.
- Extension of TripInfoContentFilterGroup by IncludeEstimatedTimes and IncludeSituationInfo.

26.2.2 Technical additions/corrections

- All optional elements of the type xs:boolean were converted from FIXED values to DEFAULT values in the schema. The valid default values are stated in the regulation.
- The mandatory element StopSeqNumber (in structures CallAtStopStructure, DatedCallAtLocationStructure, LegBoardStructure, LegAlightStructure and LegIntermediateStructure) is now optional.
- The data structure not used RouteDescriptionGroup was removed from the schema.
- Elements of the type InternationalTextStructure can now occur multiple times to facilitate multilingualism.
- The element Language was removed from LocationParamStructure.
- In Requests (ServiceRequestContext, SubscriptionRequestContext) the client can now specify his preferred languages.
- In Responses (ServiceResponseContext) EKAP can define several languages.
- The element Location in LocationInformationResponseStructure was renamed to LocationResult.
- The coding UTF-8 is explicitly stipulated for all XML contents.
- JourneyAttributeStructure was replaced by ServiceAttributeStructure.
- LegAttribute was removed by TimeLegStructure. Trip attributes are stated in ServiceJourneyGroup under DatedJourneyStructure instead.
- Summary of StopRequestRequestStructure and ActivateOutsideSpeakerRequestStructure in a VehicleInteractionRequestStructure.
- FeederDistributorStructure and DatedCallAtLocationStructure were extended by LineDirectionGroup and OperatorRef so that control centres can reliably recognise a trip.
- TicketStructure extended by TariffLevelGroup for depicting price and fare levels.
- Introduction of a type ServiceCallStructure which includes ServiceTimeGroup. Use of this type in TimeStopStructure, LegBoardStructure, LegAlightStructure, LegIntermediateStructure, CallAtStopStructure and DatedCallAtLocationStructure in order to incorporate information about the arrival or departure at a point.

26.2.3 Documental additions/corrections

- Chapter 6was contextually extended and correspondingly renamed.
- The new section 6.5 was added for explaining stop sequence numbers and trip sections.
- In the subchapter 17.2.3, the name of the element IndividualRouteContextStructure in IndividualRouteLocationContextStructure was corrected.
- In the subchapter 12.3.3 the data type PreviousCall and OnwardCall was changed to CallAtNearStop and CallAtStop.
- In the subchapter 13.8.1, the name of the element Extensions in Extension was corrected.

26.3 Version 1.3 (regulation and schema)

26.3.1 Functional extensions

- Added the ability for storing, retrieving, and deleting of several values at once in PersonalisationService.
- Added the ability for storing, retrieving, and deleting of trips (several at once) in PersonalisationService.
- Added MultiPoint functionality from OJP to TripRequest, including TimeAllowance for each location context and a policy filter defining how to handle the case of several origin/destination inputs.

26.3.2 Technical additions/corrections

- Added UsageValidityType to TicketStructure and added the UsageValidityTypeEnumeration based on NeTEx
- Added ResponseContexts to RefineRequests (IndividualRouteRefineRequest, StopEventRefineRequest, TripInfoRefineRequest, TripRefineRequest)
- NumberOfResultsBefore and NumberOfResultsAfter in NumberOfResultsGroup may be 0
- Added FacilityRef to Accesspath
- Chapter 7.4.1: Changed types of
  ParticipantCodeType, OperatorCodeType, LineCodeType, DirectionCodeType, JourneyCodeType, VehicleCodeType, FacilityCodeType, OwnerCodeType, OperatingDayCodeType
  from xs:NMTOKEN to xs:normalizedString

26.3.3 Documental additions/corrections

- Document was translated to English. All explanation issues are documented bilingual (German & English) and all technical issues are only documented in English
- Specification of Line-ID and Direction-ID. If ID’s are not available, values NO_LINE and NO_DIRECTION have to be used.

- Reference made to CEN TS16614-NeTEx Part 3 (2016) Public transport - Network and Timetable Exchange (NeTEx) - Part 3: Public transport fares exchange format
- Changed in the German part: “Verkehrstag” to “Betriebstag”

Abbildungsverzeichnis / List of figures

Tabellenverzeichnis / List of tables

Regelwerke – Normen, Empfehlungen / Standards, recommendations

(1)	CEN, EN 12896:2006	Reference Data Model for Public Transport
(2)	CEN, EN 28701:2012	Intelligent transport systems - Public transport - Identification of Fixed Objects in Public Transport (IFOPT)
(3)	CEN, TS 15531 Part 1. (2011)	SIRI - Service Interface for Realtime Information, Part 1.
(4)	CEN, TS 15531 Part 2. (2011)	SIRI - Service Interface for Realtime Information, Part 2
(5)	CEN, TS 15531 Part 5	SIRI - Service Interface for Realtime Information, Part 5
(6)	CEN TS16614- Part 3 (2016)	Public transport - Network and Timetable Exchange (NeTEx) - Part 3: Public transport fares exchange format
(7)	Journeyweb	Department for Transport. (19. 04 2012). Dft - Journeyweb. Called on 15.05.2013 by http://www.dft.gov.uk/journeyweb/
(8)	ISO 8601:2004. (2004)	Data elements and interchange formats - Information interchange - Representation of dates and times. ISO, International Organisation for Standardisation.
(9)	VDV 430 (2014)	Customer interface architecture
(10)	VDV 431 -1 (2014)	System architecture EKAP
(11)	VDV 432 (2016)	Identification of stops - use of global ID in Germany

All VDV documents are available on https://www.vdv.de/ip-kom-oev.aspx, the according XSD File is avaliable on https://github.com/VDVde

Impressum

Verband Deutscher Verkehrsunternehmen e. V. (VDV)
Kamekestraße 37-39 · 50672 Köln
T 0221 57979-0 · F 0221 57979-8000
info@vdv.de · www.vdv.de

Ansprechpartner

Dipl.-Ing. Berthold Radermacher
T 0221 57979-141
F 0221 57979-8141
radermacher@vdv.de

Verband Deutscher Verkehrsunternehmen e. V. (VDV)
Kamekestraße 37-39 · 50672 Köln
T 0221 57979-0 · F 0221 57979-8000
info@vdv.de · www.vdv.de