

Public transport — Open API for distributed journey planning

Einführendes Element — Haupt-Element — Ergänzendes Element

Élément introductif — Élément central — Élément complémentaire

ICS:

Descriptors:

Document type: Technical Specification

Document subtype:

Document stage: Working Document

Document language: E

C:\Data\Clients\Ministere\SG8\Alignement\Doc\DJPS-final3\TC_278_WI_00278420_(E)-RS-170118-final3.doc STD Version 2.7g

Contents

	Page
European Foreword.....	5
0 Introduction	6
0.1 General.....	6
0.2 An Open API for distributed journey planning (OJP)	6
0.3 The public transport information tensions	7
0.4 Distributed journey planning architecture beyond scope	8
0.4.1 The distributed journey planning approach	8
0.4.2 Distributed or centralised approaches	8
0.4.3 The basis for the Open API.....	9
0.4.4 Other possible uses for the Open API.....	9
0.5 The European ITS Directive	9
1 Scope	10
2 Normative references	10
3 Terms and definitions	11
3.1 Introduction	11
3.2 Explanation of terms used in OJP schema	11
3.3 Explanation of extension terms in OJP Schema.....	22
4 Symbols and abbreviations	24
5 Use cases.....	25
5.1 General.....	25
5.2 Key tasks for Distributed Journey Planning.....	26
5.2.1 Planning a component of a trip	26
5.2.2 Discovering relevant stops	27
5.2.3 Obtaining information about accessibility and services for those with special needs.....	27
5.2.4 Seeking route information that can be displayed on maps	27
5.3 Other possible tasks for a Distributed Journey Planning system.....	28
5.3.1 Requesting a stop timetable.....	28
5.3.2 Requesting times for all intermediate stops in a trip	28
5.3.3 Requesting expected events at a particular stop.....	28
5.3.4 Requesting information about the fares and ticket options for a particular trip.....	28
5.3.5 Other possible questions.....	28
6 System Architectures, Metadata and Data	29
6.1 General considerations	29
6.2 Metadata requirements	30
6.3 Core data requirements.....	31
7 Open API for Distributed Journey Planning – OJP Services	33
7.1 Departure Monitor	33
7.1.1 Purpose.....	33
7.1.2 Interactions	34
7.1.3 Concerned Components.....	34
7.1.4 Function 1: Departure Monitor	34
7.2 Fare Information	35

7.2.1	Purpose	35
7.2.2	Interactions.....	35
7.2.3	Concerned Components	36
7.2.4	Function 1: Tariff Zones for Stop / Station	36
7.2.5	Function 2: Static Fare Information	37
7.2.6	Function 3: Trip-Related Fare Information.....	37
7.3	Location text matching	37
7.3.1	Purpose	37
7.3.2	Interactions.....	38
7.3.3	Concerned Components	38
7.3.4	Function: Location text matching	38
7.4	Object Information Service	38
7.4.1	Purpose	38
7.4.2	Interactions.....	39
7.4.3	Concerned Components	39
7.4.4	Function 1: Object Information	39
7.4.5	Function 2: Finding relevant exchange points	39
8	Open API for Distributed Journey Planning – Interface Description	40
8.1	Notation of XML-Elements and XML-Structures	40
8.1.1	Display of XML Elements in the Text	40
8.1.2	Display of Relationships.....	40
8.1.3	Table Notation of XML Structures.....	41
8.1.4	Message Exchange	43
8.1.5	Use of SIRI Procedure.....	43
8.1.6	HTTP and REST	44
8.1.7	Roles of Server and Client.....	45
8.2	Identification of Objects beyond system borders	45
8.2.1	Stops and Stopping Points	46
8.2.2	Localities and Districts	46
8.2.3	Addresses and POIs.....	47
8.2.4	Organisations: Transport Companies and Transport Authorities.....	47
8.2.5	Lines and Line Directions	47
8.2.6	Journeys	48
8.2.7	Vehicles	49
8.2.8	Operating Days	49
8.2.9	Owners.....	50
8.2.10	Stop- and Vehicle Equipment	50
8.2.11	Participating Systems / IT Systems.....	50
8.2.12	Incident Messages.....	50
8.2.13	Fare Authority.....	50
8.2.14	Tariff Zones	50
8.2.15	Tickets and Traveller Cards.....	50
8.3	Services and XML Schemas.....	51
8.3.1	Services Provided	51
8.3.2	XML Schemas Used Across Services	52
8.3.3	Imported Schemas.....	52
8.3.4	Error States When Operating OJP Services	52
8.3.5	Error Codes from SIRI	53
8.3.6	General OJP Error States	54
8.3.7	Time Zones.....	54
8.4	Common XML Structures.....	54
8.4.1	Root Element OJP	55

8.4.2	OJP_Utility	58
8.4.3	OJP_ModesSupport.....	60
8.4.4	OJP_Common.....	63
8.4.5	OJP_LocationSupport	66
8.4.6	OJP_JourneySupport.....	70
8.4.7	OJP_FacilitySupport.....	77
8.4.8	OJP_SituationSupport	79
8.4.9	OJP_RequestSupport	80
8.4.10	OJP_FareSupport.....	83
8.5	Service Location Information	87
8.5.1	Description	87
8.5.2	Simple Types.....	87
8.5.3	Request Structures	87
8.5.4	Response Structures	90
8.6	Service Exchange Points	91
8.6.1	Description	91
8.6.2	Request Structures	91
8.6.3	Response Structures	92
8.7	Service Intermodal Trip Information	93
8.7.1	Description	93
8.7.2	Distributed Planning of Intermodal Trips	93
8.7.3	Request Structures	95
8.7.4	Response Structures	100
8.8	Service Stop Events.....	109
8.8.1	Description	109
8.8.2	Request Structures	109
8.8.3	Response Structures	110
8.10	Service Tickets and Fare Calculation	114
8.10.1	Description	114
8.10.2	Request Structures	114
8.10.3	Response Structures	116
8.11	Discovery and Capabilities	118
Annex A (informative) A distributed approach to journey planning across Europe		120
Annex B (informative) Lessons from experiences of Distributed Journey Planning to date.....		125
B.1	Introduction	125
B.2	JourneyWeb.....	125
B.3	EU-Spirit	127
B.4	DELFI.....	127
B.5	APII-SIM	128
B.6	Shared journey planning methodology.....	129
Annex C : Bibliography		131

European Foreword

This document (TC 278 WI 00278420) has been prepared by Technical Committee CEN/TC 278 “Intelligent transport systems”, the secretariat of which is held by NEN.

This document is a working document.

History of this document *[this paragraph to be removed after document's approval]*

Following the Technical Review of an earlier draft of this Technical Specification (WI 00278374) it was clear that there was a strong wish for this proposed TS to be more tightly aligned with the terminology and definitions established in the Transmodel, IFOPT (now integrated into Transmodel v6), SIRI and NeTEx standards. France agreed to pay for the necessary additional work to be undertaken and CEN established a new work item to cover this activity. This document presents the result of this additional work.

0 Introduction

0.1 General

The availability of accurate and timely information about public transport (PT) services has been an increasing expectation over the past decade or more and systems have been developed to assist in the compilation and delivery of such information making best use of the rapid advances in information technology (IT) capabilities. Multi-modal information systems typically have started with an urban or regional focus to meet the information needs of those making relatively local journeys – whilst information requirements for longer-distance journeys have been delivered primarily by mono-modal information systems from the rail and airline industries.

However there have been some pioneering systems over the past 10 years that have extended these models, notably

- EU-Spirit in Northern Europe
- JourneyWeb in Great Britain
- DELFI in Germany

These three systems employed different architectures that collated data from multiple sources in order to be able to offer information about longer-distance journeys that involved travel in the area of more than one regional information system in a single transaction. This technique of bringing information together from two or more information systems when necessary is referred to as "distributed journey planning".

The ability to extend such systems to wider applications has been greatly enhanced through the way in which public transport data is now increasingly standardised by following the principles set out in the "Public Transport Reference Data Model" (Transmodel) EN standard, and its related implementation Standards and specifications.

0.2 An Open API for distributed journey planning (OJP)

A review of the three long-established Distributed Journey Planning systems that have been working in Europe over the past 10 years (EU-Spirit, JourneyWeb and DELFI) found that, whilst the architecture of each of these systems was different, the nature of the enquiries sent between the systems, and the content of the responses sent in return, were essentially the same. This suggested that it would be possible to define a single Open Journey Planning API to support all distributed journey planning systems.

The Open Journey Planning API (OJP) will therefore allow a system to engineer just one interface that it can make available widely (to authorised users or openly as they so choose) rather than having to engineer separate APIs for each bipartite exchange arrangement that may be required with other systems.

However existing journey planning systems (and probably some that will be developed in the future) may require their own specific APIs for use with their closest partner systems, where the volume of enquiries is such that efficiency considerations demand a tightly specified API for such clients. The intention of the Open API is to provide an opportunity for just one universal channel to exchange information to lower-volume users – once created then there is little reason not to allow as many users of this API as may wish to use it.

0.3 The public transport information tensions

The greatest use of public transport (in terms of the number of passenger journeys) happens in urban areas where frequent and regular services cater for the needs of relatively short-distance journeys. Usage then declines as journey distances get longer – with inter-regional and international journeys comprising the smallest number of public transport journeys.

However the need for information about PT services is least in areas with frequent and regular services, where passengers quickly get to know about the services they rely on for most of their journeys – and therefore their need to check information systems is relatively infrequent. Longer distance journeys, however, are made less often and for a variety of reasons there is a much greater need to obtain information for such journeys before setting off. So the need for information is greatest for the very journeys that are made least often. It is difficult to make a business case to provide information systems geared specifically to the needs of the longer-distance travellers, therefore. Instead it becomes important to find ways of meeting the information needs of those passengers by using information collated and delivered primarily for the much larger group of those making short-distance journeys.

The distributed journey planning systems mentioned above were a measured response to these tensions, allowing data to be shared across multiple systems in different ways to ensure that someone wanting to plan a journey anywhere in (for instance) Germany could go to their own regional journey planner which collected relevant information from other regional planners in order to satisfy a particular enquiry. In Great Britain Transport Direct provided a national journey planning service collating data from the 11 regional traveline journey planners. And in the Baltic area EU-Spirit collated information from several sources to allow international planning between several European states. In Slovenia a proposal to extend the national journey planning system to a multi-national distributed system was included in the SIJPRIS service, although to date it has not yet been implemented.

Over time technology and techniques have allowed these particular systems to drive greater efficiencies as regional systems have been able to merge into ones covering larger areas. This has reduced the complexity of the distributed journey planning process within the areas covered by these systems. The moves to open data have also enabled larger "consolidated" datasets to be used in journey planners (notably Google Transit, or Traveline's national journey planner in Great Britain) – but these are systems offering much less rich information than that from the more local or regional systems, and the data on them is often less timely than is possible on the local and regional systems.

Taking all these factors into account there remain advantages in the distributed journey planning model notwithstanding the trends towards travel information systems that are designed to cover ever larger areas. Key points would appear to be that distributed systems are sharing the most up to date information from the local authorised source in a way that cannot be achieved with systems that collate data for much larger areas. This is particularly important in areas where local public transport market is deregulated (as has been the case in most of Great Britain since 1986) where bus services can change on any date of an operator's choosing rather than there being only one or two service change dates each year from which any changes of services are known well in advance (as is the case in many other parts of Europe).

In the foreseeable future distributed journey planning will continue to provide an effective mechanism for extending the geographical scope of any journey planner with a minimum of effort – so long as there is a single standardised API as proposed in this Technical Specification which will ensure that only one API would need to be engineered to allow standard questions to be asked of other systems, and to allow answers to standard questions from other systems to be sent in response.

For the enquirer there is one other important advantage of distributed journey planning – and that is that the questions can be asked, and answers read, within a layout that the user is familiar with – and in their own natural language.

0.4 Distributed journey planning architecture beyond scope

Distributed Journey Planning depends not just on there being an available API for the exchange of data. It also requires the system responding to an end-user's enquiry to be able to work out what enquiry to send to one or more other information systems, and how to merge the responses with data from its own repositories in order to create one or more seamless journey plans for the enquirer. There are several different approaches to the "architecture" for distributed journey planning – and these are beyond the scope of this Technical Specification. The following paragraphs, however, outline some of the key considerations that any implementation of distributed journey planning will need to take into account.

0.4.1 The distributed journey planning approach

One of the key considerations for building a distributed journey planning system is to define what supporting data (metadata) is required and where it is to be held. At its simplest the process of making an enquiry typically has several stages :

- a) An enquirer goes to his **home system** and composes an enquiry expressing the location of the start and end points in their own terms or as permitted by the user interface
- b) The enquirer's **home system** seeks to match the enquirer's locations to locations understood by the journey planner, and then converting them into terms (perhaps geographic coordinates) that can be understood by the home and other distributed journey planning engines
- c) The **home system** establishes what questions it needs to ask and from what journey planning systems (both its own and those of one or more distributed partners) it needs to ask for information that it does not already have in its own databases
- d) The **home system** then collates the information received in response to the questions asked of the different systems to create a seamless and efficient journey plan which it can then deliver to the enquirer.

In some systems the **home system** does not itself undertake the distributed journey planning. Instead the **home system** passes that task to a separate distributing journey planning system which completes the process and returns the answers to the **home system**.

For the enquirer it is important to make the process as simple and efficient as possible – so the process of matching locations with system gazetteers can be a critical one. Ideally the enquirer should be able to specify a location as a station or stop name, a topographic place, a street address, a postcode (if this covers a meaningful small area), and possibly Points of Interest. Such data for locations within the geographical scope of the **home system** is likely to be held already – but if the location is outside that geographical scope where does the equivalent data come from? – and how does the home system know that it needs to find data for a distant location?

0.4.2 Distributed or centralised approaches

So one of the key considerations for building a distributed journey planning system is to define what supporting data (metadata) is required and where it is to be held. Somehow the home system needs to be able to recognise that a requested origin or destination is not in its own geographical area – and once it has done that it also needs to recognise which system(s) will be able to provide journey planning answers for that location. One way of managing this would be for a network of distributed journey

planning systems to share a central repository of gazetteers (indexes of geographical entities – localities, addresses, stops & stations, etc) to resolve these questions (and probably to go on to make the necessary enquiries of the relevant journey planning systems) before handing back the information to the originating home system. This would be a centralised model for handling journey enquiries that required the distributed service. Alternatively each participating system could hold gazetteer data for all the participating systems' areas – and the enquiry process could then work on a peer-to-peer decentralised basis. To get an efficient approach it is necessary to consider all required support data – not only the gazetteers, but also how access to the timetable data for long-distance PT services (notably trains, coaches, ferries and flights) can be achieved in a way that allows it to be used in the creation of the effective journey plans.

0.4.3 The basis for the Open API

Recent work in Germany's IP-KOM research project has brought together the lessons learned from various information systems (including EU-Spirit, JourneyWeb and DELFI) and developed the TRIAS schema to support future information systems in Germany. Because this work has brought together the experience of the three long-standing distributed journey planning models, it was decided that it should form the basis of the proposed standard Open API for distributed journey planning. This Technical Specification is based, therefore, on the TRIAS schema developed in Germany with appropriate extensions to meet the full requirements for Distributed Journey Planning.

A number of existing European Standards and Technical Specifications will underpin the work – notably Transmodel (Public Transport Reference Data Model), IFOPT (Identification of Fixed Objects in Public Transport), SIRI (Service Interface for Realtime Information) and NeTEx (Network and Timetable Exchange). The experiences from EU-Spirit, JourneyWeb and DELFI systems will feed into the work, along with references to relevant national implementation standards that have supported Distributed Journey Planning to date. The Open API will depend on the consistency of data from all sources that comes from the implementation of the existing European standards for public transport information.

0.4.4 Other possible uses for the Open API

Whilst the Open API is intended primarily to support distributed journey planning, experience of such APIs to date has shown that they can also be used for other purposes. For instance they can be used for communication between personal journey planning apps and a journey planning service (without any distributed journey planning requirement). Or they can be used to enable a park-and-ride planner to combine car journey planning with public transport journey planning, connecting the two modes at the park-and-ride car parks. Or they can support the use of taxis as a mode to access public transport in areas where conventional public transport does not exist or is very sparse. A standard Open API will provide many opportunities to use and re-use public transport and associated data in the delivery of innovative information services.

0.5 The European ITS Directive

The forthcoming Delegated Regulation of the ITS Directive with regard to the provision of EU-wide multimodal travel information services will provide the necessary requirements to make EU-wide multimodal travel information services accurate and available across borders. It establishes the specifications necessary to ensure the accessibility, exchange and update of travel and traffic data and distributed journey planning for the provision of multimodal information services in the European Union. The Delegated Regulation recommends the use of the Open API standard in relation to the requirements specified in Article 7 'Linking Travel information Services'.

1 Scope

This Technical Specification defines a schema for establishing an Open API for Distributed Journey Planning that can be implemented by any local, regional or national journey planning system in order to exchange journey planning information with any other participating local, regional or national journey planning system.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

CEN/TS 16614-1, *Public transport - Network and Timetable Exchange (NeTEx) - Part 1: Public transport network topology exchange format*

CEN/TS 16614-2, *Public transport - Network and Timetable Exchange (NeTEx) - Part 2: Public transport scheduled timetables exchange format*

EN 15531-1:2015, *Public transport - Service interface for real-time information relating to public transport operations - Part 1: Context and framework (SIRI)*

EN 15531-2:2015, *Public transport - Service interface for real-time information relating to public transport operations - Part 2: Communications (SIRI)*

EN 15531-3:2015, *Public transport - Service interface for real-time information relating to public transport operations - Part 3: Functional service interfaces (SIRI)*

CEN/TS 15531-4:2011, *Public transport - Service interface for real-time information relating to public transport operations - Part 4: Functional service interfaces: Facility Monitoring (SIRI)*

CEN/TS 15531-5:2016, *Public transport - Service interface for real-time information relating to public transport operations - Part 5: Functional service interfaces: Situation Exchange (SIRI)*

EN12896:2006, *Public transport – Reference data model (Transmodel v5.1)*

EN 12896-1:2016, *Public transport - Reference data model – Part 1 : Common concepts (Transmodel v6)*

EN 12896-2:2016, *Public transport - Reference data model - Part 2: Network topology (Transmodel v6)*

EN 12896-3:2016, *Public transport - Reference data model – Part 3 : Timing information and vehicle scheduling (Transmodel v6)*

3 Terms and definitions

3.1 Introduction

This section explains how specific terms are used in the OJP Schema and, where relevant, notes their ancestry from, semantic correspondence with, and any differences from, the use of terminology in other public transport standards. Terminology used in the German TRIAS schema, on which the OJP Schema is based, has been converted as far as possible to the standard terminology used in Transmodel, SIRI, NeTeX or IFOPT.

So where relevant there is a commentary to show that the definition of a term is identical to that used in Transmodel v6 (shown as *[TMv6]*) or Transmodel v5.1 *[TMv5.1]*, or NeTeX *[NeTeX]* or SIRI v2 *[SIRI]* or a commentary how the term differs from that in Transmodel v5.1 or v6, NeTeX or SIRI v2. Terms used in TMv6 or other referenced standards are shown in CAPITALS. Some of the references to NeTeX come from informative rather than normative sections, but are underpinned by the Transmodel v5.1 normative standard.

The dictionary definitions of words in everyday use, such as Journey and Trip, are not sufficiently precise to be used in the specification of an information system – and therefore these (and other) words have very specific definitions (as shown below) when used in the technical sections of this document (sections 3 onwards). A Journey Planning System can have many outputs – in the context of this Technical Specification the focus is on its Trip Planning functionality.

3.2 Explanation of terms used in OJP schema

AbsoluteBearing

absolute compass bearing in degrees

AccessFeatureType

[corresponds to AccessFeatureType of PATH LINK in TMv6]

type of physical feature of PATH LINK

Address *[TMv6]*

descriptive data associated with a PLACE that can be used to describe the unique geographical context of a PLACE for the purposes of identifying it. May be refined as either a ROAD ADDRESS, a POSTAL ADDRESS or both.

Altitude

altitude in metres above sea level

Attribute

generic attribute (with associated text) that can be associated with many facilities, PT service restrictions, etc.

BookingArrangement

[generalisation of BOOKING ARRANGEMENTS in TMv6]

arrangement for booking any TRIPLE(s)

CalcTime

calculation time in milliseconds

CallAtNearStop

indication of the walk distance and time to a nearby stop where relevant

CallAtStop

[same as CALL in SIRI]

the meeting of a VEHICLE JOURNEY with a specific SCHEDULED STOP POINT

CertificateId

form of identification that can be used as a Message Integrity Property (public key cryptography)

Connection

The physical (spatial) possibility for a passenger to change from one public transport vehicle to another to continue a trip, defined by two SCHEDULED STOP POINTs. Different times may be necessary to cover the link between these points, depending on the kind of passenger.

ContinuousLeg

[a specialised type of RIDE in TMv5.1 and NeTEx]

leg of a TRIP that is not bound to a timetable

ContinuousModes

modes that run at any time without a timetable

ContinuousService

[specialisation of SERVICE JOURNEY in TMv6]

vehicle movement on a continuous, non-timetabled service.

DataFrameRef

[corresponds to VERSION FRAME of data in TMv6]

identifier of the set of data being used by an information system, which allows a comparison to be made with the versions of data being used by overlapping systems. Corresponds to VERSION FRAME Id

DatedJourney

[corresponds to DATED VEHICLE JOURNEY in TMv6]

passenger carrying VEHICLE JOURNEY for one specified DAY TYPE for which the pattern of working is in principle defined by a SERVICE JOURNEY PATTERN. DatedJourney details include its operating days.

Delivery

process of delivering messages

DirectionCode

[related to DIRECTION in TMv6]

identifier of a GROUP of DIRECTIONS of the ROUTEs belonging to the same LINE created for the purpose of filtering and organising timetables

Distance

physical measurement of the distance between two specific points in metres

EntitlementProduct

[specialisation of a SERVICE ACCESS RIGHT in NeTEx that may be materialised as a TRAVEL DOCUMENT]

precondition to access a service or to purchase a FARE PRODUCT issued by an organisation that may not be a PT operator (eg: military card, concessionary card, etc)

Equipment *[TMv6]*

item of equipment installed either fixed (PLACE EQUIPMENT) or on-board vehicles (VEHICLE EQUIPMENT). A service (LOCAL SERVICE such as LEFT LUGGAGE, TICKETING SERVICE) is considered as immaterial equipment as well. TYPE OF EQUIPMENT is a classification of EQUIPMENT.

ErrorMessage

message structure that indicates the error state, which could be "OK" if nothing went wrong

ExchangePoints

[specialisation of CONNECTION END in TMv6]

points at which partial solutions from two different journey planning systems may establish a connection

Facility *[TMv6]*

named amenity available to the public at a SITE or on a SERVICE. A FACILITY has no further properties other than a name. An EQUIPMENT or LOCAL SERVICE is used to describe the further properties provided as part of particular FACILITY.

Fare *[corresponds to FARE PRODUCT in TMv5.1 and NeTEx]*

different FARE PRODUCTS that may be available with related information

FareAuthority

[view of the aggregation of ORGANISATION and RESPONSIBILITY ROLE in TMv6]

ORGANISATION which has the RESPONSIBILITY ROLE for the definition of FARE PRODUCTS

FareParam

[corresponds to the FARE parameter model in TMv5.1 and NeTEx]

parameters which are used to determine the price to be paid for a FARE PRODUCT by a specific passenger

FarePassenger

[specialisation of USER PROFILE in TMv5.1 and NeTEx]

attributes of a passenger that influence the price to be paid by that passenger for a FARE PRODUCT

FareProductBooking

[aggregation of DISTRIBUTION CHANNEL and BOOKING ARRANGEMENT in NeTEx]

information to support the process of booking a FARE PRODUCT

FareProductPrice

[corresponds to FARE PRODUCT PRICE in TMv5.1 and NeTEx]

price at which a FARE PRODUCT can be purchased

FareProductValidity *[TMv5.1 and NeTEx]*

conditions of use for a particular FAREPRODUCT

GeoArea

[specialisation of ZONE in TMv6]

LINK SEQUENCE (one-dimensional) forming the boundary of a ZONE

GeoCircle

definition of a circular GeoArea

GeoPositions

[generalisation of LOCATION in TMv6]

Longitude, Latitude in WGS84 (and Altitude if required)

GeoRectangle

definition of a rectangular GeoArea

GeoRestrictions

Application of geographical restrictions to a search

GuidanceAdvice

various types of guidance advice given to travellers

IndividualModes

modes which an individual powers themselves (such as walk, cycle)

IndividualTransportOptions

enumeration of the types and capabilities of Individual Modes

InitialLocationInput

input of a GeoPosition for a starting point

InternationalText

[specialised view of ALTERNATIVE NAME in TMv6]

alternative identified text to be used in specified languages

Journey

[specialisation of JOURNEY in TMv6]

corresponds to a NORMAL DATED VEHICLE JOURNEY or to a DATED VEHICLE JOURNEY but may be extrapolated from a VEHICLE JOURNEY

Latitude

latitude in degrees from equator

LegAlight

situation at a stop or station at which the passenger alights from a TRIPLEG including time-related information

LegAttribute

facilities available on a specified (part of a) TRIPLEG

LegBoard

situation at a stop or station at which the passenger boards a TRIPLEG including time-related information

LegIntermediate

situation at a stop or station that lies between the LegBoard and LegAlight stop or station including time-related information

LegTrack

LINK PROJECTION of a TRIPLEG onto the topography of the route being followed

Length

distance in metres between two points normally along a defined route

Line *[TMv6]*

group of ROUTEs which is generally known to the public by a similar name or number.

LineDirection*[corresponds to DIRECTION in TMv6]*

a GROUP of DIRECTIONS of the ROUTEs belonging to the same LINE created for the purpose of filtering and organising timetables.

LineIdentity *[TMv6]*

number, letter or name that identifies a particular LINE

LinkProjection *[TMv6]*

oriented correspondence from one LINK of a source layer, onto an entity in a target layer: e.g. LINK SEQUENCE, COMPLEX FEATURE, within a defined TYPE OF PROJECTION

Location

spatially-related input (such as name, postcode, coordinate, etc) given by an enquirer and used by the LocationInformation service

LocationInformationService

service establishing the existence in an information system of a location of various types as seen by an enquirer

LocationText

text used by an enquirer of an information system to indicate an origin, destination or other location

Longitude

longitude in degrees from Greenwich

MessageIntegrityProperties

properties that allow checks for message integrity

Mode*[specialisation of MODE in TMv6]*

an extended range of VEHICLE MODEs, aggregating them with some SUBMODEs

MultiPointTrip*[special concept relevant to distributed journey planning]*

Trip that is defined by multiple alternative Origins and/or Destinations

MultiTripFare*[specialisation of FARE PRODUCT in TMv5.1 and NeTEx]*

one or more FARE PRODUCTS covering multiple TRIPs or TRIPLEGs

NoChangeAt

no-change-at restrictions for a TRIP, i.e. SCHEDULED STOP POINTs or STOP PLACES at which no TRANSFER is allowed within a TRIP

NotVia

Not-via restrictions for a TRIP, i.e. SCHEDULED STOP POINTs or STOP PLACES that the TRIP is not allowed to pass through

NumberOfResults

parameter to control the number of TRIP results before/after a point in time. May NOT be used when departure time at origin AND arrival time at destination are set

Occupancy

[corresponds to OCCUPANCY in SIRI]
passenger load status of a VEHICLE

OJPService

IT service within the OJP schema

OJPSubscription

subscription arrangement for a participant in a Distributed Journey Planning system

OpenPercent

percent value as integer, no upper limit

OperatingDay *[TMv6]*

day of public transport operation of which the characteristics are defined in a specific SERVICE CALENDAR and which may last more than 24 hours.

Operator *[TMv6]*

company providing public transport services.

OptimisationMethod

types of algorithm that can be used for planning a TRIP (fastest, least walking, etc)

OsmTag

tag used in the Open Street Map system

Owner

[specialisation of an ORGANISATION in TMv6]
an ORGANISATION with ownership as the RESPONSIBILITY ROLE

Participant

[corresponds to PARTICIPANT in SIRI]
IT system that is participating in a communication with other participant(s)

PassengerCategory

[specialised simplified view of USER PROFILE in TMv5.1 and NeTEx]
classification of passengers by age or other factors that may determine the fare they will need to pay

PathGuidance

[extended view of a NAVIGATION PATH in TMv6 to include the textual navigation instructions]
description of a piece of a TRIP. May include geographic information, turn instructions and accessibility information

PathGuidanceSection

one or more path guidance sections that build the TRIPLEG

PathLink *[specialised view of a PATH LINK in TMv6]*

link within a PLACE or between two PLACES (that is STOP PLACES, ACCESS SPACES or QUAYS, BOARDING POSITIONS, POINTS OF INTEREST etc or PATH JUNCTIONS) that represents a step in a possible route for pedestrians, cyclists or other out-of-vehicle passengers within or between a PLACE.

Percent

percent value as integer, limited to max value 100

PhoneNumber

[specific attribute of CONTACT DETAILS in TMv6]

phone number for a specified organisation, department or purpose

Place *[TMv6]*

geographic PLACE of any type which may be specified as the origin or destination of a trip

PlaceContext

[view of PLACE in TMv6]

PLACE and access to it by individual transport

PlaceData

types of PLACE-related data that can be used as filters

PlaceInformation

Information about a specific PLACE

PlaceType

[corresponds to TYPE OF PLACE in TMv6]

classification of basic types of PLACE

PlaceUsage

use of a PLACE as an origin, destination or via

PointOfInterest

[corresponds to POINT OF INTEREST in TMv6 with related information]

type of PLACE to or through which passengers may wish to navigate as part of their TRIP and which is modelled in detail by journey planners.

PointOfInterestCategory

[view of POINT OF INTEREST CLASSIFICATION in TMv6]

categorisation of POINTs OF INTEREST in respect of the activities undertaken at them

Priority

priority values ranging from 1 (highest) to 5 (lowest)

PrivateCode

code within scope of a private referential system

PrivateMode

[category of MODE in TMv6]

MODEs offered by private individuals

PrivateService

specific service operated by a Private Mode

ProgressBetweenStops

[similar to SIRI]

type for progress between stops

PtMode *[from SIRI]*

mode of public transport service, corresponds to VEHICLE MODE and SUBMODE

PtSubmode

[corresponds to SUBMODE in TMv6]

submode of a public transport mode

RouteDescription

[attribute of ROUTE in TMv6]

descriptive text for a ROUTE, eg: "Airport via City Centre"

Service

Depending on context the word service can refer to a PT service (being an overall package of one or more public transport routes, timetables and travel facilities offered); or an IT service; or a journey planning, information or other functional service; or an immaterial facility such as Left Luggage. The context is often clear when it is used with another term (as in PT service, special service, service link, service pattern, etc).

ServiceDelivery *[IT service, taken from SIRI]*

response from producer to consumer to deliver payload data. Either answers a direct ServiceRequest or asynchronously satisfies a subscription. May be sent directly in one step, or fetched in response to a DataSupplyRequest

ServiceFacility

[view of a SERVICE FACILITY SET in TMv6]

(a view of) SERVICE FACILITY SET (a set of FACILITIES available for a specific VEHICLE TYPE (e.g. carriage equipped with low floor) possibly only for a service (or for a SERVICE JOURNEY or a JOURNEY)

ServiceRequest *[an IT service, taken from SIRI]*

a request for the immediate delivery of data. Answered with a ServiceDelivery (or a DataReadyRequest)

ServiceResponse *[an IT service, taken from SIRI]*

response message containing a payload and various attributes of its own

ServiceStatus

parameters which describe the current status of a DATED VEHICLE JOURNEY

ServiceViaPoint

[specialisation of VIA in TMv6]

STOP PLACE or SCHEDULED STOP POINT as a VIA for a particular SERVICE PATTERN. Specialisation of a VIA.

SharingModel

sharing service loan and return scheme

SharingService

transport service that provides shared vehicles

Signature

data for transmission of message signatures (public key cryptography), used to prove Message Integrity

SituationFullRef

reference to situation message. Message details might be found in response context or through other communication channels

Situations *[as SIRI]*

container for the structured description of situations in public transport or on the road (individual transport)

Speed

relative speed in percent. If given slows the standard speed (below 100) or increases it (above 100)

StaticFare

[specialisation of a FARE PRODUCT in TMv5.1 and NeTEx]

list of FARE PRODUCTS under the responsibility of a FARE AUTHORITY, independent of any specific trip result

Status

whether the request was processed successfully or not. Default is "true"

StopAttributes

[properties of a SCHEDULED STOP POINT or STOP PLACE in TMv6]

selection of attributes of ACCESSIBILITY LIMITATION related to a SCHEDULED STOP POINT extended by attributes referring to some PLACE EQUIPMENT

StopCallStatus *[cf SIRI CALL]*

current status of each stop along the route of a PT Service

StopEvent *[cf SIRI CALL]*

departure or arrival event or both

StopEventContent *[cf SIRI CALL]*

parameters which define the extent of information to be returned in a Stop Event result

StopEventData *[cf SIRI CALL]*

parameters which establish various filtering criteria for Stop Event results

StopEventParam *[cf SIRI CALL]*

parameters which define what is to be included in a Stop Event result

StopEventPolicy *[cf SIRI CALL]*

parameters which define the number, Time Window and types of Stop Event to be included

StopFacility

[corresponds to a SITE FACILITY SET in TMv6]

FACILITY that applies to a stop

StopFare

stop-related FARE PRODUCT information

StopPlace

[extended view of STOP PLACE in TMv6]

a STOP PLACE extended by ACCESSIBILITY LIMITATION properties and of the associated equipment, comprising one or more locations where vehicles may stop and where passengers may board or leave vehicles or prepare their trip, and which will usually have one or more wellknown names

StopPoint

[extended view of SCHEDULED STOP POINT in TMv6]

a SCHEDULED STOP POINT extended by ACCESSIBILITY LIMITATION attributes and with identifier and name where passengers can board or alight from vehicles

SubMode *[TMv6]*

variant of a MODE, as for instance international or domestic rail (rail being the MODE)

Subscription *[cf SIRI]*

arrangement which allows a specific organisation to access an information system - usually taking the form of security parameters which may or may not need to be paid for

SubscriptionRequest *[cf SIRI]*

request from a subscriber to a producer for a subscription to a particular IT service. Answered with a SubscriptionResponse

TariffZone

[corresponds to TARIFF ZONE in TMv5.1 and NeTEx]

A ZONE used to define a zonal fare structure in a zone-counting or zone-matrix system.

TerminateSubscription *[cf SIRI]*

request to end a subscription to a particular IT service

Ticket

[corresponds to a FARE PRODUCT in TMv5.1 and NeTEx]

a FARE PRODUCT which may be materialised as a TRAVEL DOCUMENT and which gives the passenger an entitlement to travel on one or more specific TRIPLEGs subject to specified conditions

TimedLeg

[corresponds to a RIDE in TMv6 (with related information)]

passenger TRIPLEG with timetabled schedule.

TimeWindow

the window of opportunity that the traveller has to perform a specific TRIPLEG

TopographicPlace *[TMv6]*

A type of PLACE providing the topographical context when searching for or presenting travel information, for example as the origin or destination of a trip. It may be of any size (e.g. County, City, Town, Village) and of different specificity (e.g. Greater London, London, West End, Westminster, St James's).

TrackSection

LINK PROJECTION on the infrastructure network of the TRIPLEG together with time information

Transfer

travel between two POINTs located sufficiently near to each other that they represent for a passenger a possibility to reach one of the POINTs when starting at the other in a timescale which is realistic when undertaking a TRIP, eg: ACCESS or CONNECTION

TransferDuration

[attribute of a CONNECTION (not INTERCHANGE) in TMv6]

calculated duration in a response taking into account the request parameters.; TransferDuration plus waiting time is the minimum interval between arrival and departure time.

TransferLeg*[specialisation of NAVIGATION PATH in TMv6]*

description of a TRIPLEG which links other TRIPLEGs where a TRANSFER between different LOCATIONS is required

TransferModes

MODEs dedicated to perform TRANSFERs

Transition

transition types for interchanges

Trip*[extended view of TRIP PATTERN in TMv5.1 (to include time-related information)]*

whole journey from passenger origin to passenger destination in one or more TRIPLEGs

TripContent

parameters that control the level of detail of the TRIP results

TripData

data to be included/excluded from search, eg: modes, operators, etc.

TripFare

Fare for a specified TRIP

TripFareProduct*[specialisation of FARE PRODUCT in TMv5.1 and NeTEx]*

FAREPRODUCT covering a part or the whole of a TRIP from boarding the first public transport vehicle to alighting from the last public transport vehicle (corresponds to a package of PREASSIGNED FARE PRODUCTS)

TripInfoService

short for Trip Information Service. OJP Service that provides details on a single VEHICLE JOURNEY that may be used by a passenger TRIP

TripLeg*[generalisation of RIDE in TMv5.1 covering more MODEs]*

a single stage of a TRIP that is made without change of MODE or VEHICLE JOURNEY (ie: between each TRANSFER)

TripLocation

location of a passenger currently travelling in a VEHICLE

TripMobility

parameters the user can set to restrict the mobility options, particularly for interchanging

TripParam

parameters which define the content expected in a TRIP plan

TripPolicy

parameters which control the TRIP search behaviour

TripSummary

summary of the details of a TRIP

TripVia

VIA restrictions for a TRIP

TurnAction

the range of alternative turns that can be described

TypeOfFareClass

classes of travel available on a particular PT service which will affect the price to be paid

ValidDayBitType

sequence of bits (values 0 or 1) coded as a text string and representing days

VatRate

the relevant Value Added Tax rate

Vehicle

[a specialisation of VEHICLE in TMv6 to include privately operated vehicles as well as public transport ones]

specific type of vehicle used for carrying passengers

VehiclePosition

geographical and logical position of a vehicle

VehicleProgress

progress of a vehicle relative to timetable service pattern

Version *[TMv6]*

a group of operational data instances which share the same VALIDITY CONDITIONS. A version belongs to a unique VERSION FRAME and is characterised by a unique TYPE OF VERSION

WebLink

URL of a web resource with label

Weekday

[derived from DAY OF WEEK and PROPERTY OF DAY in TMv6]

enumeration of individual the seven DAYS OF WEEK, along with public holidays

WeekdayTimePeriod

[specialisation of TIME BAND in TMv6]

aggregation of TIME BAND, DAY OF WEEK and PROPERTY OF DAY

Zone *[TMv6]*

A two-dimensional PLACE within the service area of a public transport operator (administrative zone, TARIFF ZONE, ACCESS ZONE, etc.).

3.3 Explanation of extension terms in OJP Schema

The following are extensions that can be associated with other Dictionary entries

Base

basic or default requirement

Code

shorthand reference for a particular item of data

Enumeration

set of values that can be associated with the term

Filter

parameter that controls the level of detail of a particular type of information

FilterGroup

collection of parameters that control the level of detail of a particular type of information

FilterStructure

structure of a set of filter parameter(s)

Group

group of elements related to a particular topic

Input

requirement for specific type of information to be supplied for the system to work

InputStructure

structure of the way an INPUT is to be expressed

Ref

code that refers to something specific, normally of a specified type (such as VEHICLE JOURNEY, STOP POINT, DAY TYPE, etc)

RefGroup

group of REFs related to a specific topic

ReferenceStructure

structure of the way that a REF code is expressed in a particular context

RefStructure

same as "Reference Structure"

Request

structured message requesting specific information

RequestStructure

structure of a Request for a particular type of information

Response

structured message providing specific information that has been requested

ResponseStructure

structure of a Response for a particular type of information

Result

partial or complete outcome from a data search or calculation including status information

ResultStructure

structure of a Result from an enquiry

4 Symbols and abbreviations

API	Application Programming Interface
DJP	Distributed Journey Planning
DS	Distributing System
EHS	Enquirer's Home System
EU	European Union
HTTP	HyperText Transfer Protocol
IFOPT	Identification of Fixed Objects in Public Transport
IP-KOM	I nternet- P rotokoll basierte K ommunikationsdienste (IP-based communications services)
ISO	International Standards Organisation
IT	Information Technology
NeTEx	Network and Timetable Exchange
OJP	Open Journey Planner
Open API	An API which is published for widespread implementation
POI	Point of Interest
PT	Public Transport
REST	Representational State Transfer
RS	Responding System
SIRI	Service Interface for Real-time Information
SOAP	Simple Object Access Protocol
TM	Transmodel
TPEG	Transport Protocol Experts Group
TRIAS	Travellers' Realtime Information and Advisory Standard
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Universal Resource Locator
VDV	Verband Deutscher Verkehrsunternehmen (D)

WGS	World Geodetic Standard
WSDL	Web Services Description Language
XML	eXtensible Mark-up Language
XSD	XML Schema Document

5 Use cases

5.1 General

The following use cases describe the requirements for the standardised OJP schema that is being proposed for Distributed Journey Planning (DJP). It covers both key tasks that are essential for the core functions in a DJP system and some additional tasks that might usefully complement such systems. It is suggested that all implementations should cover the essential tasks – and each should also describe in a discovery mechanism those other functions which that particular implementation is able to handle.

The following descriptions assume a relatively simple and straightforward example of Distributed Journey Planning in which there are four key "actors" or "roles" in the process :

- 1) **enquirer** – the person asking for information
- 2) the enquirer's **home system** – the journey planning system to which the enquirer is connected
- 3) a **distributing system** - the system that distributes journey planning enquiries to other systems
- 4) **responding system(s)** – the system(s) that respond to questions from the distributing system.

An enquirer requests a trip from a journey planning system which is referred to as the enquirer's **home system** – and the home system then has to recognise that the enquiry requires information from other systems. It therefore passes the enquiry to a **distributing system** (which may be integrated with the home system or physically separate) which makes enquiries of one or more **responding system(s)** from which it will glean additional information in order to put together a comprehensive trip plan to meet the enquirer's request. Once the **distributing system** has the required comprehensive trip plan it returns this to the home system, which in turn presents the results to the enquirer.

The **home system** will have a user interface through which it receives enquiries from enquirers, and through which it responds to those enquirers. It may have its own journey planning capability for a specific geographical territory (which could also cover only certain modes of transport) – or it may only have sufficient intelligence to know that it needs to draw information from other journey planning systems by passing the enquiry to the distributing system.

The **distributing system** will need to know from which journey planning systems it needs to draw all relevant information to meet the requirements of each enquiry it receives, and it needs to be able to communicate with the relevant responding systems – which it would do using an API.

To describe the geographical context of a distributed journey planning system there are three key concepts

- 1) the **local region** – the territory for which the journey planner itself can plan trips without information from other systems
- 2) the **adjacent regions** – the regions which are adjacent to the local region and have their own "local" journey planning systems
- 3) **remote regions** – the regions which have their own "local" journey planning systems but which are not adjacent to the local region

There are five typical distributed journey planning scenarios (in addition to the non-distributed scenario of a local point to a local point), each of which may require handling in a different way depending on the architecture of the distributed journey planning system:

- 1) local point to remote point.
- 2) local point to adjacent region point.
- 3) remote point to remote point (same region).
- 4) remote point to remote point (different regions)
- 5) remote point to remote point (different but adjacent regions)

A point in the above descriptions refers to any location which can be translated into geographical coordinates – this might be, for example, the centre of a topographic place (city, town, village), the centre of a postcode, or a specific address or point of interest. Or it can also refer to a public transport stop which might be translated into either its unambiguous Reference (code) or its geographical coordinates.

In some distributed journey planning systems there may be one or more additional types of responding system, particularly where there is a separate journey planner providing information for long-distance travel (such as a rail and/or coach journey planner, or ones covering flights and/or ferries, or ones covering car journey planning as an access option to and from public transport). Such additional planners may have a larger territorial area than the regional planners that cover local transport.

As noted earlier, this Technical Specification is concerned with the communication protocols necessary to pass enquiries and responses between participating DJP systems. It is not concerned with the architecture or logic of the journey planning engine that sits in any distributing system, through which the user's enquiry is broken down into appropriate components. Each component is then forwarded to a relevant local journey planning system with DJP capabilities (a **responding system**) in order to collect information about each component of the requested trip plan.

5.2 Key tasks for Distributed Journey Planning

5.2.1 Planning a component of a trip

An enquirer wishes to plan a trip and places the enquiry with a journey planning system that has DJP capabilities (the enquirer's **home system**). Part of the requested trip lies in an area covered by a different journey planning system which also has DJP capabilities. The initiating **home system** (the one which received the end-user's enquiry) passes the journey planning details to a **distributing system** which creates a journey planning request to pass to the **responding system** for the component of a trip in the responding system's territory. The request comprises one or more origin points, one or more destination points, a preferred departure or arrival time at each of either all of the origin or all of the destination points, and any preference that the enquirer may have stipulated for modes (and sub-

modes) of transport to be included in or excluded from the trip plan, along with any specific accessibility / mobility requirements. Maximum walk distances and estimated walk speeds should be included, with a separate connection walk speed that may be different. It may also include a via (or not via) point, or a point at which a transfer may or may not take place, and it could specify preferred operator(s) of PT services, or specific PT services. It could include the journey planning algorithm that is preferred (such as fastest, least walking or least connections). And it could ask for the response to just give the essential details, or it could ask for all stops to be shown, with scheduled times or currently projected times from a real-time system. If cycling is offered as an option to reach the start of a public transport trip (or to complete the trip) then this would need to be requested and parameters would specify maximum cycle distance and expected cycling speed. Finally the number of trip options to be returned needs to be specified – in terms of a number of trip options, or a time period to be covered by the trip options offered, and whether trips are required from each requested origin or just any one or more of them, and likewise to each requested destination or only to any one or more of them.

5.2.2 Discovering relevant stops

In order to specify a DJP request it will be necessary for the **distributing system** to determine the set of stops from one of which the requested component of the trip has to start – and likewise the set of stops at one of which it has to end. This set of stops are those to or from which the **distributing system** is able from its own resources (or from responses to other component enquiries) to plan the requested overall trip. There may be a need, therefore, to establish the identity of potential connection stops on the border between journey planning systems. This would require an exchange of questions and answers, delivering the name of the stop, its coordinates and the code by which it is recognised by the relevant **responding systems**.

5.2.3 Obtaining information about accessibility and services for those with special needs

Travellers with mobility, sensory or other disabilities may have special needs when using public transport. The user making an enquiry will need to be able to describe their own needs in a simple and codified way that can be interpreted intelligently by journey planners, so that solutions offered best meet the needs of the enquirer. The schema for DJP systems therefore has to be able to convey the specific needs of such travellers, whilst the responses have to be able to carry the information necessary to describe the accessibility or other features of the PT services being proposed in the trip plan in sufficient detail that the traveller can be confident of what will be offered when making the trip. For this to be matched to available information about accessibility restrictions it would be necessary to return information about the leg path – so that information about access paths and connection paths is linked to the physical restrictions that apply to them (ramps, escalators, lifts, stairs, etc).

5.2.4 Seeking route information that can be displayed on maps

A trip plan will comprise several distinct components. At a minimum it will normally involve an access leg to get to the public transport system, a public transport leg and a final access leg to get from the public transport system to the required destination of the passenger's trip. Whilst the passenger should be able to rely on the public transport vehicle taking them from the boarding point to the alighting point, many passengers for one reason or another like to follow their route on a map. But the requirement for a map is much greater for the access legs at each end of a trip – and for any transfers necessary in the course of the trip. To that end it should be possible to request sufficient details of the track (sequenced map coordinates) of the trip to be made such that it can be plotted over a topographic map of the trip at various scales or zoom levels, sufficient to see detail where it is most needed but also to be able to see the whole trip in context.

5.3 Other possible tasks for a Distributed Journey Planning system

5.3.1 Requesting a stop timetable

Whilst a traveller may wish to plan a specific trip, it may equally be useful to be able to see a list of all departures from a specific stop within a specified time window – a "stop timetable". A request for such a timetable might specify not only the stop but possibly an individual platform within such a stop.

5.3.2 Requesting times for all intermediate stops in a trip

Whilst a trip may be specified by simply its boarding and alighting stops and times, it is also possible to ask for trips to be described with details of all stops between the boarding and alighting points and the times at which these stops are expected to be reached. This would help a passenger follow the progress of their trip along the route.

5.3.3 Requesting expected events at a particular stop

With an increasing number of operations having real-time monitoring and prediction systems the departure times of vehicle journeys can be obtained not only in the form of a timetable, but also in the form of predicted departure times calculated in real time. So whilst this information is not available when pre-planning a trip a long time in advance, it becomes available close to the scheduled departure times (and becomes increasingly accurate as the scheduled time gets ever closer). A "stop event" request and response should deliver the best information known at the current time for departures from a stop in the next short while – perhaps 30mins or an hour. Whilst this information may be available locally, the DJP system provides an opportunity for such information to be available in the user's native language and in a style of presentation with which they are familiar if they make the request through the initiating system rather than seeking the information from a local information system.

5.3.4 Requesting information about the fares and ticket options for a particular trip

One of the most challenging aspects of public transport information is that related to fares and tickets, with significant differences in the way in which fares are determined, in the nature of the types of ticket that are available, and the way in which tickets can be purchased. As journey planning systems increasingly provide at least basic information about fares and tickets so a DJP system should be able to interrogate such systems and convey the relevant information to enquirers. The first priority for detailed information would be to give the price for a specific planned trip on the assumption that it is paid for with single trip tickets. However such information places public transport at a disadvantage as many public transport systems have much cheaper ticket options for return tickets, multi-trip tickets, multi-person tickets, etc. So a second priority would be to provide more general information about ticketing systems in a specific geographical area. In addition there may be a requirement to provide hyperlinks to on-line retailers of relevant tickets.

5.3.5 Other possible questions

Whilst the Open API will cover all normal forms of journey planning request, it is recognised that some journey planning systems may wish to ask for supplementary information if it is available. Examples of such requests might be to seek a "rain safe" trip (one which minimises exposure to the weather), or to ask for an estimate of energy usage for each leg of a trip. Such requests could be added to the schema on a bi-partite basis between systems that can exchange such additional information, without undermining the generality of the Open API specification.

6 System Architectures, Metadata and Data

This technical specification does not seek to specify how a distributed journey planning system should work – the specification is agnostic on this question as experience has shown that each of the methods used to date has required a very similar question-and-answer protocol, which in essence is what the API provides. However the experience of the three main distributed systems to date (and that being developed in France) offers some indications of the alternatives that can be considered – and further details of the three historic systems (EU-Spirit, DELFI and JourneyWeb) and the emerging French system (APII-SIM) are given in Annex B.

6.1 General considerations

The core of Distributed Journey Planning lies in the fact that several journey planning systems have to contribute to fulfil a complex (or very long-distance) journey planning task. This task is split into smaller sub-tasks by the **distributing system**. The sub-tasks are then sent to **responding systems** that take care of them.

This approach has some working assumptions:

- 1) The **distributing system** is able to split the original task into sensible sub-tasks.
- 2) The **distributing system** knows from which **responding systems** reasonable solutions for a sub-task can be expected.
- 3) The **distributing system** knows the exchange points (and their identifiers) that will link the parts of the overall trip (the solutions of each **responding system** for a sub-task). In most cases this will comprise the stop identifiers of relevant stations, airports, ferry ports or coach hubs.

Points 1 and 2 above emphasise the necessity of knowing the coverage of each **responding system** in terms of area and transport services. Splitting a long-distance (maybe cross-continent) trip request into pieces involves identifying parts of the overall trip that can be covered by **responding systems** that have incorporated timetable data for certain areas or transport services. This knowledge about responding systems' capabilities can be kept in static files or in an online register (see 8.11).

Probably the most crucial difference in the architectures that have been used for distributed systems to date has been the way in which relevant data is held, and where the various stages of computation take place.

For a participating system there are two questions to be asked for each enquiry

- Can this enquiry be responded to using only information already in this system? If so, then distributed planning is not required.
- If a distributed journey plan is required, which other journey planning systems can provide the components that might make up the journey plan solution.

If a distributed journey plan is required then the architecture of the distributed system will determine how the calculation is made. Some of the questions and options are :

- Is it possible for the **home system** to include long-distance public transport information to the exchange points in all other regions? – if so the journey planning calculations are simpler and the risk of sub-optimal solutions being offered is minimised

- To what extent can each of the participating systems include data for PT services which traverse its own region's immediate regional boundary? This would help to ensure that such trips can be planned on a single system (rather than distributed) – avoiding the undoubtedly difficult situation where services repeatedly cross the boundary between adjacent journey planning systems
- Can the distributed journey be planned on the Enquirer's **home system**, which itself draws information from one or more other systems to cover the long-distance leg (if not held in the user's home system) and the distant local leg of the trip solution (peer-to-peer arrangement)?
- Is a distributed enquiry passed to a central **distributing system** which then draws information from both the user's home system and from other relevant **responding systems** before passing the solutions back to the home system? The central **distributing system** might itself host a multi-modal journey planner covering the long-distance PT services required to join the respective exchange points in each participating region (centralised arrangement).
- A central system might start by planning the core long-distance element of the trip first, and then add the feeder legs into and from that long-distance trip ... or it might try to plan the trip sequentially from origin to destination
- What is the most efficient sequence of planning the trip? If long-distance data is held within the user's home and distant **responding systems**, then it is likely that one end of the trip has fewer potential exchange points than the other – so allowing the end with the larger number of exchange points to plan its own local leg of the trip and the long-distance leg. This is less complex and therefore more efficient than the other way around.
- Whatever planning method is used both forward and backward planning will be necessary to ensure that the solutions are optimised, by ensuring that the best trips are offered
- No matter how the journey planning systems are separated in terms of geographical or PT service scope, there is always an option to co-locate different systems (or copies of different systems) if this would provide advantages in terms of speed of inter-communication between the distributed system partners, as they could be linked across a very high-speed Local Area Network. This then maintains the separate management of each local journey planning system whilst avoiding the latency involved in communication between servers over the internet.

6.2 Metadata requirements

The journey planning methods noted above imply a need to have an understanding of the overall network of public transport services, as well as the geographic scope of each participating journey planner (the **responding systems**). This requires several sets of metadata which might include :

- Gazetteers which allow the journey planning systems to determine the journey planning system areas in which the origin and destination respectively are situated
- Lists of potential exchange points that might be passed through in any distributed journey plan – typically these represent locations on the long-distance public transport network at which the long-distance legs of trips might start or finish ... so the interface between the "trunk" legs and the "local" legs at each end of the overall trip.

Identifying the exchange points (and their identifiers) that can be used for joining the parts of the overall trip delivered by each of the **responding systems** could be done in one of the two following ways:

- by static files that list the exchange points for each responding system (or each pair of responding systems)
- by an online-request for exchange points that can be sent to each responding system.

6.3 Core data requirements

Experience has shown that journey planners generally need to be able to identify a location for the start and end of a trip from a user's input of

- Any stop, station or other public transport terminal
- A topographic place (city, suburb, town, village or hamlet)
- A wide range of different types of points of interest
- A named street
- The postal street addresses of an individual property
- A postcode (particularly relevant where these are very precise, as in the UK where a postcode typically covers no more than about 50 addresses on a specified street)
- A point on a map.

There are two options how to lookup these types of locations (see also Figure 1). The enquirer's **home system** could collect all topographic place data from all **responding systems** (if no such central topographic place database already exists) and harmonise them into a local copy of a system-wide topographic place gazetteer. This typically involves an off-line process of collecting the data and making all entries unique. The other option is to look up relevant matches from the **responding systems** at the time the user request is received. This latter option does not need an off-line pre-processing step, but makes dealing with the request at run-time much more complex. In any case it is crucial that the journey planning requests are sent to the **responding systems** with object identifiers for the origin and destination location (and all exchange points) that the **responding systems** are able to interpret correctly.

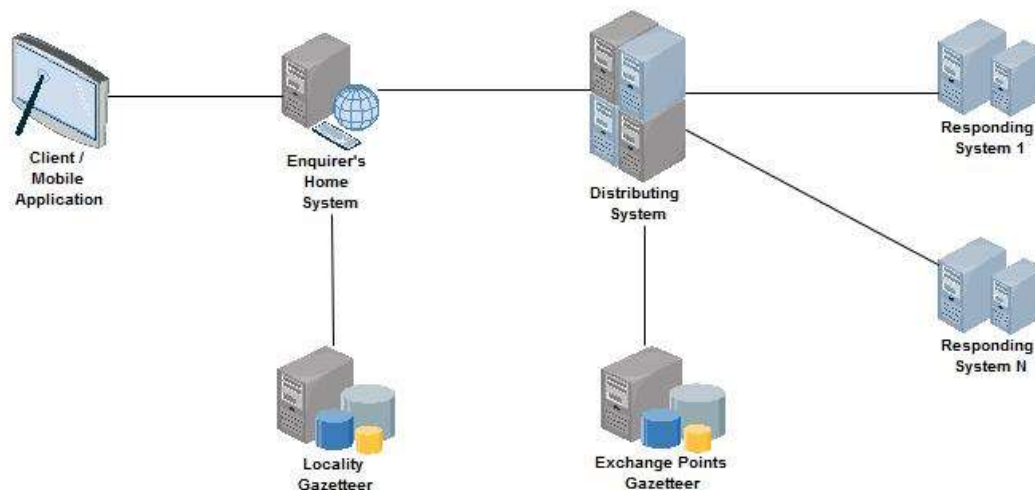


Figure 1 System Architecture of a Distributed Journey Planning environment.

*If a system-wide Topographic place Gazetteer does not exist, the Enquirer's **home system** has to retrieve distant localities (and their object identifiers) from the **responding systems** (via the **distributing system**) at run-time.*

For all these types of locations to be usable each needs to be associated with a precise geo-location in the form of coordinates following a recognised standard (including those which can be converted to a different standard coordinate system if necessary). For locations which are "areas" (such as cities, towns, villages or large points of interest such as a hospital or a park) then it is normal that an arbitrary central point is taken as a proxy for the overall location, although it may also be possible to represent large points of interest by a set of points (in effect "entrances" to the relevant area) each of which could be used as an alternative location for the POI.

The journey planner will also need a topographic map which allows the routing of trips to follow appropriate roads, alongside the ability to plan walk legs along all available pedestrian routes, and the ability to present the origin, destination, interchanges and stops, and routes. In some locations it may be necessary to prevent the use of certain paths or roads by a routing engine (eg: not allowing pedestrians to walk along motorways or through road tunnels). If the journey planning engine finds interchange opportunities by proximity rather than by such opportunities being identified in the stops data, then mechanisms may be needed to add barriers to prevent interchanges connecting stops that have no physical connections (eg: on two sides of a river which has no bridge).

The schedules of all public transport services need to be available – and should be encoded to a unique representation of each physical stopping point (matching the gazetteer entry location for such points).

Consistent rules are required to ensure that adequate time is allowed for connections between PT services – taking into account the time that may be required for ticket purchase, and movement between the respective stopping points, as well as taking account of potential routine perturbation of operations (for instance allowing up to, say, 3 minutes for the possible late running of a service from which a connection is to be made to an otherwise on time onward leg of the trip). For airports and ferry ports the connection times also need to allow for security and baggage handling arrangements – and for these there are differences between those travelling with or without luggage and between those who like to have generous time allowances at such locations, and those who are happy with tightly timed connections.

Data ideally should include accessibility information associated with vehicle journeys, physical stops and transfer locations, such that the planner can advise whether trips can be made by wheelchair users, or those with mobility problems that may require lifts rather than stairs or escalators.

Booking information should be included for PT services which require booking in advance (either because they have reserved seats, such as a long-distance coach service, or because they are demand-responsive services). In general this would be expected to be a hyperlink to a third-party site on which reservations can be made.

Arrangements for accessing fare information may also be included. Requirements for this differ widely from country to country, and between different types of PT service. Where information can be found on-line or where ticket purchase is possible on-line then a hyperlink to such sources or services should be included.

In France (and maybe elsewhere) an option for car-pooling is envisaged, where a car driver offers to provide transport to one or more passengers to share his or her car.

7 Open API for Distributed Journey Planning – OJP Services

7.1 Departure Monitor

7.1.1 Purpose

The departure monitor service shows the departures from a certain stop / station. Real-time information systems can also integrate the available forecast data and the information about interruptions. The OJP API is oriented towards end-users of the information, and can look backwards as well as predicting forwards in time. This is in contrast to SIRI, which is oriented towards communication between information systems and focuses only on observations and predictions of vehicle positions in current time,

The arrival monitor variant informs of the arrival times of the various modes of transport at a stop / station.

If, for instance, a coordinate has been chosen as the point of departure instead of a stop / station, the departures (arrivals) at the stops / stations within a specified radius are shown (perhaps with information about the distance to the stop).

The departure monitor service can also be used to provide the data for the operation of a digital passenger information monitor.

7.1.2 Interactions

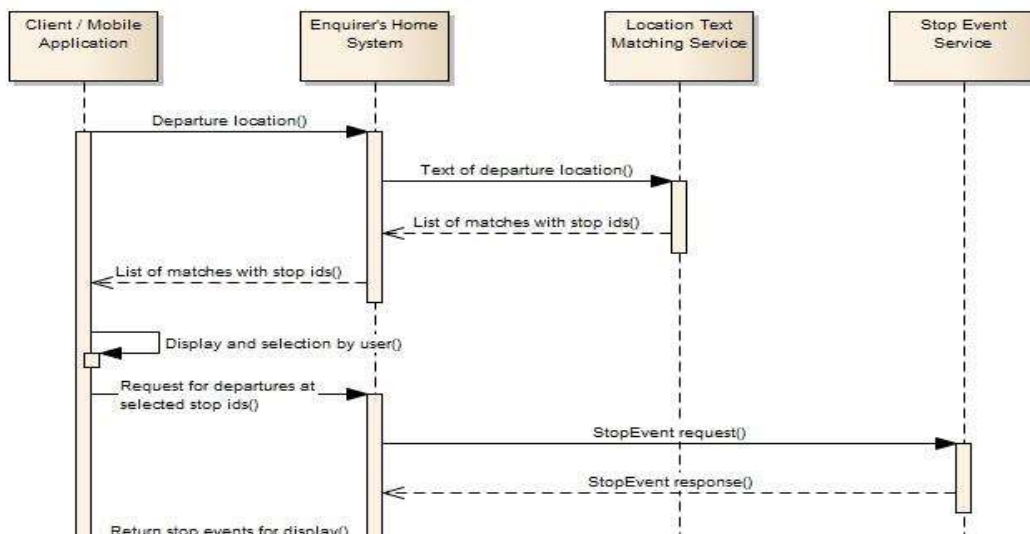


Figure 2 — Sequence diagram for a departure monitor

7.1.3 Concerned Components

Service provider: Responding System

Service requester: Home System

7.1.4 Function 1: Departure Monitor

Input data

- Departure location (as coordinate, stop / station, stop area, address, POI)
- Date and time
- Departure or arrival variant (or both)
- User options:
 - need for low-floor vehicles, wheelchair transport
 - transport mode filter
 - line filter
 - direction filter
 - possibility of transporting bicycles
 - fare restrictions (e.g. no lines with supplementary fare)
- Technical parameters, like number of desired events, time slot
- Degree of desired additional information (with mode of transport before and after the stop / station etc.)

- Use of forecast and interruption data (including SIRI SX or SM messages, or information about time restrictions for a stop point)

Output data

- Departure / arrival events, including
 - journey ID
 - departure stop / station
 - target time and forecast or information about journey interruption
 - line number
 - mode of transport
 - direction
 - stops / stations and forecast arrival and departure times up to the approaching stop / station
 - stops / stations and forecast arrival and departure times after the latest stop / station
 - transport company
 - references
 - URL of booking systems

7.2 Fare Information

7.2.1 Purpose

Distinction is made between two kinds of fare information, i.e. static and trip-related fare information. The static information includes e.g. lists of fare offers, assignments of tariff zones to stops / stations and URLs to further fare information. Trip-related information includes e.g. the various fares valid for a certain trip (or a section of a trip) on a certain day. The fare offer for a trip can comprise the cash fare (fare for a single trip, fare for multiple trips, fare for children etc.) and the time-related fare (fare for a day, a week, a month or a year, fare for students for a semester).

7.2.2 Interactions

Two flows are imaginable for the trip-related fare information, which are seen in the below figures.

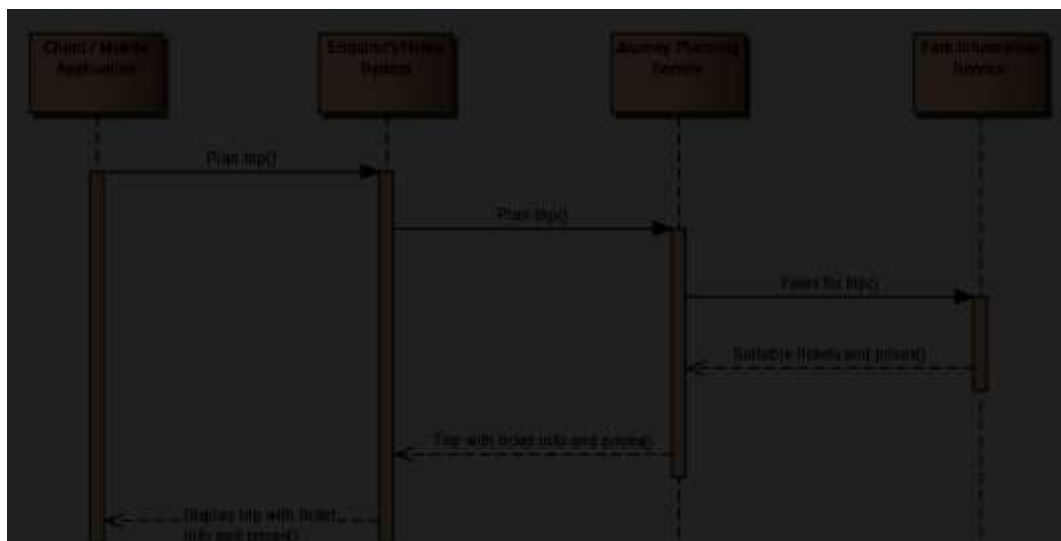


Figure 3 — Sequence diagram for a flow integrated into the calculation of the trip

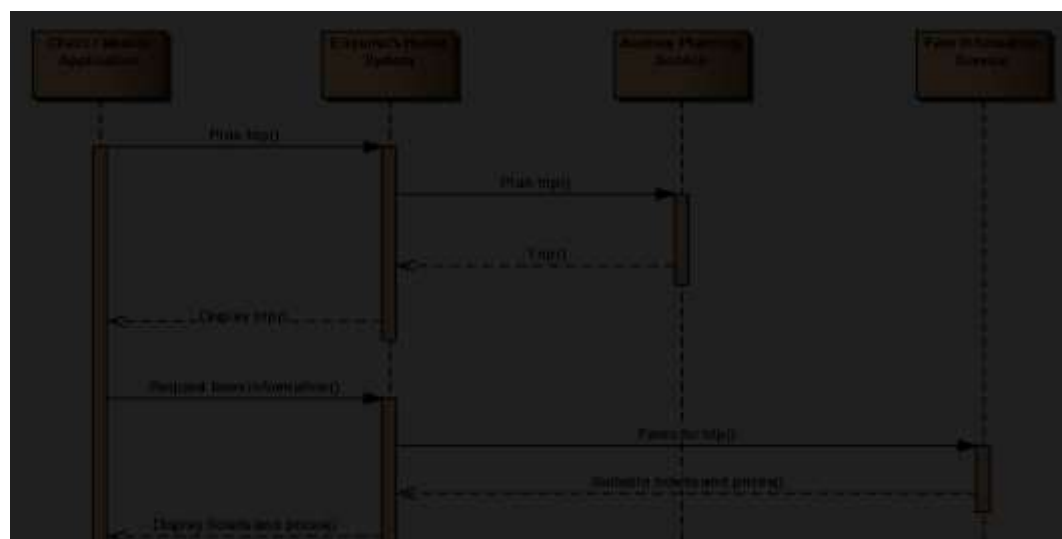


Figure 4 — Sequence diagram for a flow after calculation of a trip

7.2.3 Concerned Components

Service provider: fare calculator (Responding System)

Service requester: Home System

7.2.4 Function 1: Tariff Zones for Stop / Station

Input data

— Stop ID

Output data

— List of tariff zones (if assigned to the stop / station)

7.2.5 Function 2: Static Fare Information

Input data

- Fare authority code (e.g. “VVS”)
- Key date
- Optional: passenger category (e.g. “adults”, “children”, “seniors”)

Output data

- List of fare offers
- URLs to further information

7.2.6 Function 3: Trip-Related Fare Information

Input data

- Specific trip inclusive of all stops / stations passed, lines, timetable times, date

Output data

- List of tariff zones passed (if the area is divided into tariff zones)
- List of fare offers with indication of the validity for a trip (e.g. “valid for the complete trip”)
- URLs to further information.

7.3 Location text matching

7.3.1 Purpose

Usually, reference is made to local geographical objects via clear alphanumerical object IDs within a Responding System and its services. However, often the passenger uses the name of the local objects (name of stop / station, postal address, POIs).

Location text matching determines which objects were probably meant by comparing the register for local geographical objects with the user’s textual input. Object types (e.g. stop / station, address), geographical restrictions or other object features can be specified in the query.

7.3.2 Interactions

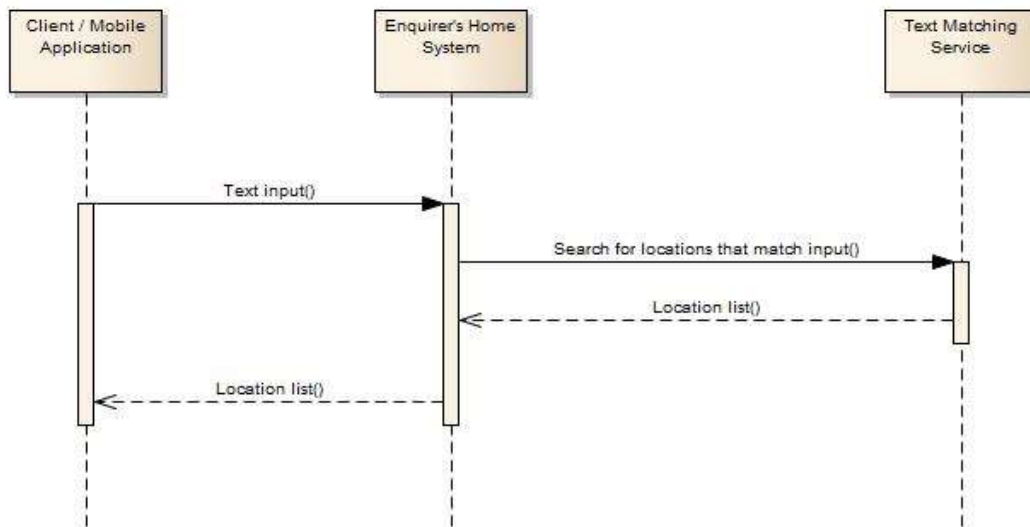


Figure 5 — Sequence diagram for a location text matching

7.3.3 Concerned Components

Service provider: Geo-name register (Responding System)

Service requester: Home System

7.3.4 Function: Location text matching

Input data

- Textual input
- Object types allowed
- Geographical restriction / weighting
- Further object restrictions (attributes)

Output data

- List of local objects (name, geographical position, attributes) with probabilities per local object

7.4 Object Information Service

7.4.1 Purpose

A Home System or a Responding System has to be able to make the geographical objects of several Responding Systems available to a user in a transparent way. Therefore, a Responding System has to be able to transmit all its geographical objects upon a query. The transmission of the geographical objects should follow after as few queries as possible; ideally, after only one.

Thus, a Responding System can e.g. “learn” the names and object designations of all objects of another Responding System by way of such a “group query” so that it can call on them for comparison with the user’s textual inputs.

The object information service outputs all local geographical objects of a Responding System, but perhaps they are restricted according to object types, special features or geographically.

7.4.2 Interactions

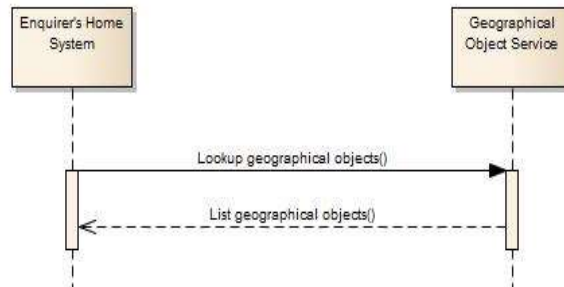


Figure 6 — Sequence diagram for an object information service

7.4.3 Concerned Components

Service provider: geo-name register (Responding System)

Service requester: Home System, Responding System

7.4.4 Function 1: Object Information

Input data

- Object types allowed
- Geographical restriction
- Further object restrictions (attributes)

Output data

- List of local objects (name, geographical position, attributes).

7.4.5 Function 2: Finding relevant exchange points

When a long-distance trip is requested that exceeds the capabilities of each local planner the Distributing System needs to split the request for the total trip into smaller pieces that can be processed by local planners. The splices are at the exchange points which are known by neighbouring systems.

Therefore a Distributing System has to know about the exchange points between relevant journey planning systems before it can start to request for the partial trips.

Input data:

- Reference to the adjacent system(s) between which the exchange points are to be retrieved (optional)

Output data:

- List of all exchange points this Responding System has knowledge of (maybe filtered by some adjacent systems specified in the request), their identifiers and names.

8 Open API for Distributed Journey Planning – Interface Description

8.1 Notation of XML-Elements and XML-Structures

The OJP (Open Journey Planner) interfaces presented in this document are defined using XML schemas. Thus, the objects that are exchanged over the interface exist as XML elements. The description of XML elements are displayed in tabular form in the current document, which originated from SIRI (CEN, EN 15531 Part 3). It is very compact and clearly arranged and provides a range of structural information, which would only be visible in the XML scheme definition. This chapter describes the notation of the table format which is used more intensively starting in chapter 8.4.

All element names, data types and attributes are written in English to make exchange with European partners more convenient.

8.1.1 Display of XML Elements in the Text

This document is intended to provide consistent notation of XML elements to make technically important information available to the reader.

- XML elements are written in upper camel case, bold and italics, e.g.: ***VehicleJourneyRef***. The element names – where possible and meaningful – are derived from Transmodel terms. If a corresponding term for a concept or object is missing in Transmodel, an attempt was made to adopt a corresponding term from JourneyWeb or a matching concept from DELFI.
- Data types are displayed italics, e.g. : *xsd:boolean*.
- Code examples are displayed in smaller font.

8.1.2 Display of Relationships

When a dataset described by an object model (ie: a set of object instances) is exchanged as a data request (for example in an XML based API such as OJP) it must be serialised into a stream of bytes representing both the data values and the relationships between the instances, the two both organised by delimiters such as XML tags. In practice there are three ways of representing the relationships between object instances when serialised:

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

An *implicit mechanism* is used when one element is contained in another, ie: grouped spatially within child tags. In this way a direct child relationship can be expressed by immediate proximity. An *explicit mechanism*, in contrast, indicates a relationship by the use of the key of the referenced object as an object that references it; an *internal reference* is an object key where the referenced object is also defined in the same document as the referencing object (and so whose scope of uniqueness need only be local to the exchange, eg: an identifier of a message); an *external reference* is an object key whose referent is defined outside the document (and whose scope must be defined by a namespace common to both parties of the exchange, eg: a persistent stop number). References may consist of compound keys (for more detail see 8.2).

It is helpful to be able to distinguish between the use of an identifier to provide the key of an object as part of the object definition and its use as a reference to indicate a relationship to the object from another object. In OJP, the following convention is used to do this:

- An identifier is an attribute (ie: a child element) of the defining element that indicates a primary key for the defining element. These identifiers end with a signal word like “code” or “identifier” (sometimes “number” in SIRI), eg: a journey contains the key JourneyCode.
- If an object is referenced by another object, the referencing element (external key) ends with “Ref”. For example, the reference for a journey (from a departure board) is: JourneyRef.
- An object instance and its reference use a common underlying data type for the identifier. For example JourneyCode and JourneyRef are both of type JourneyCodeType. This allows for a degree of automated type checking so that only objects of the allowed type are made.

8.1.3 Table Notation of XML Structures

The current document displays XML structures in tabular notation (see Table 1). There is a separate table for each important OJP-request/response-element. Additional tables are indicated for all essential 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 for the following tables. They use a consistent set of rules to describe XML elements and their related conditions.

Table 1 — Example for the tabular notation of an XML structure

Grouping	Element name		Min : Max	File type	Description
<i>ContinuousServiceStructure</i>				<i>+Structure</i>	A passenger movement on a continuous, non-timetabled service.
	<i>a</i>	<i>ContinuousMode</i>	-1:1	<i>walk / demandResponsive / replacementService</i>	Continuous transport mode (see 8.4.3.1).
	<i>b</i>	<i>IndividualMode</i>		<i>walk / cycle / taxi / self-drive-car / others-drive-car / motorcycle / truck</i>	Individual transport mode (see 8.4.3.1).
<i>DatedJourney</i>	<i>OperatingDay</i>		1:1	<i>→OperatingDay</i>	Reference to an Operating Day (see 8.4.4.1).
	<i>VehicleRef</i>		0:1	<i>→Vehicle</i>	Reference to a vehicle (see 8.4.4.1).
<i>ServiceJourney</i>	<i>JourneyRef</i>		1:1	<i>→Journey</i>	Reference to a journey (see 8.4.4.1).
<i>LineIdentity</i>	<i>LineRef</i>		1:1	<i>→Line</i>	Reference to a line (see 8.4.4.1).
	<i>DirectionRef</i>		1:1	<i>→Direction</i>	Reference to a direction (see 8.4.4.1).
<i>Service</i>	<i>Mode</i>		1:1	<i>+Mode</i>	Vehicle mode (see 8.4.3).
	<i>PublishedLineName</i>		1:1	<i>InternationalText</i>	Line name or service description as known to the public, eg: "512", "S8" or "Circle Line".
	<i>OperatorRef</i>		0:1	<i>→Operator</i>	Reference to an operator. (see 8.4.4.1).
	<i>RouteDescription</i>		0:1	<i>InternationalText</i>	Descriptive text for a route, eg: "Airport via City Centre".
	<i>Via</i>		0:*	<i>+ServiceViaPoint</i>	Via points of the service that may help identify the vehicle to the public (see 8.4.6.1).
	<i>Attribute</i>		0:*	<i>+GeneralAttribute</i>	Note or service attribute (see 8.4.4.10).
<i>ServiceOrigin</i>	<i>OriginStopPointRef</i>		0:1	<i>→StopPoint</i>	First stop of the vehicle journey; origin stop point (see 8.4.5.1).
	<i>OriginText</i>		0:1	<i>InternationalText</i>	Label for first stop.

<i>ServiceDestination</i>	<i>DestinationStopPointRef</i>	0:1	<i>→StopPoint</i>	Last stop of vehicle journey; destination stop point (see 8.4.5.1).
	<i>DestinationText</i>	0:1	<i>InternationalText</i>	Label for last stop.
	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>	Reference to situation message. Message details might be found in response context or through other communication channels (see 8.4.8.2).

8.1.3.1 Grouping

The first column, when filled, contains an identifier that organises the elements into meaningful groupings, e.g. *Service* or *ServiceOrigin*. This is purely for documentation purposes and corresponds in most cases to the names of an XML group that is used in the XML schema. The use of groupings is intended to organise elements to improve clarity and reusability.

8.1.3.2 Element Name

Element names are written in italics in the second column, e.g. *OperatingDay*. If an element is required, it is written in **bold**. Optional elements are not printed in bold. The name of the structure itself is displayed in the upper left of the table.

Elements that are derived by extension or are used anonymously, have three colons “:::” in the name field.

8.1.3.3 Multiplicity & Choice (min:max)

The third column (Min:Max) indicates whether an element is required or optional or whether it occurs once or multiple times in the superior element. For this purpose, the standard UML conventions “min:max” are used, so that e.g. “0:1” stands for an optional, single element, “**1:1**” indicated a required, single element, “0:*” stands for an optional, multiple element etc. Required elements are written in **bold**.

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

8.1.3.4 Data Type

The data types are displayed in the fourth column in *italics*, e.g. *InternationalText*. If the namespace differs from an OJP namespace, it is also indicated, e.g. “*xs:dateTime*” or “*siri:PtSituationElement*”.

- A complex data type, which itself contains structures as child elements, is indicated in the data type column with “+Structure”.
- Where elements are used as references (external key) for other objects, the type of referenced object is used as a data type with a preceding arrow. For example “*→StopPoint*” as the type of a reference (StopPointRefStructure) for an object of type “StopPointType”.
- Enumerated types are mostly displayed directly with usable values, e.g. “walk | cycle”. Only in a few cases with extensive enumerations, which are reused in many places, is a type declared and referenced.

- In order to save space, data types have been truncated. So endings like “Structure” and “Type” are not written out. Instead of “InternationalTextStructure” the data type will be indicated as “InternationalText”.

8.1.3.5 Explanation

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

8.1.4 Message Exchange

This chapter explains how OJP messages are exchanged. There are two basic procedures in use

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

These procedures are already established and in use, e.g. in the SIRI interfaces.

8.1.5 Use of SIRI Procedure

In SIRI, the message exchange procedure mentioned above has been defined and described, see CEN, EN 15531, Part 2. This procedure has some advantages: previously tested processes and existing SIRI implementations can be used when implementing OJP services, something that can save time and costs.

The basic procedure is the 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 7).



Figure 7 — Request with synchronous response (from SIRI, (CEN, EN 15531, Part 2)).

Requests with synchronous response are used for almost every OJP service (one exception is the notification service). The role of the requester is taken on by the Home System, which sends requests to the Responding System. If a Distributing System is "in the middle" then a concatenation of the data request / response pattern (as shown in Figure 7) applies.

Somewhat more complicated is the subscription mechanism. A data consumer is interested in new messages, but does not know when they will occur. Instead of making regular requests and thus creating a base load (and to risk learning about the new message only as a result of the next regular request), they can set up a subscription.

Figure 8 shows the basic connections. The data consumer has to fill two roles, that of the subscriber and that of the notification consumer. The data consumer makes a subscription request from the server. In doing so, they indicate to the server which type of incidents should be sent. The server sets up the

subscription by registering it with the subscription manager. Afterwards, something occurs when an incident arises that needs to be communicated to consumers. In this case the server, as the notification producer, sends the message to the data consumers with the new incident (delivery). This process is repeated until the subscription runs out or is ended by the data consumers.

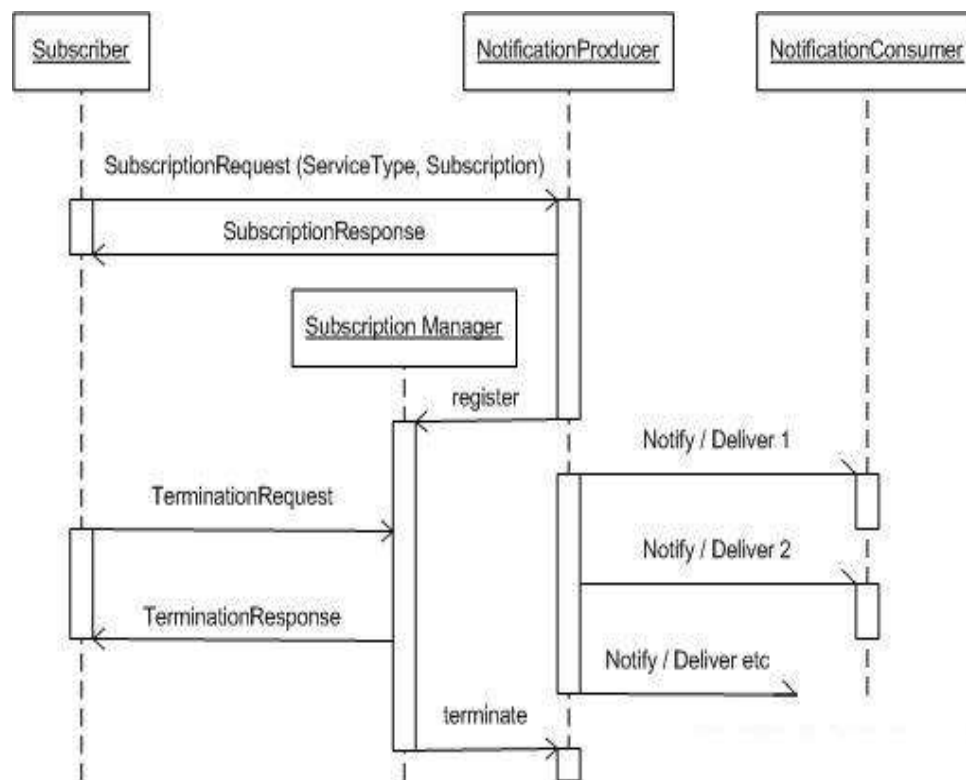


Figure 8 — Publish / Subscribe with asynchronous notifications (from SIRI, (CEN, EN 15531, Part 2)).

Both client and server have to fulfil two roles, the client has the roles of the subscriber and notification consumer, and the server has the roles of the notification producers and subscription manager. In most implementations there is no distinction and each individual software component fulfils both roles.

The subscription process is completed by additional requests. The status requests allow the status of the interface partner to be requested and its availability to be tested. The heartbeat request, one which is sent regularly by the server, enables a data consumer to recognise when a server is available and sends signals (ping or heartbeat). Details can be found in SIRI (CEN, EN 15531, Part 2), Chapter 5.

In OJP services the subscription mechanism exists as a notification service when a data consumer (through their Home System) wants to inform themselves about incidents or other occurrences.

8.1.6 HTTP and REST

The conversion of SIRI messaging procedure occurs in OJP using HTTP/1.1 (Hypertext Transfer Protocol¹) as a transport protocol and XML (Extensible Markup Language²) for the message content. This is done following the rules of the the SIRI Lite protocol described in "**Chapter 12- SIRI for Simple Web Services – SIRI Lite**" of CEN, EN 15531, Part 2.

¹ <http://tools.ietf.org/html/rfc2616>

² <http://www.w3.org/XML/>

An HTTP-request is immediately responded to by a server using the previously opened IP port. For example, for a planned trip a client sends a request as an HTTP-request with the XML-element OJP and TripRequest as one of the child elements in the POST-Block. The server answers synchronously in the HTTP-response with XML element OJP and TripResponse as one of the indirect child elements.

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

For larger messages, the use of a compression procedure is recommended. This type of method is also specified for HTTP.

System designers will need to consider the need for security and assurance of data, such as through the use of VPN or HTTPS technologies – but this is beyond the scope of this Technical Specification.

8.1.7 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.

Only for the notification service, if the subscription procedure is used, is the situation more complicated. In this case, the data consumer and the data provider both have to assume the roles of client and server in the sense of HTTP. Because when the data provider (notification producer) wants to send a message to the data consumer, it becomes a client (in the sense of HTTP) and the data consumer then becomes a server. In other words, the active part that initiates a communication is always an HTTP-client and the passive part that responds to a message always plays the role of an HTTP-server.

8.2 Identification of Objects beyond system borders

To ensure that different systems can reference the same object, an object ID is required that is recognised by all systems. Within the scope of the OJP interfaces, stops, routes and transport authorities are examples of such object types that require information exchange over interface services. For this reason, there is a need for referencing systems that are generally known and can be commonly used.

That does not necessarily mean that a software system has to use these object keys in their own operation. It suffices, when the system understands the general object references and is able to depict the relevant information using internal identifiers.

Object identifiers can be made up of a prefix and an identifier string, separate by a colon.

- The prefix is a namespace (or 'Codespace' in the NeTEx sense), a string indicating the coding system being used (for long identifiers it may also indicate a common root value, as say declared by a Codespace element). The prefix should not contain colons. Where there is a nationwide system the country code may conveniently be used as a namespace (eg: de, fr, ch, etc). If no prefix is explicitly stated then a default code space is assumed by prior agreement.
- The identifier string is a key value within the chosen coding system that uniquely identifies an instance of the object. It may be atomic - based on a single attribute - or composite - based on multiple attributes, some of which may be keys of other object types. Colons may be used to separate the attribute values of a composite key, for example '9162:1:2'. Where a composite key is used, a namespace should also always be used so as to avoid confusion between the namespace delimiter and the first colon of the composite key, ie: that 'de:9162:1:2' should be read as 'de' + '9162:1:2' and not 9162 + 1:2. An IFOPT-oriented syntax is used for the schemas 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 an implementation of the OJP Schema it is possible to declare one or more namespaces and namespace identifiers that are in use in a given data exchange (eg: by adding a tag to do this to the request/response). The namespace identifiers can then be used both to formally indicate the coding system being used, and to reduce verbosity.

In what follows the referencing systems required for different object types will be presented.

8.2.1 Stops and Stopping Points

When referencing stops and stopping points there was a European Norm IFOPT (CEN, EN 28701:2012, 2012), now mainly covered in Transmodel v6. However in section 6.8.1 of IFOPT, a syntax for the structure of a referencing key is described in detail (and does not appear in TMv6).

Structure of an IFOPT object key
Country_code:region:stop_number:stop_area:stop_point

The following example shows the (hierarchical) structure of keys for a stop, stop area and a stop point

Practical example:

Public transport stop “Karlsplatz (Stachus) in Munich”:

Stop object	Unique ID
Stop Karlsplatz (Stachus) in Munich	de:9162:1
Stop area Subway U4/5	de:9162:1:2
Stop 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 OJP location information services from the Responding System.

8.2.2 Localities and Districts

For the operation of OJP interfaces we recommend the use of a topographic place key that is structurally similar to the IFOPT Norm for stops:

Structure of a topographic place key
Country_code:district_code: topographic_place

Example	
Ilmenau	de:16070029:1

Client systems that do not have their own data supply can obtain object references for districts and localities using OJP location information services from the Responding System.

8.2.3 Addresses and POIs

To operate OJP 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.

8.2.4 Organisations: Transport Companies and Transport Authorities

An organization code is used to uniquely reference transport companies and transport authorities. In order to have these codes remain unique across multiple systems, it is recommended that comprehensive databases (optimally national ones) of transport companies is built.

Structure of an organization code
Country_code:organization_code

Examples	
Transport and Tariff Assoc. Stuttgart	de:vvs
Stuttgarter Straßenbahn AG	de:ssb
Fernverkehr Deutsche Bahn	de:dbag
DB Regio Baden-Württemberg	de:dbregiobw

8.2.5 Lines and Line Directions

The line keys of the responsible data suppliers are used to uniquely reference lines. Those considered a responsible data supplier are the representative transport company (licence holder) or the responsible transport authority.³ In order to keep these codes unique across multiple data suppliers, the organization code (cf. 8.2.4) is prefixed as a namespace.

³ In order to avoid redundancy in routes from multiple data suppliers we recommend compiling a comprehensive (when possible national) database of lines.

Structure of a line key
Country_code:organization_code:line_key

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

A direction code from the responsible data supplier is used to uniquely reference line directions. The direction code can be freely selected by the data supplier and first becomes understandable to passengers with accompanying text. The direction code is only used in a line context so that the prefixing of line keys as a namespace is not necessary.

Structure of a direction code
Direction_code

Examples	
Outbound	0
Inbound	I
Outbound	1
Inbound	2
Towards the city	T
Away from the city	A

8.2.6 Journeys

A journey key from the responsible data supplier is used to uniquely reference journeys. The journey key can be freely selected by the data supplier as long as it is unique in the namespace.

Structure of a journey key
Country_code:organization_code:line_key:journey_key

Example	
Journey 1512 of line U1 in Stuttgart	de:vvs:20001:1512

If an organization (transport authority) does not organise their journeys into lines (e.g. rail transport), the line key can remain empty.

Example	
ICE 612 of DB AG	de:dbag::612

8.2.7 Vehicles

A vehicle code of the responsible data supplier is used to uniquely reference vehicles.

In real-time interfaces (SIRI ET), control centres send a vehicle code with a timetabled journey so that a Responding System for a specific operating day can know which journey is performed by which vehicle. For this reason, the assignment from vehicle code to journey key has to be unique for each operating day.

Structure of a vehicle code
Country_code:organization_code:vehicle_code

Example	
Vehicle 5812 of SSB AG	de:ssb:5812

8.2.8 Operating Days

A timetabled journey only becomes a specific journey when combined with an operating day. An operating day can also include times past twelve midnight and therefore differ from a calendar day. Yet any discrepancy in days and its extent is not relevant for passenger information. Passengers are only provided time and date details according to the calendar day principle.

An operating day in OJP is displayed according to the ISO 8601 Norm.

Example	
29 March 2013	2013-03-29

8.2.9 Owners

The term 'owner' refers to operators of stop assets and passenger information devices. Normally they are transport companies, but can also be (for example) a cinema owner who has set up and operates a monitor for the display of the next departures from a nearby stop. Owners are referenced in the same manner as those of transport companies and authorities (cf. 8.2.4).

8.2.10 Stop- and Vehicle Equipment

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

8.2.11 Participating Systems / IT Systems

The OJP 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 are responsive, they require codes (in SIRI they are known as participant references).

Structure of a system code
Country_code:organization_code:system_code

Example	
Public Information System of VVS	de:vvs:publicEKAP

8.2.12 Incident Messages

Incident and service disruption messages are transferred using SIRI SX structures. The allocation of IDs for incident messages is also regulated there. The message IDs are transferred in the context of the participating system (see 8.2.11) and are therefore globally unique.

8.2.13 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 authorities that are responsible for authority fares, companies for their own company fares. Fare authority referencing is done in the same fashion as for transport companies and authorities (cf. 8.2.4).

8.2.14 Tariff Zones

The coding of tariff zones is the responsibility of the respective fare authority (cf. 8.2.13). Tariff zones are specified in the context of the respective fare authority and are therefore globally unique.

8.2.15 Tickets and Traveller Cards

Ticket coding is performed by the respective fare authority (cf. 8.2.13). Tickets are specified in the context of the respective fare authority and are therefore globally unique.

The coding of traveller cards, e.g. BahnCard50 of the Deutschen Bahn AG, is the responsibility of the respective fare authority (cf. 8.2.13). A traveller card code has to be indicated in the namespace of the fare authority.

Structure of a code for traveller cards
Country_code:organization_code:traveller_card_code

Example	
BahnCard50 of DB AG	de:dbag:BC50

8.3 Services and XML Schemas

This document describes interface definitions for services between software components.

The OJP interface services are defined as XML schemas. An overview of the services and their implementation as XML schemas makes up the first part of the section below. Several structure definitions are useful in multiple services and therefore are hierarchically defined in several schema files as a common basis to enable repeated use. The approach followed is very strongly oriented towards the principles of object orientation. The commonly used structure definitions are described in the second part of this section. The third part introduces the XML schemas that are imported from SIRI. A classification of error states can be found in the fourth part of this section.

8.3.1 Services Provided

The OJP interface family currently comprises the following services:

Table 2 — List of the OJP services and their request elements.

Service	Name of request element	Schema file	Section
Location information	LocationInformationRequest	OJP_Locations.xsd	8.5
Exchange points	ExchangePointsRequest	OJP_Locations.xsd	8.6
Trip request	TripRequest	OJP_Trips.xsd	8.7
Distributed journey planning	MultiPointTripRequest	OJP_Trips.xsd	8.7.2
Departure board	StopEventRequest	OJP_StopEvents.xsd	8.8
Trip/Vehicle information	TripInfoRequest	OJP_TripInfo.xsd	8.9
Ticket price calculation	FareRequest	OJP_Fare.xsd	8.10

8.3.2 XML Schemas Used Across Services

In order to avoid the need to redundantly define structures that are used in more than one service, a commonly used basic XML schema was introduced to include them hierarchically. The inclusion sequence and the customization of schema files have been selected to place content-related elements together in a file and to only include in each schema, that which is the required for the task.

The commonly used basic schema files are described in detail in section 8.4. The hierarchic dependency between the individual schemas is shown in the following figures. The blue-coloured schema files are imported from SIRI.

The XSD file describing the XML schema can be freely downloaded from <http://www.vdv.de/ojp>.

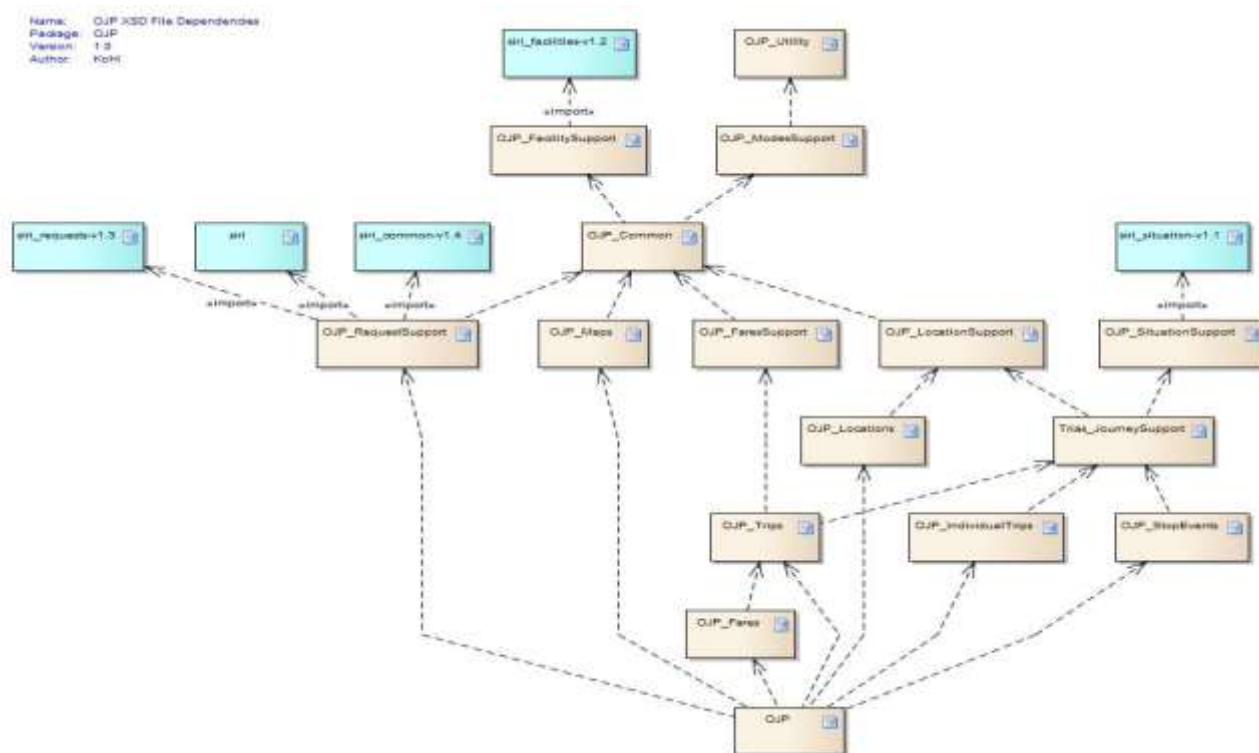


Figure 9 — Overview of hierarchy of schema files

8.3.3 Imported Schemas

Several schema files are imported from the SIRI interfaces to OJP. These files are depicted in blue in Figure 9. By importing from SIRI definitions the SIRI process for message exchange can also be used for the exchange of OJP messages. Additionally, specific structure definitions can be reused from SIRI, a fact that ensures the consistence between these interface standards. This affects the definition of modes, incidents and stop assets or facilities.

The imports have been taken from SIRI v2.0.

8.3.4 Error States When Operating OJP Services

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

- Are inherited from the SIRI services,
- Describe general OJP error situations across services or
- Indicate service-specific error situations.

The OJP 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 (**OJPGENERIC_**).

8.3.5 Error Codes from SIRI

In SIRI (CEN, EN 15531, Part 2), section 5.7, a number of error codes are defined that play an important role for the message transfer procedure. These codes have been divided into the following groups: success, systemic error and application error (cf. Table 3). The table lists the error codes taken from SIRI v2.0; the OJP schema, however, does not enumerate a list of codes that have to be used. Each system should use those which are relevant and any additional ones that may be necessary.

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

Group	Condition	Description
Success	OK (true)	Request successful
Systemic Error	<i>RequestTimeout</i>	Server not responding.
	<i>InvalidRequest</i>	The server does not "understand" the request. The client should not repeat the request.
	<i>Unauthorized</i>	User name and password are required for the request, or credentials not satisfied.
	<i>Forbidden</i>	The server "understands" the request, but cannot carry it out.
	<i>NotFound</i>	The requested URL was not found.
Distribution	<i>UnapprovedKeyAccessError</i>	Error: Recipient of a message to be distributed is not available. +SIRI v2.0
	<i>UnknownParticipant</i>	Recipient for a message to be distributed is unknown. +SIRI v2.0
	<i>UnknownEndpoint</i>	Endpoint to which a message is to be distributed is unknown. +SIRI v2.0
	<i>EndpointDeniedAccess</i>	Distribution message could not be delivered because not authorised. +SIRI v2.0
	<i>EndpointNotAvailable</i>	Recipient of a message to be distributed is not available. +SIRI v2.0
Access	<i>UnapprovedKey</i>	User authentication key is not approved. SIRI v2.0

Group	Condition	Description
Application Error	<i>VersionNotSupported</i>	Service is not available.
	<i>CapabilityNotSupported</i>	Service does not support the requested capability.
	<i>ServiceNotAvailable</i>	Functional service is not available to use (but it is still capable of giving this response).
	<i>AccessNotAllowed</i>	Requestor is not authorised to the service or data requested.
	<i>InvalidDataReferences</i>	Request contains references to identifiers that are not known. +SIRI v2.0
	<i>BeyondDataHorizon</i>	Request is for data outside of real-time data horizon.
	<i>NoInfoForTopic</i>	Valid request was made but service does not hold any data for the requested topic expression.
	<i>ParametersIgnored</i>	Request contained parameters that were not supported by the producer. A response has been provided but some parameters have been ignored. +SIRI v2.0
	<i>UnknownExtensions</i>	Request contained extensions that were not supported by the producer. A response has been provided but some or all extensions have been ignored. +SIRI v2.0
	<i>UnknownSubscriber</i>	Subscriber not found.
	<i>UnknownSubscription</i>	Subscription not found.
	<i>AllowedResourceUsageExceeded</i>	Valid request was made, but request would exceed the permitted resource usage of the client.
	<i>OtherError</i>	Other Error Type.

8.3.6 General OJP Error States

In *ErrorMessage*, the following general error states can appear:

Table 4 — General OJP error messages that can appear in any message.

<i>OJPGENERIC_REQUESTNOTSUPPORTED</i>	The server does not support the specific request (eg: MultiPointTripRequest).
<i>OJPGENERIC_FEATURENOTSUPPORTED</i>	The server does not support the requested feature (eg: parameter NotVia in TripRequest)
<i>OJPGENERIC_LANGUAGENOTSUPPORTED</i>	For the display of texts within the result, the server does not support (at least within the context of this request) the language required by the requestor.
<i>OJPGENERIC_EXCEPTIONFROMREQUESTEDLANGUAGE</i>	For the display of texts within the result, the server does not support the language required by the requestor for all of the occurring text elements.
<i>OJPGENERIC_DATAFRAMEREFNOTAVAILABLE</i>	The server cannot provide the data frame (data version) required by the requestor.

8.3.7 Time Zones

It should be noted that xs:dateTime does not require timezone to be declared, but this always must be declared in systems which involve more than one timezone.

8.4 Common XML Structures

This chapter is intended to explain basic XML structures that are commonly used by several OJP services. The organization of the sections in this chapter follows the structure of the XML schema files.

8.4.1 Root Element OJP

The XML schema file OJP.xsd defines the general root element *OJP* which is shared by all messages of all OJP services.

The root element is named ***OJP*** and is a choice between ***OJPRequest*** (***OJPRequestStructure*** defined as a SIRI ***RequestGroup***) and ***OJPResponse*** (***OJPResponseStructure*** defined as a SIRI ***ResponseGroup***). ***OJPRequest*** and ***OJPResponse*** both propose all the possible SIRI request and response elements, but OJP focus is only on the ***ServiceRequest*** (see ***ServiceRequestStructure***) and the ***ServiceDelivery*** (see ***ServiceDeliveryStructure***).

The subsequent sections describe the complex structures defined in OJP.xsd.

8.4.1.1 ServiceRequestStructure

Table 5 — Description of *ServiceRequestStructure*.

		<i>ServiceRequestStructure</i>		+Structure	Request from a Consumer to a Producer for immediate delivery of data. Answered with a <i>ServiceDelivery</i> . (For <i>Fetches Delivery</i> this will be after a further <i>DataReadyRequest</i>).
		<i>ServiceRequestContext</i>	0:1	+Structure	General request properties – typically configured rather than repeated on request.
log		<i>RequestTimestamp</i>	1:1	xsd:dateTime	Timestamp on request.
Auth.		<i>AccountId</i>	0:1	+Structure	Account Identifier. May be used to attribute requests to a specific user account for authentication or reporting purposes +SIRI v2.0
		<i>AccountKey</i>	0:1	+Structure	Authentication key for request. May be used to authenticate the request to ensure the user is a registered client. +SIRI v2.0
Endpoi nt Propert ies		<i>Address</i>	0:1	EndpointAddress	Address to which response is to be sent: [<i>Notify</i>] endpoint. If omitted, this may also be determined from <i>RequestorRef</i> and preconfigured data, or the http request.
		<i>RequestorRef</i>	1:1	→ParticipantCode	Identifier of Requestor. May be used to identify an individual participant system or individual device client. If used for a device client should be an anonymous token, divulged with user consent.
		<i>MessageIdentifier</i>	0:1	MessageQualifier	Arbitrary identifier that may be given to message.
Delegat or Endpoi nt		<i>DelegatorAddress</i>	0:1	EndpointAddress	Address of originating system to which delegated response is to be returned. +SIRI 2.0. If request has been proxied by an intermediate aggregating system this provides tracking information relating to the original requestor. This allows the aggregation to be stateless.

		DelegatorRef	0:1	→ParticipantCode	Identifier of delegating system that originated message. +SIRI 2.0
		Concrete service subscription			If more than one, shall all be same type.
Payload	a	OJPFareRequest	-1:*	OJPFareRequest	Request for retrieving fare information (see 8.10.2.1).
	b	OJPPlaceInformationRequest		OJPPlaceInformationRequest	Request for resolving locations (see 8.5.3.1).
	c	OJPStopEventRequest		OJPStopEventRequest	Request for stop-centric arrivals/departures (see 8.8.2.1).
	d	OJPTripRequest		OJPTripRequest	Request for inter-modal journey planning (see 8.7.3.1).
	e	OJPEXchangePointsRequest		OJPEXchangePointsRequest	Request for Exchange Points (see 8.6.2.1)
	f	OJPTripInfoRequest		OJPTripInfoRequest	Request for Trip Information (see 8.9.2.1)
	g	OJPMultiPointTripRequest		OJPMultiPointTripRequest	Request for Multi Point Trips (see 8.7.3.2)

Table 6 — Description of *ServiceRequestContext* (from SIRI)

ServiceRequestContext			+Structure	General request properties – typically configured rather than repeated on request.	
Server Endpoint Address	CheckStatusAddress		0:1	EndpointAddress	Address to which CheckStatus requests are to be sent.
	SubscribeAddress		0:1	EndpointAddress	Address to which requests for new subscriptions are to be sent.
	ManageSubscriptionAddress		0:1	EndpointAddress	Address to which requests to manage existing subscriptions are to be sent. If absent, same as SubscribeAddress .
	GetDataAddress		0:1	EndpointAddress	Address to which requests to return data are to be sent.
Client Endpoint Address	StatusResponseAddress		0:1	EndpointAddress	Address to which CheckStatus responses and HeartbeatNotification messages are to be sent. If absent, same as SubscriberAddress .
	SubscriberAddress		0:1	EndpointAddress	Address to which subscription responses are to be sent.
	NotifyAddress		0:1	EndpointAddress	Address to which notifications requests are to be sent. If absent, same as SubscriberAddress .
	ConsumerAddress		0:1	EndpointAddress	Address to which data is to be sent. If absent, same as NotifyAddress .
Namespace	DataNameSpaces		0:1	+Structure	Scope for identifiers
NameSpace	a	StopPointNameSpace	0:1	xsd:anyUrl	Namespace for stop references.
	b	LineNameSpace	0:1	xsd:anyUrl	Namespace for Line names and Directions.
	c	ProductCategoryNameSpace	0:1	xsd:anyUrl	Namespace for product categories

	<i>d</i>	ServiceFeatureNameSpace	0:1	<i>xsd:anyUrl</i>	Namespace for Service Features
	<i>e</i>	VehicleFeatureNameSpace	0:1	<i>xsd:anyUrl</i>	Namespace for Vehicle features
<i>Language</i>	Language		0:1	<i>xml:lang</i>	Default language.
<i>Location</i>	<i>a</i>	WgsDecimalDegrees	0:1	<i>EmptyType</i>	Geospatial coordinates are given as WGS84 latitude and longitude, decimal degrees of arc.
	<i>b</i>	GmlCoordinateFormat		<i>srsNameType</i>	Name of GML Coordinate format used for Geospatial points in responses.
<i>Units</i>	DistanceUnits		0:1	<i>xsd:normlizedString</i>	Units for <i>DistanceType</i> . Default is metres. +SIRI v2.0
	VelocityUnits		0:1	<i>xsd:normlizedString</i>	Units for <i>VelocityType</i> . Default is metres per second. +SIRI v2.0
<i>Temporal Span</i>	DataHorizon		0:1	<i>PositiveDurationType</i>	Maximum data horizon for requests
	RequestTimeout		0:1	<i>PositiveDurationType</i>	Default Timeout for requests
<i>Delivery Method</i>	DeliveryMethod		0:1	<i>DeliveryMethodEnum</i>	Delivery interaction pattern to use to deliver data.
	MultipartDespatch		0:1	<i>xsd:boolean</i>	Whether multi-part delivery is allowed, i.e. the breaking up of updates into more than one delivery messages with a MoreData flag,
	ConfirmDelivery		0:1	<i>xsd:boolean</i>	Whether Consumers should issue an acknowledgement on successful receipt of a delivery. Default is 'false'.
<i>Resource Use</i>	MaximumNumberOfSubscriptions		0:1	<i>xsd:positiveInteger</i>	Maximum number of subscriptions that can be sustained by a subscriber
<i>Prediction</i>	AllowedPredictors		0:1	<i>AllowedPredictorEnum</i>	Who may make a prediction. Documentation only. Default is 'anyone'.
	PredictionFunction		0:1	<i>xsd:string</i>	Allows a name to be given to the prediction function. Documentation only.
<i>any</i>	Extensions		0:1	<i>any</i>	Placeholder for user extensions.

8.4.1.2 ServiceDeliveryStructure

Table 7 — Description of *ServiceDeliveryStructure*.

ServiceDelivery			<i>+Structure</i>	Response from Producer to Consumer to deliver payload data. Either answers a direct ServiceRequest, or satisfies a subscription asynchronously. May be sent directly in one step, or be fetched in response to a Data Supply Request.
<i>Attributes</i>	srsName	0:1	<i>xsd:string</i>	Default GML coordinate format for any spatial points defined in response by Coordinates parameter.
<i>Log</i>	ResponseTimestamp	1:1	<i>xsd:dateTime</i>	Time individual response element was created.
<i>Endpoint properties</i>	ProducerRef	0:1	<i>→ParticipantCode</i>	Participant reference that identifies producer of data. May be available from context.
	Address	0:1	<i>EndpointAddress</i>	Address to which any acknowledgment should be sent. Only needed if ConfirmDelivery specified.
	ResponseMessageIdentifier	0:1	<i>MessageQualifier</i>	An arbitrary unique reference associated with the response which may be used to reference it.

	RequestMessageRef		0:1	→MessageQualifier	Reference to a unique message identifier associated with the request which gave rise to this response.
Delegator endpoint	DelegatorAddress		0:1	EndpointAddress	Address of originating system to which delegated response is to be returned. +SIRI 2.0. If request has been proxied by an intermediate aggregating system this provides tracking information relating to the original requestor. This allows the aggregation itself to be stateless.
	DelegatorRef		0:1	→ParticipantCode	Identifier of delegating system that originated message. +SIRI 2.0
Status	Status		0:1	xsd:boolean	Whether the complete request could be processed successfully or not. Default is <i>true</i> . If any of the individual requests within the delivery failed, should be set to <i>false</i> .
	ErrorCondition		0:1	See below	Description of any error or warning conditions that applies to the overall request. More Specific error conditions should be included in the error conditions attached to each functional service response that fails.
	a	CapabilityNotSupportedError	1:1	+Error	Capability not supported.
	b	OtherError		+Error	Error other than a well-defined category.
	c	Description	0:1	→ErrorDescription	Description of Error.
	MoreData		0:1	xsd:boolean	Whether there are more delivery messages making up this data supply group. Default is <i>false</i> . Optional SIRI Capability: <i>MultipartDespatch</i> .
Payload	Concrete SIRI Service:				One or more of a single type of the following:
	a	OJPFareResponse		+OJPFareResponse	Response to fare request (see 8.10.3.1).
	b	OJPLocationInformationResponse		+OJPLocationInformationResponse	Response to location information request (see 8.5.4.1).
	c	OJPStopEventResponse		+OJPStopEventResponse	Response with stop-centric arrivals/departures (see 8.8.3.1).
	d	OJPTripResponse		+OJPTripResponse	Response to inter-modal trip request (see 8.7.4.1).
	e	OJPExchangePointsResponse		+ExchangePointsResponse	Response to Exchange Points request (see 8.6.2.1)
	f	OJPTripInfoResponse		+TripInfoResponse	Response to Trip Information request (see 8.9.3.1)
	g	OJPMultiPointTripResponse		+MultiPointTripResponse	Response to Multi-Point Trip Request (see 8.7.3.2)

8.4.2 OJP_Utility

The XML schema file OJP_Utility.xsd contains a range of basic types and structures that can be re-used when defining more complex structures. The definitions within this file have a more general scope and are not specifically related to the public transport domain.

8.4.2.1 Simple Types

This schema file defines the following simple types:

Table 8 — List of simple types OJP_Utility.xsd.

<i>PercentType</i>	<i>xs:nonNegativeInteger</i>	Percent value as integer, limited to max value 100.
<i>OpenPercentType</i>	<i>xs:nonNegativeInteger</i>	Percent value as integer, no upper limit.
<i>ValidDayBitType</i>	<i>xs:string</i>	String that consists of only zeros and ones representing days.
<i>DistanceType</i>	<i>xs:nonNegativeInteger</i>	Data type for distances, in metres.
<i>LengthType</i>	<i>xs:nonNegativeInteger</i>	Data type for lengths, in metres.
<i>SpeedType</i>	<i>xs:nonNegativeInteger</i>	Data type for speed, in metres per second.
<i>PriorityType</i>	<i>xs:nonNegativeInteger, [1,5]</i>	Data type for priority values, from 1 (highest) to 5 (lowest).
<i>LongitudeType</i>	<i>xs:decimal</i>	Longitude from Greenwich.
<i>LatitudeType</i>	<i>xs:decimal</i>	Latitude from equator.
<i>AltitudeType</i>	<i>xs:decimal</i>	Altitude metres from sea level..
<i>AbsoluteBearingType</i>	<i>xs:nonNegativeInteger</i>	The absolute compass bearing in degrees. North = 0, clockwise.
<i>PhoneNumberType</i>	<i>xs:normalizedString</i>	Type definition for phone numbers

8.4.2.2 InternationalTextStructure

OJP_Utility defines the following complex types:

Table 9 — Description of *InternationalTextStructure*.

<i>InternationalTextStructure</i>			<i>+Structure</i>	A text together with a text ID and a code for its language.
	Text	1:1	<i>siri:NaturalLanguageStringStructure</i>	Text.
	TextId	0:1	<i>xs:NMTOKEN</i>	ID of the Text.

Elements of type *InternationalText* can be used to express texts in different languages.

8.4.2.3 GeoPositionStructure

Table 10 — Description of *GeoPositionStructure*.

<i>GeoPositionStructure</i>			<i>+Structure</i>	Geographical position expressed as WGS84.
	Longitude	1:1	<i>Longitude</i>	Longitude from Greenwich Meridian. -180 (West) to +180 (East).
	Latitude	1:1	<i>Latitude</i>	Latitude from equator. -90 (South) to +90 (North).
	Altitude	0:1	<i>Altitude</i>	Altitude metres from sea level.

8.4.2.4 WebLinkStructure

Table 11 — Description of *WebLinkStructure*.

<i>WebLinkStructure</i>			<i>+Structure</i>	URL pointing to a resource on the web.
	Label	1:1	<i>InternationalText</i>	Labelling text for link.
	Url	1:1	<i>xs:anyURI</i>	URL pointing to web resource.

OJP responses contain at some places elements of the type `WebLinkStructure` to allow for subsequent retrieval of further information via a web link. A client system may use the embedded URL to process further actions or simply obtain more information. As client systems may be different (eg: smartphone app or desktop web browser) providers of the web contents behind these URLs are challenged to support as many different client systems as possible in an optimised way.

8.4.3 OJP_ModesSupport

The XML schema file `OJP_ModesSupport.xsd` defines several base types and structures for the classification of modes of transport. These definitions have their origin in the TPEG standard which is also used in SIRI.

8.4.3.1 Simple Types

This schema file defines the following simple types:

Table 12 — List of simple type definitions in `OJP_ModesSupport.xsd`.

<i>IndividualModesEnumeration</i>	<i>walk / cycle / taxi / self-drive-car / others-drive-car / motorcycle / truck</i>	Modes of individual transport.
<i>PrivateModesEnumeration</i>	<i>carPooling</i>	Mobility modes offered by private persons.
<i>ContinuousModesEnumeration</i>	<i>walk / demandResponsive / replacementService</i>	Types of continuous modes (that run on any time, without a timetable).
<i>TransferModesEnumeration</i>	<i>walk / parkAndRide / bikeAndRide / carHire / bikeHire / protectedConnection / guaranteedConnection / remainInVehicle / changeWithinVehicle / checkIn / checkOut</i>	Types of transfer modes.
<i>PtModesEnumeration</i>	<i>all / unknown / air / bus / trolleyBus / tram / coach / rail / intercityRail / urbanRail / metro / water / cableway / funicular / taxi</i>	Classes of public transport modes (following TPEG pti_table 01).
<i>RailSubmodeEnumeration</i>	<i>unknown / undefined / local / highSpeedRail / suburbanRailway / regionalRail / interregionalRail / longDistance / international / sleeperRailService / nightRail / carTransportRailService / touristRailway / railShuttle / replacementRailService / specialTrain / crossCountryRail / rackAndPinionRailway</i>	Sub-classes of mode train (following TPEG pti_table 02).
<i>CoachSubmodeEnumeration</i>	<i>unknown / undefined / internationalCoach / nationalCoach / shuttleCoach / regionalCoach / specialCoach / sightseeingCoach / touristCoach / commuterCoach</i>	Sub-classes of mode coach (following TPEG pti_table 03).
<i>MetroSubmodeEnumeration</i>	<i>unknown / undefined / metro / tube / urbanRailway</i>	Sub-classes of mode underground (following TPEG pti_table 04).

<i>BusSubmodeEnumeration</i>	<i>unknown / undefined / localBus / regionalBus / expressBus / nightBus / postBus / specialNeedsBus / mobilityBus / mobilityBusForRegisteredDisabled / sightseeingBus / shuttleBus / schoolBus / schoolAndPublicServiceBus / railReplacementBus / demandAndResponseBus / airportLinkBus</i>	Sub-classes of mode bus (following TPEG pti_table 05).
<i>TramSubmodeEnumeration</i>	<i>unknown / undefined / cityTram / localTram / regionalTram / sightseeingTram / shuttleTram</i>	Sub-classes of mode tram (following TPEG pti_table 06).
<i>WaterSubmodeEnumeration</i>	<i>unknown / undefined / internationalCarFerry / nationalCarFerry / regionalCarFerry / localCarFerry / internationalPassengerFerry / nationalPassengerFerry / regionalPassengerFerry / localPassengerFerry / postBoat / trainFerry / roadFerryLink / airportBoatLink / highSpeedVehicleService / highSpeedPassengerService / sightseeingService / schoolBoat / cableFerry / riverBus / scheduledFerry / shuttleFerryService</i>	Sub-classes of water-borne modes (following TPEG pti_table 07).
<i>AirSubmodeEnumeration</i>	<i>unknown / undefined / internationalFlight / domesticFlight / intercontinentalFlight / domesticScheduledFlight / shuttleFlight / intercontinentalCharterFlight / internationalCharterFlight / roundTripCharterFlight / sightseeingFlight / helicopterService / domesticCharterFlight / SchengenAreaFlight / airshipService / shortHaulInternationalFlight / canalBarge</i>	Sub-classes of flying modes (following TPEG pti_table 08).
<i>TelecabinSubmodeEnumeration</i>	<i>unknown / undefined / telecabin / cableCar / lift / chairLift / dragLift / telecabinLink</i>	Sub-classes of mode telecabin (following TPEG pti_table 09).
<i>FunicularSubmodeEnumeration</i>	<i>unknown / funicular / allFunicularServices / undefinedFunicular</i>	Sub-classes of mode funicular (following TPEG pti_table 10).
<i>TaxiSubmodeEnumeration</i>	<i>unknown / undefined / communalTaxi / waterTaxi / railTaxi / bikeTaxi / blackCab / miniCab / allTaxiServices</i>	Sub-classes of mode taxi (following TPEG pti_table 11).

8.4.3.2 The subsequent sections describe the complex structures defined in OJP_ModesSupport. IndividualTransportOptionsStructure

Table 13 — Description of *IndividualTransportOptionsStructure*.

<i>IndividualTransportOptionsStructure</i>			<i>+Structure</i>	Individual transport modes and their usage limits as stated by the passenger.
	Mode	1:1	<i>IndividualModesEnumeration</i>	Mode of individual transport. When mode self-drive-car is used the connection to another mode will need a parking space for the car. Thus, this mode is a generalisation of all park&ride use cases. The mode others-drive-car, however, only needs a place to let a person be set down from a car.
	<i>MaxDistance</i>	0:1	<i>Distance</i>	Maximal distance in metres. If given, it restricts the maximal distance of routes with the given mode.
	<i>MaxDuration</i>	0:1	<i>xs:duration</i>	Maximal duration. If given, it restricts the maximal time of routes with the given mode.
	<i>MinDistance</i>	0:1	<i>Distance</i>	Minimal distance in metres. If given, it restricts the minimal distance of routes with the given mode.
	<i>MinDuration</i>	0:1	<i>xs:duration</i>	Minimal duration. If given, it restricts the minimal time of routes with the given mode.
	<i>Speed</i>	0:1	<i>OpenPercent</i>	Relative speed in percent. If given slows the standard speed (below 100) or fasten it (above 100).

8.4.3.3 PtSubmodeChoiceGroup

Table 14 — Description of *PtSubmodeChoiceGroup*.

<i>PtSubmodeChoiceGroup</i>				<i>+Group</i>	Group to select public transport submodes.
	<i>a</i>	<i>AirSubmode</i>	-0:1	<i>AirSubmodeEnumeration</i>	Air submodes.
	<i>b</i>	<i>BusSubmode</i>		<i>BusSubmodeEnumeration</i>	Bus submodes.
	<i>c</i>	<i>CoachSubmode</i>		<i>CoachSubmodeEnumeration</i>	Coach submodes.
	<i>d</i>	<i>FunicularSubmode</i>		<i>FunicularSubmodeEnumeration</i>	Funicular submodes.
	<i>e</i>	<i>MetroSubmode</i>		<i>MetroSubmodeEnumeration</i>	Underground submodes.
	<i>f</i>	<i>RailSubmode</i>		<i>RailSubmodeEnumeration</i>	Rail submodes.
	<i>g</i>	<i>TelecabinSubmode</i>		<i>TelecabinSubmodeEnumeration</i>	Telecabin submodes.
	<i>h</i>	<i>TramSubmode</i>		<i>TramSubmodeEnumeration</i>	Tram submodes.
	<i>i</i>	<i>WaterSubmode</i>		<i>WaterSubmodeEnumeration</i>	Water submodes.

8.4.3.4 ModeStructure

Table 15 — Description of *ModeStructure*.

<i>ModeStructure</i>			<i>+Structure</i>	Classification and naming of a mode.
<i>Mode</i>	PtMode	1:1	<i>PtModesEnumeration</i>	Defines the public transport mode.
<i>PtSubmodeChoice</i>	:::	-0:1	<i>PtSubmodeChoice</i>	Defines the public transport submode (see. 8.4.3.3)
	<i>Name</i>	0:1	<i>InternationalText</i>	Name of the mode.
	<i>ShortName</i>	0:1	<i>InternationalText</i>	Short name or acronym of the mode.
	<i>Description</i>	0:1	<i>InternationalText</i>	Additional text that further describes the mode.

8.4.3.5 PtModeFilterStructure

Table 16 — Description of *PtModeFilterStructure*.

<i>PtModeFilterStructure</i>			<i>+Structure</i>	List of public transport modes to include or exclude.
	<i>Exclude</i>	0:1	<i>xs:Boolean</i>	Whether modes in list are to be included or excluded from search. Default is excluded.
	<i>PtMode</i>	0:*	<i>PtModesEnumeration</i>	List of public transport modes to be included/excluded.
<i>PtSubmodeChoice</i>	:::	0:*	<i>PtSubmodeChoice</i>	List of public transport submodes to be included/excluded.

8.4.3.6 PrivateModeFilterStructure

Table 17 — Description of *PrivateModeFilterStructure*.

<i>PrivateModeFilterStructure</i>			<i>+Structure</i>	List of private mobility offers to include or exclude.
	<i>Exclude</i>	0:1	<i>xs:Boolean</i>	Whether modes in list are to be included or excluded from search. Default is excluded.
	<i>PrivateMode</i>	0:*	<i>PrivateModesEnumeration</i>	List of private mobility offers to be included/excluded.

8.4.4 OJP_Common

8.4.4.1 Simple Types

This schema file defines the following simple types:

Table 18 — List of simple type definitions in *OJP_Common.xsd*.

<i>ParticipantCodeType</i>	<i>xs:normalizedString</i>	Participant identifier.
<i>OperatorCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a transport operator.
<i>LineCodeType</i>	<i>xs:NMTOKEN</i>	Line identifier.
<i>DirectionCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a direction (typically outward or return).
<i>JourneyCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a timetable journey.
<i>VehicleCodeType</i>	<i>xs:NMTOKEN</i>	Vehicle identifier.
<i>FacilityCodeType</i>	<i>xs:NMTOKEN</i>	Facility identifier.
<i>OwnerCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a responsible organisation (owner).
<i>OperatingDayCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of an operating day.

A common understanding of identifiers of lines, transport operators etc. has to exist across system boundaries. Therefore a few agreements have to be put in place (for more details see section 8.2).

The subsequent sections describe the complex structures defined in *OJP_Common.xsd*.

8.4.4.2 ErrorMessageStructure

Table 19 — Description of *ErrorMessageStructure*.

<i>ErrorMessageStructure</i>			<i>+Structure</i>	Structured error messages.
	<i>Code</i>	1:1	<i>xs:normalizedString</i>	Code for the error situation.
	<i>Text</i>	0:1	<i>+InternationalText</i>	Description of the error situation.

8.4.4.3 PrivateCodeStructure

Table 20 — Description of *PrivateCodeStructure*.

<i>PrivateCodeStructure</i>			<i>+Structure</i>	Code within scope of a private (proprietary) referential system.
	<i>System</i>	1:1	<i>xs:NMTOKEN</i>	Code of the foreign referential system.
	<i>Value</i>	1:1	<i>xs:NMTOKEN</i>	Object code within this private/foreign system.

8.4.4.4 OperatorFilterStructure

Table 21 — Description of *OperatorFilterStructure*

<i>OperatorFilterStructure</i>			<i>+Structure</i>	Filter for in/exclusion of transport operators.
	<i>Exclude</i>	0:1	<i>xs:boolean</i>	Whether operators in list are to be included or excluded from search. Default is excluded.
	<i>OperatorRef</i>	0:*	<i>→Operator</i>	Reference to transport operator (see 8.4.4.1).

8.4.4.5 LineDirectionStructure

Table 22 — Description of *LineDirectionStructure*.

<i>LineDirectionStructure</i>			<i>+Structure</i>	Reference to a line, possibly narrowed down to a specific direction.
	<i>LineRef</i>	1:1	<i>→LineCode</i>	Reference to a line (see 8.4.4.1).
	<i>DirectionRef</i>	0:1	<i>→DirectionCode</i>	Reference to a direction of a line (see 8.4.4.1).

8.4.4.6 LineDirectionFilterStructure

Table 23 — Description of *LineDirectionFilterStructure*.

<i>LineDirectionFilterStructure</i>			<i>+Structure</i>	Filter for in/exclusion of lines (and directions).
	<i>Line</i>	1:*	<i>+LineDirection</i>	Reference to line or line/direction Linie (see 8.4.4.5).
	<i>Exclude</i>	0:1	<i>xs:boolean</i>	Whether lines in list are to included or excluded from search. Default is excluded.

8.4.4.7 SharingServiceStructure

Table 24 — Description of *SharingServiceStructure*.

<i>SharingServiceStructure</i>			+Structure	Structure for describing a mobility service with shared-service vehicles
	<i>OperatorRef</i>	1:1	→Operator	Identifier of the operator of the sharing service. (see 8.4.4.1).
	<i>Name</i>	0:1	xs:string	Public name of the service.
	<i>SharingModel</i>	0:1	singleStationBased / multipleStationBased / nonStationBased	Type of the sharing scheme.
<i>SharingServiceUsage</i>	<i>TimeBufferBefore</i>	0:1	xs:duration	Typical time a user will need to check in and get the vehicle ready for use.
	<i>TimeBufferAfter</i>	0:1	xs:duration	Typical time a user will need to lock the vehicle and check out.
	<i>InfoURL</i>	0:1	+WebLink	Link to web page providing more details about the service (see 8.4.2.4).

8.4.4.8 OperatingDaysStructure

Table 25 — Description of *OperatingDaysStructure*.

<i>OperatingDaysStructure</i>			+Structure	A list of operating days coded as bit pattern between a starting day and end day.
	<i>From</i>	1:1	xs:date	Start date of period.
	<i>To</i>	1:1	xs:date	End date of period.
	<i>Pattern</i>	1:1	ValidDayBitType	Bit pattern for operating days between start date and end date. The length of the pattern is equal to the number of days from start date to end date. A bit value of "1" indicates that an event actually happens on the day that is represented by the bit position.

8.4.4.9 WeekdayTimePeriodStructure

Table 26 — Description of *WeekdayTimePeriodStructure*.

<i>WeekdayTimePeriodStructure</i>			+Structure	Time period on a weekday.
	<i>Weekday</i>	0:1	Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday / PublicHoliday	Type of weekday.
	<i>StartTime</i>	1:1	xs:time	Start time of period.
	<i>Duration</i>	1:1	xs:duration	Time duration of period.

8.4.4.10 GeneralAttributeStructure

Table 27 — Description of *GeneralAttributeStructure*.

<i>GeneralAttributeStructure</i>			<i>+Structure</i>	Structured attribute classification with associated text.
	<i>Text</i>	1:1	<i>+InternationalText</i>	Text of the attribute to be shown to the user.
	<i>Code</i>	1:1	<i>xs:NMTOKEN</i>	Internal code of the attribute. Can be used for detection of double occurrences.
<i>AllFacilities</i>	:::	0:1	<i>+AllFacilitiesGroup</i>	TPEG classification (see 8.4.7.4).
	<i>Mandatory</i>	0:1	<i>xs:boolean</i>	Defines whether the attribute has to be shown to the user. Default value is <i>false</i> .
	<i>Importance</i>	0:1	<i>Percent</i>	Importance of the attribute (for prioritising multiple attributes).
	<i>InfoURL</i>	0:1	<i>xs:anyURI</i>	URL of additional information on this general attribute. If available, the whole attribute text has to be used as the marked link.

8.4.5 OJP_LocationSupport

The XML schema file OJP_LocationSupport.xsd defines a set of re-usable base types and structures for referencing and describing locations (stop places, stops points, localities, addresses and points of interest).

8.4.5.1 Simple Types

This schema file defines the following simple types:

Table 28 — List of simple type definitions in OJP_LocationSupport.xsd.

<i>StopPointCodeType</i>	<i>xs:normalizedString</i>	Identifier of a Scheduled Stop Point
<i>StopPlaceCodeType</i>	<i>xs:normalizedString</i>	Identifier of a Stop Place
<i>TopographicPlaceCodeType</i>	<i>xs:normalizedString</i>	Identifier of a Topographic Place
<i>PointOfInterestCodeType</i>	<i>xs:normalizedString</i>	Identifier of a Point Of Interest
<i>AddressCodeType</i>	<i>xs:normalizedString</i>	Identifier of an Address

The identifiers of stop places, stop points etc. have to meet some requirements in order to be interpretable across system boundaries. More details on this topic can be found in chapter 8.2.

The subsequent sections describe the complex structures defined in OJP_LocationSupport.xsd.

8.4.5.2 StopPointStructure

Table 29 — Description of *StopPointStructure*.

<i>StopPointStructure</i>			<i>+Structure</i>	Complete model of a stop point.
<i>StopPoint</i>	<i>StopPointRef</i>	1:1	<i>→StopPoint</i>	Reference to a Stop Point (see. 8.4.5.1).
	<i>StopPointName</i>	1:1	<i>+InternationalText</i>	Name or description of stop point for use in passenger information.
	<i>NameSuffix</i>	0:1	<i>+InternationalText</i>	Additional description of the stop point that may be appended to the name if enough space is available. Eg: "opposite main entrance".

	<i>PlannedBay</i>	0:1	<i>+InternationalText</i>	Name of the bay where to board/alight from the vehicle. According to planned timetable.
	<i>EstimatedBay</i>	0:1	<i>+InternationalText</i>	Name of the bay where to board/alight from the vehicle. As to the latest realtime status.
	<i>PrivateCode</i>	0:*	<i>+PrivateCode</i>	Code of this stop point in private/foreign/proprietary coding schemes (see. 8.4.4.3).
	<i>ParentRef</i>	0:1	<i>→StopPlace</i>	Reference to the stop place to which this stop point belongs (see 8.4.5.1).
	<i>TopographicPlaceRef</i>	0:1	<i>→TopographicPlace</i>	Reference to the Topographic Place to which this Scheduled Stop Point belongs (see 8.4.5.1).
<i>StopAttributes</i>	<i>WheelchairAccessible</i>	0:1	<i>xs:boolean</i>	Whether this stop is accessible for wheelchair users. Default is <i>false</i> .
	<i>Lighting</i>	0:1	<i>xs:boolean</i>	Whether this stop is lit. Default is <i>false</i> .
	<i>Covered</i>	0:1	<i>xs:boolean</i>	Whether this stop offers protection from weather conditions like rain, snow, storm etc. Default is <i>false</i> .

8.4.5.3 StopPlaceStructure

Table 30 — Description of *StopPlaceStructure*.

<i>StopPlaceStructure</i>			<i>+Structure</i>	Complete model of a stop place.
<i>StopPlace</i>	<i>StopPlaceRef</i>	1:1	<i>→StopPlace</i>	Reference to a Stop Place (see 8.4.5.1).
	<i>StopPlaceName</i>	1:1	<i>+InternationalText</i>	Name of this stop place for use in passenger information.
	<i>NameSuffix</i>	0:1	<i>+InternationalText</i>	Additional description of the stop place that may be appended to the name if enough space is available. Eg: "Exhibition Centre".
	<i>PrivateCode</i>	0:*	<i>+PrivateCode</i>	Code of this stop place in private/foreign/proprietary coding schemes (see. 8.4.4.3).
	<i>TopographicPlaceRef</i>	0:1	<i>→TopographicPlace</i>	Reference to the Topographic Place to which this STOP PLACE belongs (see 8.4.5.1).
<i>StopAttributes</i>	<i>WheelchairAccessible</i>	0:1	<i>xs:boolean</i>	Whether this stop place is accessible for wheelchair users. Default is <i>false</i> .
	<i>Lighting</i>	0:1	<i>xs:boolean</i>	Whether this stop place is lit. Default is <i>false</i> .
	<i>Covered</i>	0:1	<i>xs:boolean</i>	Whether this stop place offers protection from weather conditions like rain, snow, storm etc. Default is <i>false</i> .

8.4.5.4 TopographicPlaceStructure

Table 31 — Description of *TopographicPlaceStructure*.

<i>TopographicPlaceStructure</i>			<i>+Structure</i>	Model of a TopographicPlace
	<i>TopographicPlaceCode</i>	1:1	<i>→TopographicPlace</i>	Identifier of the Topographic Place (see 8.4.5.1).
	<i>TopographicPlaceName</i>	1:1	<i>+InternationalText</i>	Name or description of Topographic Place for use in passenger information.
	<i>PrivateCode</i>	0:*	<i>+PrivateCode</i>	Code of this Topographic Place in private/foreign/proprietary coding schemes.
	<i>ParentRef</i>	0:1	<i>→TopographicPlace</i>	Reference to a parent Topographic Place (to model, for example, the relation of a town district to the town itself). (see 8.4.5.1).
<i>Zone</i>	<i>Points</i>	3:*	<i>+GeoPosition</i>	Area covered by the Topographic Place described as a polygon.

8.4.5.5 PointOfInterestStructure

Table 32 — Description of *PointOfInterestStructure*.

<i>PointOfInterestStructure</i>			<i>+Structure</i>	Model of a Point of Interest (POI).
	<i>PointOfInterestCode</i>	1:1	→ <i>PointOfInterest</i>	Identifier of this Point of Interest.
	<i>PointOfInterestName</i>	1:1	+ <i>InternationalText</i>	Name or description of point of interest for use in passenger information.
	<i>NameSuffix</i>	0:1	+ <i>InternationalText</i>	Additional description of the point of interest that may be appended to the name if enough space is available. Eg: "Exhibition Centre".
	<i>PointOfInterestCategory</i>	0:*	+ <i>PointOfInterestCategory</i>	Categories associated with this point of interest. See 8.4.5.6. If there are more than one they are listed in descending relevance.
	<i>PrivateCode</i>	0:*	+ <i>PrivateCode</i>	Code of this point of interest in private/foreign/proprietary coding schemes.
	<i>TopographicPlaceRef</i>	0:1	→ <i>TopographicPlace</i>	Reference to the Topographic Place to which this Point Of Interest belongs (see 8.4.5.1).

8.4.5.6 PointOfInterestCategoryStructure

Table 33 — Description of *PointOfInterestCategoryStructure*.

<i>PointOfInterestCategoryStructure</i>			<i>+Structure</i>	Model of a point-of-interest category defined by key-value-pairs.
	<i>OsmTag</i>	1:*	+ <i>OsmTag</i>	List of tags (key-value pairs) as used in OpenStreetMap. ⁴ See 0.
	<i>PointOfInterestClassification</i>	1:*	<i>xs:string</i>	Classification of the POI (when it is not from OSM). The codification of the classification Id may include the codification source (for example "IGN:[<i>classificationCode</i>]")

8.4.5.7 OsmTagStructure

Table 34 — Description of *OsmTagStructure*.

<i>OsmTagStructure</i>			<i>+Structure</i>	OpenStreetMap tag structure.
	<i>Tag</i>	1:1	<i>xs:NMTOKEN</i>	Name of the OpenStreetMap tag (eg: amenity, leisure, tourism, bike, ...)
	<i>Value</i>	1:1	<i>xs:NMTOKEN</i>	Value of the OpenStreetMap tag (eg: yes, hostel, charging_station, ...)

⁴ http://wiki.openstreetmap.org/wiki/Map_Features

8.4.5.8 PointOfInterestFilterStructure

Table 35 — Description of *PointOfInterestFilterStructure*.

<i>PointOfInterestFilterStructure</i>			<i>+Structure</i>	Structure for filtering by point-of-interest categories
	<i>Exclude</i>	0:1	<i>xs:boolean</i>	Specifies whether points of interest should be searched only within the given categories (<i>Exclude=false</i>) or do not meet any of the given categories (<i>Exclude=true</i>). Default is <i>false</i> .
	<i>PointOfInterestCategory</i>	1:*	<i>+PointOfInterestCategory</i>	List of point-of-interest categories. See 8.4.5.6. If the list contains more than one category the search for points of interest will combine them by a logical "OR" in the case of <i>Exclude=false</i> and by a logical "AND" in the case of <i>Exclude=true</i> .

8.4.5.9 AddressStructure

Table 36 — Description of *AddressStructure*.

<i>AddressStructure</i>			<i>+Structure</i>	Model of an Address.
	<i>AddressCode</i>	1:1	<i>→Address</i>	Identifier of this Address (see 8.4.5.1).
	<i>PrivateCode</i>	0:*	<i>+PrivateCode</i>	Code of this Address in private/foreign/proprietary coding schemes.
	<i>AddressName</i>	1:1	<i>+InternationalText</i>	Name or description of Address for use in passenger information.
	<i>NameSuffix</i>	0:1	<i>+InternationalText</i>	Additional description of the Address that may be appended to the name if enough space is available. eg.: "Crossing with Peterstraße".
<i>AddressDetail</i>	<i>CountryName</i>	0:1	<i>xs:string</i>	Country of the Address.
	<i>PostCode</i>	0:1	<i>xs:string</i>	Postal code of the Address.
	<i>TopographicPlaceName</i>	0:1	<i>xs:string</i>	Topographic Place name of the Address. If set it should at least contain the city name.
	<i>TopographicPlaceRef</i>	0:1	<i>→TopographicPlace</i>	Reference to the Topographic Place to which this Address belongs (see 8.4.5.1).
	<i>Street</i>	0:1	<i>xs:string</i>	Street name of the Address.
	<i>HouseNumber</i>	0:1	<i>xs:string</i>	House number of the address. If none is given, either a crossing street can be given, or the whole street is meant.
	<i>CrossRoad</i>	0:1	<i>xs:string</i>	Crossing. This can be used to be more specific without using house numbers.

8.4.5.10 PlaceStructure

Table 37 — Description of *PlaceStructure*.

<i>PlaceStructure</i>				<i>+Structure</i>	Base model for all types of locations (stop point, stop place, geo location, TopographicPlace, POI or address).
	<i>a</i>	<i>StopPoint</i>	-0:1	<i>+StopPoint</i>	Model of a stop point (see 8.4.5.2).
	<i>b</i>	<i>StopPlace</i>		<i>+StopPlace</i>	Model of a stop place (see 8.4.5.3)
	<i>c</i>	<i>TopographicPlace</i>		<i>+TopographicPlace</i>	Model of a TopographicPlace (see 8.4.5.4).
	<i>d</i>	<i>PointOfInterest</i>		<i>+PointOfInterest</i>	Model of a POI (see 8.4.5.5).
	<i>e</i>	<i>Address</i>		<i>+Address</i>	Model of an address (see 8.4.5.9).
	<i>LocationName</i>		1:1	<i>+InternationalText</i>	Public name of the location.
	<i>GeoPosition</i>		1:1	<i>+GeoPosition</i>	Geographic position (see 8.4.2.3).

	<i>Attribute</i>	0:*	<i>+GeneralAttribute</i>	Attributes associated with this location. (see 8.4.4.10).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.4.5.11 PlaceRefStructure

Table 38 — Description of *PlaceRefStructure*.

PlaceRefStructure				+Structure	Reference to a stop point, stop place, address, POI or TopographicPlace.
	a	StopPointRef	-1:1	→StopPoint	Reference to a stop point (see 8.4.5.1).
	b	StopPlaceRef		→StopPlace	Reference to a stop place (see 8.4.5.1).
	c	GeoPosition		+GeoPosition	Geographic position (see 8.4.2.3).
	d	TopographicPlaceRef		→TopographicPlace	Reference to a TopographicPlace (see 8.4.5.1).
	e	PointOfInterestRef		→PointOfInterest	Reference to a point of interest (see 8.4.5.1).
	f	AddressRef		→Address	Reference to an address (see 8.4.5.1).
	LocationName		1:1	+InternationalText	Public name of the location.

8.4.5.12 ExchangePointsFilterStructure

Table 39 — Description of *ExchangePointsFilterStructure*.

<i>ExchangePointsFilterStructure</i>			<i>+Structure</i>	Filter for exchangePoints between journey planning areas.
	<i>OnlyExchangePoints</i>	0:1	<i>xs:boolean</i>	Whether the response should return exchange points only. Default is <i>true</i> .
	<i>AdjacentSystem</i>	0:*	→ <i>ParticipantCode</i>	Unique identifier of one or more adjacent journey planning systems to which the exchange points should be retrieved (see 8.2.11). If none is given the exchange points to all adjacent systems should be returned.

8.4.6 OJP_JourneySupport

The XML schema file *OJP_JourneySupport.xsd* defines structures describing journeys on public transport vehicles. This includes vehicle journeys, departure and arrival events at stops as well as vehicle movements along the routes.

The subsequent sections describe the complex structures defined in *OJP_JourneySupport.xsd*.

8.4.6.1 ServiceViaPointStructure

Table 40 — Description of *ServiceViaPointStructure*.

<i>ServiceViaPointStructure</i>			<i>+Structure</i>	Stop point as via point (in service pattern).
StopPoint	StopPointRef	1:1	→StopPoint	Via point. Reference to a stop point (see 8.4.5.1).
	StopPointName	1:1	+InternationalText	Name or description of stop point for use in passenger information.
	NameSuffix	0:1	+InternationalText	Additional description of the stop point that may be appended to the name if enough space is available. eg: "opposite main entrance".
	PlannedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. According to planned timetable.
	EstimatedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. As to the latest realtime status.
	DisplayPriority	0:1	Priority	Priority of this via point to be displayed when space is limited.

8.4.6.2 TripViaStructure

Table 41 — Description of *TripViaStructure*.

<i>TripViaStructure</i>			<i>+Structure</i>	TripVia restrictions for a trip.
	ViaPoint	1:1	+PlaceRef	Reference to a location as via point (see 8.4.5.11)
	DwellTime	0:*	xs:duration	Duration the passenger wants to stay at the via location. Default is 0.

8.4.6.3 DatedJourneyGroup

Table 42 — Description of *DatedJourneyGroup*.

<i>DatedJourneyGroup</i>			<i>+Structure</i>	Structure for a vehicle journey at a specific date.
DatedJourney	OperatingDayRef	1:1	→OperatingDay	Reference to an Operating Day (see 8.4.4.1).
	VehicleRef	0:1	→Vehicle	Reference to a vehicle (see 8.4.4.1).
ServiceJourney	JourneyRef	1:1	→Journey	Reference to a journey (see 8.4.4.1).
LineIdentity	LineRef	1:1	→Line	Reference to a line (see 8.4.4.1).
	DirectionRef	1:1	→Direction	Reference to a direction (see 8.4.4.1).
Service	Mode	1:1	+Mode	Vehicle mode (see 8.4.3).
	PublishedLineName	1:1	+InternationalText	Line name or service description as known to the public, eg: "512", "S8" or "Circle Line".
	OperatorRef	0:1	→Operator	Reference to an operator. (see 8.4.4.1).
	RouteDescription	0:1	+InternationalText	Descriptive text for a route, eg: "Airport via City Centre".
	Via	0:*	+ServiceViaPoint	Via points of the service that may help identify the vehicle to the public (see 8.4.6.1).
	Attribute	0:*	+GeneralAttribute	Note or service attribute (see 8.4.4.10).

8.4.6.4 PrivateServiceGroup

Table 43 — Description of *PrivateServiceGroup*.

<i>PrivateServiceGroup</i>			<i>+Structure</i>	Privately offered mobility option..
	<i>JourneyRef</i>	0:1	<i>→Journey</i>	Reference to a journey (see 8.4.4.1).
	<i>PrivateMode</i>	1:1	<i>PrivateModesEnumeration</i>	Classification of mode (see 8.4.3.6).
	<i>OrganisationRef</i>	0:1	<i>→Operator</i>	Reference to an operator. (see 8.4.4.1).
	<i>InfoUrl</i>	0:1	<i>+WebLink</i>	Link to web resource providing more details on service (see 8.4.2.4).

8.4.6.5 DatedJourneyStructure

Table 44 — Description of *DatedJourneyStructure*.

<i>DatedJourneyStructure</i>				<i>+Structure</i>	Structure for a vehicle journey on a specific date.
	<i>a</i>	...	-1:1	<i>+DatedJourneyGroup</i>	Description of service journey on a specific date (see 8.4.6.3).
	<i>b</i>	...	-1:1	<i>+PrivateServiceGroup</i>	Description of privately offered mobility service (see 8.4.6.4).
<i>ServiceOrigin</i>	<i>OriginStopPointRef</i>	0:1	<i>→StopPoint</i>		First stop of the vehicle journey; origin stop point (see 8.4.5.1).
	<i>OriginText</i>	0:1	<i>+InternationalText</i>		Label for first stop.
<i>ServiceDestination</i>	<i>DestinationStopPointRef</i>	0:1	<i>→StopPoint</i>		Last stop of vehicle journey; destination stop point (see 8.4.5.1).
	<i>DestinationText</i>	1:1	<i>+InternationalText</i>		Label for last stop.
<i>ServiceStatus</i>	<i>Unplanned</i>	0:1	<i>xs:boolean</i>		Whether this DatedVehicleJourney is an additional one that has not been planned. Default is <i>false</i> .
	<i>Cancelled</i>	0:1	<i>xs:boolean</i>		Whether this DatedVehicleJourney is cancelled and will not be run. Default is <i>false</i> .
	<i>Deviation</i>	0:1	<i>xs:boolean</i>		Whether this DatedVehicleJourney deviates from the planned service pattern. Default is <i>false</i> .
	<i>Occupancy</i>	0:1	<i>manySeatsAvailable</i> <i>fewSeatsAvailable</i> <i>noSeatsAvailable</i> <i>standingAvailable</i> / <i>full</i>		How full the vehicle is. If omitted, not known.
	<i>BookingArrangements</i>	0:1	<i>+BookingArrangementContainer</i>		Container with information on booking possibilities for this service (see 8.4.10.5).
	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>		Reference to situation message. Message details might be found in response context or through other communication channels (see 8.4.8.2).

8.4.6.6 TripLocationStructure

Table 45 — Description of *TripLocationStructure*.

<i>TripLocationStructure</i>			+Structure	Location of a passenger currently traveling in a vehicle.
	OperatingDayRef	1:1	→OperatingDay	Reference to an Operating Day (see 8.4.4.1).
	JourneyRef	1:1	→Journey	Reference to a journey (see 8.4.4.1).
LineIdentity	LineRef	1:1	→Line	Reference to a line (see 8.4.4.1).
	DirectionRef	1:1	→Direction	Reference to a direction (see 8.4.4.1).

8.4.6.7 CallAtStopStructure

Table 46 — Description of *CallAtStopStructure*.

<i>CallAtStopStructure</i>			+Structure	The calling of a vehicle journey at a specific stop.
StopPoint	StopPointRef	1:1	→StopPoint	Reference to a stop point (see 8.4.5.1).
	StopPointName	1:1	+InternationalText	Name or description of stop point for use in passenger information.
	NameSuffix	0:1	+InternationalText	Additional description of the stop point that may be appended to the name if enough space is available. Eg: "opposite main entrance".
	PlannedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. According to planned timetable.
	EstimatedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. As to the latest realtime status.
ServiceArrival	TimetabledTime	1:1	xs:dateTime	Arrival time according to planned timetable.
	RecordedAtTime	0:1	xs:dateTime	Arrival time at the stop point as it was recorded.
	EstimatedTime	0:1	xs:dateTime	Expected/estimated arrival time at the stop point.
	EstimatedTimeLow	0:1	xs:dateTime	Estimated lower limit for arrival time.
	EstimatedTimeHigh	0:1	xs:dateTime	Estimated upper limit for arrival time.
ServiceDeparture	TimetabledTime	1:1	xs:dateTime	Departure time according to planned timetable.
	RecordedAtTime	0:1	xs:dateTime	Departure time at the stop point as it was recorded.
	EstimatedTime	0:1	xs:dateTime	Expected/estimated departure time at the stop point.
	EstimatedTimeLow	0:1	xs:dateTime	Estimated lower limit for departure time.
	EstimatedTimeHigh	0:1	xs:dateTime	Estimated upper limit for departure time.
StopCallStatus	Order	0:1	xs:positiveInteger	Sequence number of this stop in the service pattern of the journey.
	RequestStop	0:1	xs:boolean	The vehicle journey calls at this stop only on demand. Default is <i>false</i> .
	UnplannedStop	0:1	xs:boolean	This stop has not been planned by the planning department. Default is <i>false</i> .
	NotServedStop	0:1	xs:boolean	The vehicle will not call at this stop despite earlier planning. Default is <i>false</i> .
	SituationFullRef	0:*	+SituationFullRef	Reference to situation message. Message details might be found in response context or through other communication channels (see 8.4.8.2).

8.4.6.8 DatedCallAtLocationStructure

Table 47 — Description of *DatedCallAtLocationStructure*.

<i>DatedCallAtLocationStructure</i>			+Structure	Vehicle call at a general location on a specific date.
<i>DatedJourneyRef</i>	JourneyRef	1:1	→Journey	Reference to a DATED VEHICLE JOURNEY (see 8.4.4.1).
	OperatingDayRef	1:1	→OperatingDay	Reference to an Operating Day (see 8.4.4.1).
	CallLocation	1:1	+PlaceRef	More general location for a call than stop points. May be used with flexible services or "Area Dial-A-Ride" (see 8.4.5.11).
<i>ServiceArrival</i>	TimetabledTime	1:1	xs:dateTime	Arrival time according to planned timetable.
	RecordedAtTime	0:1	xs:dateTime	Arrival time at the stop point as it was recorded.
	EstimatedTime	0:1	xs:dateTime	Expected/estimated arrival time at the stop point.
	EstimatedTimeLow	0:1	xs:dateTime	Estimated lower limit for arrival time.
	EstimatedTimeHigh	0:1	xs:dateTime	Estimated upper limit for arrival time.
<i>ServiceDeparture</i>	TimetabledTime	1:1	xs:dateTime	Departure time according to planned timetable.
	RecordedAtTime	0:1	xs:dateTime	Departure time at the stop point as it was recorded.
	EstimatedTime	0:1	xs:dateTime	Expected/estimated departure time at the stop point.
	EstimatedTimeLow	0:1	xs:dateTime	Estimated lower limit for departure time.
	EstimatedTimeHigh	0:1	xs:dateTime	Estimated upper limit for departure time.
<i>StopCallStatus</i>	Order	0:1	xs:positiveInteger	Sequence number of this stop in the service pattern of the journey.
	RequestStop	0:1	xs:boolean	The vehicle journey calls at this stop only on demand. Default is <i>false</i> .
	UnplannedStop	0:1	xs:boolean	This stop has not been planned by the planning department. Default is <i>false</i> .
	NotServicedStop	0:1	xs:boolean	The vehicle will not call at this stop despite earlier planning. Default is <i>false</i> .

8.4.6.9 ContinuousServiceStructure

Table 48 — Description of *ContinuousServiceStructure*.

ContinuousServiceStructure				+Structure	A passenger movement on a continuous, non-timetabled service.
	a	ContinuousMode	- 1:1	walk / demandResponsive / replacementService	Continuous transport mode (see 8.4.3.1).
	b	IndividualMode		walk / cycle / taxi / self-drive-car / others-drive-car / motorcycle / truck	Individual transport mode (see 8.4.3.1).
	a	⋮	- 0:1	+DatedJourneyGroup	Description of a public transport service at a specific date (see 8.4.6.3).
	b	SharingService		+SharingService	Description of a shared mobility offer (see 8.4.4.7).
DatedJourney	OperatingDay		1:1	→OperatingDay	Reference to an Operating Day (see 8.4.4.1).
	VehicleRef		0:1	→Vehicle	Reference to a vehicle (see 8.4.4.1).
ServiceJourney	JourneyRef		1:1	→Journey	Reference to a journey (see 8.4.4.1).
LineIdentity	LineRef		1:1	→Line	Reference to a line (see 8.4.4.1).
	DirectionRef		1:1	→Direction	Reference to a direction (see 8.4.4.1).

<i>Service</i>	Mode	1:1	+Mode	Vehicle mode (see 8.4.3).
	PublishedLineName	1:1	<i>InternationalText</i>	Line name or service description as known to the public, eg: "512", "S8" or "Circle Line".
	<i>OperatorRef</i>	0:1	<i>→Operator</i>	Reference to an operator. (see 8.4.4.1).
	<i>RouteDescription</i>	0:1	<i>InternationalText</i>	Descriptive text for a route, eg: "Airport via City Centre".
	<i>Via</i>	0:*	<i>+ServiceViaPoint</i>	Via points of the service that may help identify the vehicle to the public (see 8.4.6.1).
	<i>Attribute</i>	0:*	<i>+GeneralAttribute</i>	Note or service attribute (see 8.4.4.10).
<i>ServiceOrigin</i>	<i>OriginStopPointRef</i>	0:1	<i>→StopPoint</i>	First stop of the vehicle journey; origin stop point (see 8.4.5.1).
	<i>OriginText</i>	0:1	<i>InternationalText</i>	Label for first stop.
<i>ServiceDestination</i>	<i>DestinationStopPointRef</i>	0:1	<i>→StopPoint</i>	Last stop of vehicle journey; destination stop point (see 8.4.5.1).
	<i>DestinationText</i>	0:1	<i>InternationalText</i>	Label for last stop.
	<i>BookingArrangements</i>	0:1	<i>+BookingArrangementsContainer</i>	Container with information on booking possibilities for this service (see 8.4.10.5).
	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>	Reference to situation message. Message details might be found in response context or through other communication channels (see 8.4.8.2).

8.4.6.10 VehiclePositionStructure

Table 49 — Description of *VehiclePositionStructure*.

<i>VehiclePositionStructure</i>			+Structure	Geographical and logical position of a vehicle.
	<i>GeoPosition</i>	0:1	<i>+GeoPosition</i>	Geographic position of vehicle (see 8.4.2.3).
	<i>Progress</i>	0:1	<i>Not yet operated / Operation finished / At stop / Between stops</i>	Logical progress of vehicle relative to service pattern.
	<i>Bearing</i>	0:1	<i>AbsoluteBearing</i>	Bearing in compass degrees in which vehicle is heading (see 8.4.2.1).
	<i>ProgressBetweenStops</i>	0:1	<i>+ProgressBetweenStops</i>	Provides information about the progress of the vehicle along its current link. This is the link from the previously visited stop to the current position (see 8.4.6.11).

8.4.6.11 ProgressBetweenStopsStructure

Table 50 — Description of *ProgressBetweenStopsStructure*.

<i>ProgressBetweenStopsStructure</i>			+Structure	Vehicle position between the last visited stop and the current position.
	<i>LinkDistance</i>	0:1	<i>Distance</i>	The total distance in metres between the previous stop and the next stop.
	<i>Percentage</i>	0:1	<i>Percent</i>	Percentage along link that vehicle has travelled.

8.4.6.12 PlaceContextStructure

Table 51 — Description of *PlaceContextStructure*.

<i>PlaceContextStructure</i>				<i>+Structure</i>	A location and access to it by individual transport options.
	<i>a</i>	<i>PlaceRef</i>	-1:1	<i>+PlaceRef</i>	Spatial location (see 8.4.5.11)
	<i>b</i>	<i>TripLocation</i>		<i>+TripLocation</i>	Location within a (moving) vehicle (see 8.4.6.60).
	<i>a</i>	<i>DepArrTime</i>	-0:1	<i>xs:dateTime</i>	Target departure or arrival time at the location specified by <i>PlaceRef</i> or <i>TripLocation</i> .
	<i>b</i>	<i>TimeAllowance</i>		<i>xs:duration</i>	Extra time needed before reaching/after leaving this location.
	<i>IndividualTransportOptions</i>		0:*	<i>+IndividualTransportOptions</i>	Options stated by the user how he/she could access/leave the location by individual transport (see 8.4.3.2).

PlaceContextStructure type elements are mainly used to describe the origin/destination context of a passenger. For example elements of this type define the start and end location within the journey planning service (see chapter 8.7). There the implementation of the search algorithm is responsible for itself to map the location context (eg: a coordinate) to the internal data structures (eg: nodes and edges) of the routing graph.

For this purpose the *IndividualTransportOptions* specify the options how the user may access/leave public transport at the indicated stops. This is by walk as a default. But bike, car and taxi could be considered as well. When selecting the bike the user has to make a choice for option of carrying the bike on public transport. This option could lead to different journey planning results. When selecting the car the user has to choose between self-drive and others-drive. In the first case journey planning has to include the path to or from a parking place. In the second case, however, a place to stop and get into or out of the car would be sufficient.

8.4.6.13 AbstractResponseContextStructure

Table 52 — Description of *AbstractResponseContextStructure*.

<i>AbstractResponseContextStructure</i>			<i>+Structure</i>	Abstract structure providing response contexts related to journeys.
<i>Locations</i>	<i>Place</i>	0:*	<i>+Place</i>	Container for location objects (see 8.4.5).
<i>Situations</i>	<i>Situation</i>	0:*	<i>+siri:PtSituationElement</i>	Container for SIRI SX situation objects (see 8.4.8.1).

8.4.6.14 LegAttributeStructure

Table 53 — Description of *LegAttributeStructure*.

<i>LegAttributeStructure</i>			<i>+Structure (derived from GeneralAttributeStructure, see 8.4.4.10)</i>	Attributes that are not valid on the whole service, but only on part of the VEHICLE JOURNEY.
	<i>FromStopOrderNumber</i>	0:1	<i>xs:positiveInteger</i>	The attribute is valid from the stop point with this sequence number within the leg. If missing it is valid from the beginning of the leg.
	<i>ToStopOrderNumber</i>	0:1	<i>xs:positiveInteger</i>	The attribute is valid to the stop point (inclusively) with this sequence number within the leg. If missing it is valid to the end of the leg.

8.4.6.15 LegTrackStructure

Table 54 — Description of *LegTrackStructure*.

<i>LegTrackStructure</i>			<i>+Structure</i>	The sequence of tracks a journey leg travels along.
	<i>TrackSection</i>	1:*	<i>+TrackSection</i>	One or more track sections that build the journey leg. (See 8.4.6.16).

8.4.6.16 TrackSectionStructure

Table 55 — Description of *TrackSectionStructure*.

<i>TrackSectionStructure</i>			<i>+Structure</i>	A piece of a trip. A <i>TripLeg</i> may consist of multiple <i>TrackSections</i> . Describes the geographic embedding.
	<i>TrackStart</i>	0:1	<i>+PlaceRef</i>	Start location of this track. (See 8.4.5.11).
	<i>TrackEnd</i>	0:1	<i>+PlaceRef</i>	End location of this track. (See 8.4.5.11).
<i>LinkProjection</i>	Position	2:*	<i>+GeoPosition</i>	Geographic projection as polyline. (See 8.4.2.3). (An oriented correspondence from one LINK of a source layer, onto an entity in a target layer)
	<i>RoadName</i>	0:1	<i>xs:string</i>	Name of the road this track section is attached to.
	<i>Duration</i>	0:1	<i>xs:duration</i>	Duration the passenger needs to travel through this track section.
	<i>Length</i>	0:1	<i>LengthType</i>	Length of this track section.
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.4.7 OJP_FacilitySupport

The XML schema file *OJP_FacilitySupport.xsd* provides structures from the SIRI FM service for re-use. They support transmitting information about station and vehicle facilities. The definitions within this file aim at two objectives: a) to encapsulate the import of SIRI FM structures at one single place within OJP and b) to create a level of abstraction that allows for further enhancements without the need to change the SIRI definitions.

The subsequent sections describe the complex structures defined in *OJP_FacilitySupport.xsd*.

8.4.7.1 siri:CommonFacilityGroup

The XML schema group *CommonFacilityGroup* is defined in the SIRI schema file *siri_facilities-v1.2.xsd*. It is noted here only for reasons of completeness and ease of understanding.

Table 56 — Description of *siri:CommonFacilityGroup*.

<i>siri:CommonFacilityGroup</i>			<i>+Group</i>	Classification of common facility and infrastructure properties (according to TPEG pti_table 23).
	<i>FareClassFacility</i>	0:*	<i>unknown firstClass secondClass thirdClass economyClass businessClass</i>	Classification of fare classes.

	<i>TicketingFacility</i>	0:*	<i>unknown ticketMachines ticketOffice ticketOnDemandMachines ticketSales mobileTicketing ticketCollection centralReservations localTickets nationalTickets internationalTickets</i>	Classification of ticketing facilities.
	<i>NuisanceFacility</i>	0:*	<i>unknown smoking noSmoking mobilePhoneUseZone mobilePhoneFreeZone</i>	Classification of nuisance facilities.
	<i>MobilityFacility</i>	0:*	<i>unknown suitableForWheelChairs lowFloor boardingAssistance stepFreeAccess tactilePlatformEdges onboardAssistance unaccompaniedMinorAssistance audioInformation visualInformation displaysForVisuallyImpaired audioForHearingImpaired</i>	Classification of mobility facilities.
	<i>PassengerInformationFacility</i>	0:*	<i>unknown nextStopIndicator stopAnnouncements passengerInformationDisplay audioInformation visualInformation tactilePlatformEdges tactileInformation walkingGuidance journeyPlanning lostFound informationDesk interactiveKiosk-Display printedPublicNotice</i>	Classification of passenger information facilities.
	<i>PassengerCommsFacility</i>	0:*	<i>unknown faccomms_1 passengerWifi telephone audioServices videoServices businessServices internet postoffice letterbox</i>	Classification of passenger communication facilities.
	<i>RefreshmentFacility</i>	0:*	<i>unknown restaurantService snacksService trolley bar foodNotAvailable beveragesNotAvailable bistro foodVendingMachine beverageVendingMachine</i>	Classification of refreshment facilities.
	<i>AccessFacility</i>	0:*	<i>unknown lift escalator travelator ramp stairs shuttle narrowEntrance barrier palletAccess_lowFloor validator</i>	Classification of access facilities.
	<i>SanitaryFacility</i>	0:*	<i>unknown toilet noToilet shower wheelchairAccessToilet babyChange</i>	Classification of sanitary facilities.
	<i>LuggageFacility</i>	0:*	<i>unknown bikeCarriage baggageStorage leftLuggage portorage baggageTrolleys</i>	Classification of luggage facilities.

8.4.7.2 siri:StopFacilityGroup

The XML schema group *StopFacilityGroup* is defined in the SIRI schema file *siri_facilities-v1.2.xsd*. It is noted here only for reasons of completeness and ease of understanding.

Table 57 — Description of *siri:StopFacilityGroup*.

<i>siri:StopFacilityGroup</i>			+Group	Classification of station facilities (according to TPEG pti_table 23).
<i>CommonFacilityGroup</i>	:::	0:*	<i>siri:CommonFacilityGroup</i>	Common facilities (see 8.4.7.1).
	<i>AssistanceFacility</i>	0:*	<i>unknown / police / firstAid / sosPoint / specificAssistance / unaccompaniedMinorAssistance / boardingAssistance</i>	Classification of assistance facilities.
	<i>HireFacility</i>	0:*	<i>unknown / carHire / motorCycleHire / cycleHire / taxi / recreationDeviceHire</i>	Classification of hire facilities.

8.4.7.3 *siri:ServiceFacilityGroup*

The XML schema group *ServiceFacilityGroup* is defined in the SIRI schema file *siri_facilities-v1.2.xsd*. It is noted here only for reasons of completeness and ease of understanding.

Table 58 — Description of *siri:ServiceFacilityGroup*.

<i>siri:ServiceFacilityGroup</i>			+Group	Classification of vehicle and service facilities (according to TPEG pti_table 23).
<i>CommonFacilityGroup</i>	:::	0:*	<i>siri:CommonFacilityGroup</i>	Common facilities (see 8.4.7.1).
	<i>AccommodationFacility</i>	0:*	<i>unknown / sleeper / couchette / specialSeating / freeSeating / recliningSeats / babyCompartment / familyCarriage</i>	Classification of accommodation facilities.

8.4.7.4 *siri:AllFacilitiesGroup*

The XML schema group *AllFacilitiesGroup* is defined in the SIRI schema file *siri_facilities-v1.2.xsd*. It is noted here only for reasons of completeness and ease of understanding.

Table 59 — Description of *siri:AllFacilitiesGroup*.

<i>siri:AllFacilitiesGroup</i>			+Group	Comprehensive group including all types of facilities (according to TPEG pti_table 23).
<i>ServiceFacilityGroup</i>	:::	0:*	<i>siri:ServiceFacilityGroup</i>	Vehicle and service facilities (see 8.4.7.3).
	<i>AssistanceFacility</i>	0:*	<i>unknown / police / firstAid / sosPoint / specificAssistance / unaccompaniedMinorAssistance / boardingAssistance</i>	Classification of assistance facilities.
	<i>HireFacility</i>	0:*	<i>unknown / carHire / motorCycleHire / cycleHire / taxi / recreationDeviceHire</i>	Classification of hire facilities.

8.4.8 OJP_SituationSupport

The XML schema definition file *OJP_SituationSupport.xsd* provides definitions from the SIRI Situation Exchange service (SIRI SX). They can be used for transmitting reports on incidents and events. The definitions within this file aim at two objectives: a) to encapsulate the import of SIRI SX structures at one single place within OJP and b) to create a level of abstraction that allows for further enhancements without the need to change the SIRI definitions.

The subsequent sections describe the complex structures defined in OJP_SituationSupport.xsd.

8.4.8.1 SituationsStructure

Table 60 — Description of *SituationsStructure*.

<i>SituationsStructure</i>			<i>+Structure</i>	Container for the structured description of situations in public transport or on the road (individual transport).
	<i>PtSituation</i>	0:*	<i>siri:PtSituationElement</i>	Wrapper for SIRI SX situation elements in public transport, see CEN, EN 15531 Part 5.
	<i>RoadSituation</i>	0:*	<i>siri:RoadSituationElement</i>	Wrapper for SIRI SX situation elements in individual transport, see CEN, EN 15531 Part 5.

8.4.8.2 SituationFullRefStructure

Table 61 — Description of *SituationFullRefStructure*.

<i>SituationFullRefStructure</i>			<i>+Structure</i> (derived from <i>siri:SituationFullRefStructure</i>)	Reference to a situation object.
<i>SituationFullIdentity</i>	<i>VersionCountryRef</i>	0:1	<i>ifopt:CountryRef</i>	Reference to country, may be necessary to make <i>ParticipantRef</i> unique.
	<i>ParticipantRef</i>	1:1	<i>→ParticipantCode</i>	Unique identifier of the communication partner (see 8.2.11). Provides namespace for situation identifier.
	<i>SituationNumber</i>	1:1	<i>EntryQualifier</i>	Unique situation identifier.
<i>SituationUpdateIdentity</i>	<i>VersionCountryRef</i>	0:1	<i>ifopt:CountryRef</i>	Unique identifier of a country of a participant who created update situation element. Provides namespace for <i>VersionParticipant</i> . If absent same as <i>VersionCountryRef</i> .
	<i>UpdateParticipantRef</i>	0:1	<i>→ParticipantCode</i>	Unique identifier of the communication partner (see 8.2.11). Provides namespace for situation identifier.
	<i>Version</i>	0:1	<i>SituationVersion</i>	Version number of situation update. May be omitted if reference to base situation.

8.4.9 OJP_RequestSupport

The XML schema file OJP_RequestSupport.xsd defines some base types and structures that make the standard SIRI message exchange mechanisms usable within the OJP context.

8.4.9.1 Simple Types

This schema file defines the following simple types:

Table 62 — List of simple data type defined in OJP_RequestSupport.xsd.

<i>DataFrameRefType</i>	<i>xs:NMTOKEN</i>	Data version.
<i>CalcTimeType</i>	<i>xs:integer</i>	Calculation time in milliseconds.
<i>SignatureType</i>	<i>xs:string</i>	Data type for transmission of message signatures (public key cryptography).
<i>CertificateIdType</i>	<i>xs:NMTOKEN</i>	Data type for certificate identifiers (public key cryptography).

The subsequent sections describe the complex structures defined in OJP_RequestSupport.xsd.

8.4.9.2 AbstractOJPServiceRequestStructure

Table 63 — Description of *AbstractOJPServiceRequestStructure*.

<i>AbstractOJPServiceRequestStructure</i>			+Structure	Basic structure for all direct requests (without subscription)
<i>siri:AbstractServiceRequestStructure</i>	RequestTimestamp	1:1	<i>xs:dateTime</i>	Timestamp on request.
	MessageIdentifier	0:1	<i>siri:MessageQualifier</i>	Arbitrary unique identifier that can be used to reference this message.
<i>ServiceRequestContext</i>	DataFrameRef	0:1	<i>DataFrameRef</i>	Data version to be used by the server when processing the request.
	Extension	0:1	<i>xs:anyType</i>	Extensions.

8.4.9.3 OJP delivery structures, AbstractServiceDeliveryStructure (from SIRI)

Table 64 — *AbstractServiceDeliveryStructure (from SIRI)*.

<i>siri:AbstractServiceDeliveryStructure</i>			+Structure	Common data elements of service delivery structures.
<i>Log</i>	ResponseTimestamp	1:1	<i>xsd:dateTime</i>	Time individual response element was created.
<i>Endpoint properties</i>	RequestMessageRef	0:1	→ <i>MessageQualifier</i>	For direct requests, Identifier of request that this Delivery satisfies.
	SubscriberRef	0:1	→ <i>ParticipantCode</i>	Required if Delivery is for a Subscription, Participant Reference of Subscriber.
	SubscriptionFilterRef	0:1	→ <i>SubscriptionFilterCode</i>	Unique identifier of Subscription filter to which this subscription is assigned. If there is only a single filter, then can be omitted.
	SubscriptionRef	1:1	→ <i>SubscriptionQualifier</i>	Required if Delivery is for a Subscription, Identifier of Subscription issued by Requestor. Unique within Subscriber (i.e. within ParticipantRef of Subscriber), and SIRI Functional Service type.
<i>Delegation</i>	DelegatorAddress	0:1	<i>Xsd:anyURI</i>	Address of original Consumer, i.e. requesting system to which delegating response is to be returned. +SIRI 2.0
	DelegatorRef	0:1	→ <i>ParticipantCode</i>	Identifier of delegating system that originated message. +SIRI 2.0
<i>Status</i>	Status	0:1	<i>xsd:boolean</i>	Whether the complete request could be processed successfully or not. Default is true. If any of the individual requests within the delivery failed, should be set to <i>false</i> .
	ErrorCondition	0:1	+Structure	Description of any error or warning conditions that apply to the specific functional request or response.
			<i>choice</i>	One of the following Error codes.

	<i>A</i>	<i>ServiceNotAvailableError</i>			Error: Functional service is not available to use (but it is still capable of giving this response).
	<i>b</i>	<i>CapabilityNotSupportedError</i>		<i>+ Error</i>	Error: Capability not supported.
	<i>c</i>	<i>AccessNotAllowedError</i>		<i>+Error</i>	Error: Requestor is not authorised to use the service or data requested.
	<i>d</i>	<i>InvalidDataReferencesError</i>		<i>+Error</i>	Error: Request contains references to identifiers that are not known. +SIRI v2.0.
	<i>E</i>	<i>BeyondDataHorizon</i>		<i>+Error</i>	Error: Data period or subscription period is outside of period covered by service. +SIRI v2.0.
	<i>f</i>	<i>NoInfoForTopicError</i>		<i>+Error</i>	Error: Valid request was made but service does not hold any data for the requested topic expression.
	<i>G</i>	<i>ParametersIgnoredError</i>	- 1:1	<i>+Error</i>	Error: Request contained parameters that were not supported by the producer. A response has been provided but some parameters have been ignored. +SIRI v2.0.
	<i>H</i>	<i>UnknownExtensionsError</i>		<i>+Error</i>	Error: Request contained extensions that were not supported by the producer. A response has been provided but some or all extensions have been ignored. +SIRI v2.0.
	<i>i</i>	<i>AllowedResourceUsageExceededError</i>		<i>+Error</i>	Error: Valid request was made but request would exceed the permitted resource usage of the client.
	<i>j</i>	<i>OtherError</i>		<i>+Error</i>	Error other than a well-defined category.
		<i>Description</i>	0:1	<i>→ErrorDescription</i>	Description of Error.
		<i>ValidUntil</i>	0:1	<i>xsd:dateTime</i>	End of data horizon of the data producer.
		<i>ShortestPossibleCycle</i>	0:1	<i>PositiveDurationType</i>	Minimum interval at which updates can be sent.
		<i>DefaultLanguage</i>		<i>Xsd:language</i>	Default language for text elements.

The generic *siri:AbstractServiceDeliveryStructure* is used to define each specific OJP service delivery. It encapsulates the data elements that are common to all specific OJP delivery structures. The OJP delivery structures inherit from *siri:AbstractServiceDeliveryStructure*. Their structure is defined according to the following template:

Table 65 – Template for the definition of the OJP delivery structures.

<i>OJPxxxDeliveryStructure</i>			<i>+Structure (derived from siri:AbstractServiceDeliveryStructure)</i>	Delivery structure for OJP service xxx.
	<i>OJPxxxRequest</i>	0:1	<i>+OJPxxxRequest</i>	Repetition of the request contents.
<i>ServiceResponseContext</i>	<i>DataFrameRef</i>	0:1	<i>→siri:DataFrame</i>	Data frame (data version) that has been used by the server when processing the request.
	<i>CalcTime</i>	0:1	<i>CalcTime</i>	Calculation time in milliseconds.
	<i>xxxResponseContext</i>	0:1	<i>+xxxResponseContext</i>	Context for OJP response.
<i>xxxResponse</i>	<i>xxxResult</i>	0:*	<i>+xxxResult</i>	OJP result structure.
	Extensions	0:1	<i>xs:any</i>	Placeholder for user extensions.

In the above table the placeholder xxx for one of the OJP services could be replaced, for example, by *Trip* or *StopEvent*.

8.4.10 OJP_FareSupport

The XML schema file OJP_FareSupport.xsd defines a set of base types and structures to be used for giving fare information on passenger trips.

8.4.10.1 Simple Types

This schema file defines the following simple types:

Table 66 — List of simple type definitions in OJP_FareSupport.xsd.

<i>FareAuthorityCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a fare authority, eg: "VVS" or "DBAG".
<i>TariffZoneCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a tariff zone (issued by a fare authority or transport operator).
<i>FareProductCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a fare product. Unique within the realm of a fare authority or operator fare system.
<i>EntitlementProductCodeType</i>	<i>xs:NMTOKEN</i>	Identifier of a traveller card (eg: BahnCard50, BahnCard50First etc.).
<i>TypeOfFareClass Enumeration</i>	<i>all first second third business economy</i>	Enumeration of travel classes.
<i>VatRateEnumeration</i>	<i>no full half mixed unknown</i>	Enumeration of Value Added Tax rates.
<i>PassengerCategoryEnumeration</i>	<i>Adult Child Senior Youth Disabled</i>	A simplified and specialised view of User Profile (Classification of passengers for Fare Product pricing).

The identifiers of fare authorities, tariff zones etc. have to meet some requirements in order to be interpretable across system boundaries. More details on this topic can be found in section 8.2.

The subsequent sections describe the complex structures defined in OJP_FareSupport.xsd.

8.4.10.2 **TariffZoneStructure****Table 67 — Description of *TariffZoneStructure*.**

<i>TariffZoneStructure</i>			<i>+Structure</i>	Tariff zone model with label that is known to the public.
	<i>TariffZoneRef</i>	1:1	<i>→TariffZoneCode</i>	Code of a tariff zone (see 8.2.14).
	<i>TariffZoneText</i>	1:1	<i>xs:string</i>	Label of tariff zone that is known by passengers.

8.4.10.3 **TariffZoneListInAreaStructure****Table 68 — Description of *TariffZoneListInAreaStructure*.**

<i>TariffZoneListInAreaStructure</i>			<i>+Structure</i>	List of tariff zones within the area of a Fare Authority.
<i>FareAuthority</i>	<i>FareAuthorityRef</i>	1:1	<i>→FareAuthorityCode</i>	Reference to a Fare Authority (see 8.4.10.1).
	<i>FareAuthorityText</i>	1:1	<i>xs:string</i>	Description or name of fare authority
	<i>TariffZone</i>	1:*	<i>+TariffZone</i>	One or more tariff zones (see 8.4.10.2).

8.4.10.4 **BookingArrangementStructure****Table 69 — Description of *BookingArrangementStructure*.**

<i>BookingArrangementStructure</i>			<i>+Structure</i>	Description of a booking option.
	<i>BookingAgencyName</i>	0:1	<i>+InternationalText</i>	Name of the booking agency (contractual partner).
	<i>BookingUrl</i>	0:1	<i>xs:anyURI</i>	URL of online booking service.
	<i>InfoUrl</i>	0:1	<i>xs:anyURI</i>	URL of information page.
	<i>PhoneNumber</i>	0:1	<i>PhoneNumber</i>	Phone number for booking (see 8.4.2.1).
	<i>MinimumBookingPeriod</i>	0:1	<i>xs:duration</i>	Minimum duration bookings have to be completed before trip starts.
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.4.10.5 **BookingArrangementsContainerStructure****Table 70 — Description of *BookingArrangementsContainerStructure*.**

<i>BookingArrangementsContainerStructure</i>			<i>+Structure</i>	Container for multiple <i>BookingArrangement</i> structures.
	<i>BookingArrangement</i>	1:*	<i>+BookingArrangement</i>	One or more booking opportunities (see 8.4.10.4).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.4.10.6 FareProductStructure

Table 71 — Description of FareProductStructure.

<i>FareProductStructure</i>			<i>+Structure</i>	Model of a FareProduct and related information.
	FareProductId	1:1	→FareProductCode	Unique identifier of the fare product (see 8.4.10.1).
	FareProductName	1:1	<i>xs:string</i>	Printable fare product name/product name.
<i>FareAuthority</i>	FareAuthorityRef	1:1	→FareAuthorityCode	Reference to a Fare Authority (see 8.4.10.1).
	FareAuthorityText	1:1	<i>xs:string</i>	Description or name of fare authority.
<i>FareProductPrice</i>	<i>Price</i>	0:1	<i>xs:decimal</i>	fare product price as decimal number.
	<i>NetPrice</i>	0:1	<i>xs:decimal</i>	Net fare product price as decimal number for accounting purposes.
	<i>Currency</i>	0:1	<i>xs:NMTOKEN</i>	ISO 4217 currency code, e.g. EUR for Euro or GBP for Pound Sterling
	<i>VatRate</i>	0:1	<i>VatRateEnumeration</i>	Rate of value added tax (see 8.4.10.1). Default is <i>unknown</i> .
<i>FareProductValidity</i>	<i>TypeOfFareClass</i>	0:1	<i>TypeOfFareClass Enumeration</i>	Travel class for which the fare product is valid (see 8.4.10.1).
	<i>RequiredCard</i>	0:*	→EntitlementProductCode	One or more traveller cards that are needed for purchase of this fare product. In most cases traveller cards offer discounts, eg: BahnCard50 of Deutsche Bahn (see 8.4.10.1).
	<i>ValidFor</i>	0:*	<i>PassengerCategoryEnumeration</i>	Sequence of all passenger categories, for which this fare product is valid (see 8.4.10.1).
	<i>ValidityDuration</i>	0:1	<i>xs:duration</i>	Maximum temporal validity of the fare product after purchase or fare product validation.
	<i>ValidityDurationText</i>	0:1	<i>+InternationalText</i>	Description of the temporal validity.
	<i>ValidityTariffZones</i>	0:1	<i>+TariffZoneListInArea</i>	Spatial validity of the fare product expressed as list of tariff zones for which this fare product is valid (see 8.2.14).
	<i>ValidityAreaText</i>	0:1	<i>+InternationalText</i>	Description of the spatial validity area.
<i>FareProductBooking</i>	<i>InfoUrl</i>	0:*	<i>xs:anyURI</i>	URL of information for this fare product (see 8.4.2.4).
	<i>SaleUrl</i>	0:*	<i>xs:anyURI</i>	URL to buy the fare product online (see 8.4.2.4).
	<i>BookingArrangements</i>	0:*	<i>+BookingArrangementsContainerInfo</i>	Booking options (see 8.4.10.5).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.4.10.7 TripFareResultStructure

Table 72 — Description of TripFareResultStructure.

<i>TripFareResultStructure</i>			<i>+Structure</i>	Structure containing the fare result for a passenger trip (or parts of it).
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Result-specific error messages. For possible values refer to the subsequent table. (See also 8.4.4.2).
<i>TripLegRange</i>	<i>FromTripLegIdRef</i>	0:1	<i>xs:NMTOKEN</i>	Range of TripLegs (from - to) for which a fare result (ticket) is valid. Identifies the "valid from" TripLeg.
	<i>ToTripLegIdRef</i>	0:1	<i>xs:NMTOKEN</i>	Identifies the "valid to" TripLeg.

	<i>PassedZones</i>	0:1	<i>+TariffZoneListInArea</i>	The sequence of passed tariff zones (see 8.4.10.30).
	<i>FareProduct</i>	0:*	<i>+FareProduct</i>	One or more tickets that are valid for this part of the trip (see 8.4.10.6).
	<i>StaticInfoURL</i>	0:*	<i>+WebLink</i>	URL of static information page on the web (see 8.4.2.4).

The following error codes can appear in *ErrorMessage* but do not appear in the XSD:

Table 73 — List of possible error codes in *TripFareResult*.

<i>FARES_OUTOFAREA</i>	The trip planning has found a route that leaves the area of the fare authority.
<i>FARES_JOURNEYNOTPERMITTED</i>	The trip planning result suggest a service which is not permitted by the fare authority.
<i>FARES_ADDITIONALCHARGES</i>	Passengers may be charged additional fees (eg: for road tolls or seat reservation).
<i>FARES_ADDITIONALTICKETS</i>	Additional tickets may be necessary because only parts of the passenger trip could be covered..
<i>FARES_ROUTENOTFEASIBLE</i>	Fare calculation is not possible because the suggested trip is not in compliance with the fare regulations (eg: because of round trips, TripLegs that go forth and return or exceeding the maximal total trip duration).
<i>FARES_ALREADYCOVERED</i>	The ticket that has been specified in the request is valid for the suggested trip (or parts of it as defined by <i>TripLegRange</i>).

8.4.10.8 FarePassengerStructure

Table 74 — Description of *FarePassengerStructure*.

<i>FarePassengerStructure</i>				<i>+Structure</i>	Structure of a passenger profile for calculation.
	<i>a</i>	<i>Age</i>	-	<i>xs:nonNegativeInteger</i>	Age of the passenger on the day of travel.
	<i>b</i>	<i>PassengerCategory</i>		<i>PassengerCategoryEnumeration</i>	Passenger category to which this person is belonging (see 8.4.10.1).
	<i>EntitlementProduct</i>		0:*	<i>→EntitlementProductCode</i>	One or more Entitlement Products that the passenger can make use of (see 8.4.10.1).
	<i>a</i>	<i>ZonesAlreadyPaid</i>	- 0:1	<i>+TariffZoneListInArea</i>	List of Tariff Zones for which the passenger already has a valid Fare Product (see 8.2.14).
	<i>b</i>	<i>SalesPackageElementRef</i>	- 0:*	<i>→FareProductCode</i>	One or more fare products the passenger already owns and can make use of for this trip (or at least parts of it).

8.4.10.9 FareParamStructure

Table 75 — Description of *FareParamStructure*.

<i>FareParamStructure</i>				<i>+Structure</i>	Parameters for retrieval request.
<i>FareDataFilter</i>	<i>FareAuthorityFilter</i>	0:*		<i>→FareAuthorityCode</i>	Reference to Fare Authorities to be considered (see 8.4.10.1).
	<i>PassengerCategory</i>	0:*		<i>PassengerCategoryEnumeration</i>	Passenger categories (view of User Profile) to be considered (see 8.4.10.1).
	<i>TypeOfFareClass</i>	0:1		<i>TypeOfFareClass Enumeration</i>	Travel classes to be considered (see 8.4.10.1). Refers to Type Of Fare Class

	<i>Traveller</i>	0:*	<i>+FarePassenger</i>	Number of travellers that will make the trip and for which fare information needs to be gathered (see 8.4.10.8).
--	------------------	-----	-----------------------	--

8.5 Service Location Information

8.5.1 Description

The Location Information service comprises the following four base functions:

- matching text input against possible origin and destination locations,
- retrieval of all location objects (bunch delivery),
- geographical context service that provides location objects within a bounding box,
- reverse address resolution service that delivers the nearest address for a given coordinate.

By means of abstraction these functions are assembled within one single service – the Location Information service. This way, even more possible applications have arisen, such as:

- finding the nearest stops/stations for a given coordinate,
- matching text input against the names of locations near a given coordinate.

The XML schema file OJP_Locations.xsd defines data types and structures for use in the Location Information service.

8.5.2 Simple Types

This service defines the following simple types:

Table 76 — List of simple type definitions in OJP_Locations.xsd.

<i>PlaceTypeEnumeration</i>	<i>stop address poi coord topographicPlace</i>	Type of a location object.
<i>PlaceUsageEnumeration</i>	<i>origin destination via</i>	Usage of location object in context of trip planning.

8.5.3 Request Structures

Location information can be gathered by using a ***LocationInformationRequest*** element (type *LocationInformationRequestStructure*).

8.5.3.1 LocationInformationRequestStructure

Table 77 — Description of *LocationInformationRequestStructure*.

LocationInformationRequestStructure			+Structure	Container for location information request details.	
	a	InitialInput	-1:1	+InitialLocationInput	Input data for an initial location information request (see 8.5.3.2).
	b	PlaceRef		+PlaceRef	Reference of a Place for which more details are to be retrieved. If Places are organised hierarchically it may be reasonable to identify the Place in a top-down approach with several steps of refining a Place on each level of hierarchy. Following this approach an initial request retrieves a first list of top-level Places (eg: streets) which are to be refined in a subsequent request to the next level (eg: house number intervals). The objects of the current level are presented to the user for selection. The object reference of the selected object is then sent in the next request for further refinement. (See also 8.4.5.11).
	Restrictions		0:1	+PlaceParam	More parameters for restricting the request (see 8.5.3.7).
	Extension		0:1	xs:anyType	Extensions.

8.5.3.2 InitialLocationInputStructure

Table 78 — Description of *InitialLocationInputStructure*.

<i>InitialLocationInputStructure</i>			<i>+Structure</i>	Contains the parameters for the initial location request.
	<i>LocationName</i>	0:1	<i>xs:string</i>	Name of the location object which is looked for. This is usually the user's input. If not given, the name for the resulting location objects is not relevant.
	<i>GeoPosition</i>	0:1	<i>+GeoPosition</i>	Coordinate where to look for locations. If given, the result should prefer location objects near to this geoposition. (See also 8.4.2.3).
	<i>GeoRestriction</i>	0:1	<i>+GeoRestrictions</i>	Parameters for geographical filtering restricting the search (see 8.5.3.3).

8.5.3.3 GeoRestrictionsStructure

Table 79 — Description of *GeoRestrictionsStructure*.

<i>GeoRestrictionsStructure</i>				<i>+Structure</i>	Defines restrictions for geographical filtering.
	<i>a</i>	<i>Circle</i>	-1:1	<i>+GeoCircle</i>	The filter is defined by a circle (see 8.5.3.4).
	<i>b</i>	<i>Rectangle</i>	-1:1	<i>+GeoRectangle</i>	The filter is defined by a rectangle (see 8.5.3.5).
	<i>c</i>	<i>Zone</i>	-1:1	<i>+GeoArea</i>	The filter is defined by a polygon (see 8.5.3.6).

8.5.3.4 GeoCircleStructure

Table 80 — Description of *GeoCircleStructure*.

<i>GeoCircleStructure</i>			<i>+Structure</i>	Defines a circle (based on a geographical position).
	<i>Centre</i>	<i>1:1</i>	<i>+GeoPosition</i>	Centre position (see 8.4.2.3).
	<i>Radius</i>	<i>1:1</i>	<i>Distance</i>	Radius in metres.

8.5.3.5 GeoRectangleStructure

Table 81 — Description of *GeoRectangleStructure*.

<i>GeoRectangleStructure</i>			<i>+Structure</i>	Defines a rectangle (based on geographical positions).
	<i>UpperLeft</i>	1:1	<i>+GeoPosition</i>	Upper-left (north-west) corner of the rectangle (see 8.4.2.3).
	<i>LowerRight</i>	1:1	<i>+GeoPosition</i>	Lower-right (south-east) corner of the rectangle (see 8.4.2.3).

8.5.3.6 GeoAreaStructure

Table 82 — Description of *GeoAreaStructure*.

<i>GeoAreaStructure</i>			<i>+Structure</i>	Defines a polygon (based on geographical positions).
	<i>PolylinePoint</i>	3:*	<i>+GeoPosition</i>	Vertices of the polygonal area (see 8.4.2.3).

8.5.3.7 PlaceParamStructure

Table 83 — Description of *PlaceParamStructure*.

<i>PlaceParamStructure</i>			<i>+Structure</i>	Contains the parameters controlling the search for location objects.
<i>LocationDataFilter</i>	<i>Type</i>	0:*	<i>stop / address / poi / coord / TopographicPlace</i>	Allowed location object types. If none is given, all types are allowed.
	<i>Usage</i>	0:1	<i>origin / destination / via</i>	Defines, whether location objects for origin, via, or destination are searched.
	<i>PtModes</i>	0:1	<i>+PtModeFilter</i>	Allowed public transport modes (see 8.4.3.5). Defines, which public transport modes have to be available at the returned location objects. Applies only to stops.
	<i>OperatorFilter</i>	0:1	<i>+OperatorFilter</i>	Filter for locations that are operated by certain organisations (see 8.2.4).
	<i>TopographicPlaceRef</i>	0:*	<i>→TopographicPlaceCode</i>	If at least one is set, only location objects within the given localities are allowed (see 8.4.5.1).
	<i>PointOfInterestFilter</i>	0:1	<i>+PointOfInterestFilter</i>	Filter to narrow down POI searches (see 8.4.5.8).
<i>LocationPolicy</i>	<i>Language</i>	0:1	<i>xs:language</i>	Preferred language in which to return text values.
	<i>NumberOfResults</i>	0:1	<i>xs:positiveInteger</i>	Maximum number of results to be returned. The service is allowed to return fewer objects if reasonable or otherwise appropriate. If the number of matching objects is expected to be large (eg: in the case that all objects should be delivered) this parameter can be used to partition the response delivery into smaller chunks. The location information service is expected to support a response volume of at least 500 location objects within one single response.
	<i>ContinueAt</i>	0:1	<i>xs:nonNegativeInteger</i>	Tells the server to skip the mentioned number of results in its response. Can be used in a follow-up request to get further results. The value is usually taken from the previous response (element <i>ContinueAt</i> , see 8.5.4.1. To request the remaining objects the request is repeated with <i>ContinueAt</i> set to the value given by the last response.
	<i>IncludePtModes</i>	0:1	<i>xs:boolean</i>	Tells the service to include the available public transport modes at returned stops. Default is <i>false</i> .

8.5.4 Response Structures

An element ***PlaceInformationResponse*** of the type *PlaceInformationResponseStructure* is used to respond to a location information request.

8.5.4.1 PlaceInformationResponseStructure

Table 84 — Description of *PlaceInformationResponseStructure*.

<i>PlaceInformationResponseStructure</i>			<i>+Structure</i>	Container for the results of a location information query.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
	<i>ContinueAt</i>	0:1	<i>xs:nonNegativeInteger</i>	If this value is set the service indicates that there are more location objects to be returned than one response can transmit (due to size limits). The value of <i>ContinueAt</i> can be used in a follow-up request to get further results (see 8.5.3.7). It tells the server to skip the given number of results in its response.
	<i>Place</i>	0:*	<i>+PlaceResult</i>	The location object results found by the service. They have to be sorted by the ranking how well they match to the input data in descending order. The first result in the list matches best (see 8.5.4.2).

The following error codes can appear in *ErrorMessage* but do not appear in the XSD:

Table 85 — List of possible error codes in *LocationInformationResponse*.

<i>LOCATION_NORESULTS</i>	No location objects could be found that match the input data.
<i>LOCATION_UNSUPPORTEDTYPE</i>	The requested location types are not supported by the service.
<i>LOCATION_UNSUPPORTEDCOMBINATION</i>	The combination of input data (text string, coordinates, geographical restrictions) cannot be processed by the service.
<i>LOCATION_NOREFINEMENT</i>	The given location object could not be refined.
<i>LOCATION_USAGEIGNORED</i>	The usage type has been ignored.
<i>LOCATION_UNSUPPORTEDPTMODES</i>	The service does not support any restrictions by transport modes.
<i>LOCATION_UNSUPPORTEDLOCALITY</i>	The service does not support any restrictions by localities.

8.5.4.2 PlaceResultStructure

Table 86 — Description of *PlaceResultStructure*.

<i>PlaceResultStructure</i>			<i>+Structure</i>	Result structure for a location object.
	<i>Place</i>	1:1	<i>+Place</i>	The Place object (see 8.4.5).
	<i>Complete</i>	1:1	<i>xs:boolean</i>	States whether the included Place is complete or needs further refinement. Only complete Places are fully resolved and can be used in e.g. trip requests. Incomplete Places have to be refined entering them once again into a LocationInformationRequest.
	<i>Probability</i>	0:1	<i>xs:float</i>	Probability, that this result is the one meant by the user's input. Value should be between 0 and 1.
	<i>Mode</i>	0:*	<i>+Mode</i>	List of transport modes that call at this location object. This list should only be created in case of stop points or stop places – and only when explicitly requested. (See 8.4.3.1)

8.6 Service Exchange Points

8.6.1 Description

Distributed journey planning requires several journey planning systems planning parts of the whole trip which must be assembled. Each of the planners will therefore get a sub-query to plan: the first planner from the origin of the trip to its system boundaries, the next planner has to find trips from these boundaries to its boundaries with the next systems. This process will be continued until the final system where the destination of the user's trip is located.

The boundary points where the trip calculation is handed over to the next journey planning system are called exchange points. If they are not known in advance the exchange points can be looked up from a server by using the exchange points service.

8.6.2 Request Structures

Exchange points can be gathered by using an ***ExchangePointsRequest*** element (type *ExchangePointsRequestStructure*).

8.6.2.1 ExchangePointsRequestStructure

Table 87 — Description of *ExchangePointsRequestStructure*.

<i>ExchangePointsRequestStructure</i>			+Structure	Container for exchange points request details.
<i>ExchangePointsRequest</i>	<i>PlaceRef</i>	0:1	+ <i>PlaceRef</i>	Location for which exchange points to other "neighbour" systems are to be searched. This location is usually the origin/destination of a passenger trip. May be omitted if all exchange points shall be returned. See also 8.4.5.11.
	<i>Params</i>	0:1	+ <i>ExchangePointsParam</i>	More parameters for restricting the request (see 8.6.2.2).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.6.2.2 ExchangePointsParamStructure

Table 88 — Description of *ExchangePointsParamStructure*.

<i>ExchangePointsParamStructure</i>			+Structure	Contains the parameters controlling the search for exchange points.
<i>ExchangePointsDataFilter</i>	<i>Type</i>	0:*	<i>stop</i> / <i>address</i> / <i>poi</i> / <i>coord</i> / <i>TopographicPlace</i>	Allowed location object types. If none is given, all types are allowed.
	<i>Usage</i>	0:1	<i>origin</i> / <i>destination</i> / <i>via</i>	Defines, whether location objects for origin, via, or destination are searched.
	<i>PtModes</i>	0:1	+ <i>PtModeFilter</i>	Allowed public transport modes (see 8.4.3.5). Defines, which public transport modes have to be available at the returned location objects. Applies only to stops.
	<i>OperatorFilter</i>	0:1	+ <i>OperatorFilter</i>	Filter for locations that are operated by certain organisations (see 8.2.4).
	<i>TopographicPlaceRef</i>	0:*	→ <i>TopographicPlaceCode</i>	If at least one is set, only location objects within the given localities are allowed (see 8.4.5.1).

	<i>DestinationSystem</i>	0:1	<i>siri:ParticipantRef</i>	Reference to system in which the destination (or origin) of the passenger is located. See also 8.4.4.1.
	<i>AdjacentSystem</i>	0:*	<i>siri:ParticipantRef</i>	One or more adjacent systems for which the exchange points should be retrieved. See also 8.4.4.1.
<i>ExchangePointsPolicy</i>	<i>Language</i>	0:1	<i>xs:language</i>	Preferred language in which to return text values.
	<i>NumberOfResults</i>	0:1	<i>xs:positiveInteger</i>	Maximum number of results to be returned. The service is allowed to return fewer objects if reasonable or otherwise appropriate. If the number of matching objects is expected to be large (eg: in the case that all objects should be delivered) this parameter can be used to partition the response delivery into smaller chunks. The location information service is expected to support a response volume of at least 500 location objects within one single response.
	<i>ContinueAt</i>	0:1	<i>xs:nonNegativeInteger</i>	Tells the server to skip the mentioned number of results in its response. Can be used in a follow-up request to get further results. The value is usually taken from the previous response (element <i>ContinueAt</i> , see 8.6.3.1). To request the remaining objects the request is repeated with <i>ContinueAt</i> set to the value given by the last response.

8.6.3 Response Structures

An element ***ExchangePointsResponse*** of the type *ExchangePointsResponseStructure* is used to respond to an exchange points request.

8.6.3.1 ExchangePointsResponseStructure

Table 89 — Description of *ExchangePointsResponseStructure*.

<i>ExchangePointsResponseStructure</i>			<i>+Structure</i>	Container for the results of an exchange points query.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
<i>ExchangePointsResponse</i>	<i>ContinueAt</i>	0:1	<i>xs:nonNegativeInteger</i>	If this value is set the service indicates that there are more location objects to be returned than one response can transmit (due to size limits). The value of <i>ContinueAt</i> can be used in a follow-up request to get further results (see 8.5.3.7). It tells the server to skip the given number of results in its response.
	<i>Place</i>	0:*	<i>+ExchangePointsResult</i>	The exchange points found by the service (see 8.6.3.2).

The following error codes can appear in *ErrorMessage* but do not appear in the XSD:

Table 90 — List of possible error codes in *ExchangePointsResponse*.

<i>EXCHANGEPOINTS_NORESULTS</i>	No exchange points could be found that match the query criteria.
<i>EXCHANGEPOINTS_UNKNOWNDESTINATION</i>	The destination system given in the request parameters is unknown.
<i>EXCHANGEPOINTS_UNKNOWNADJACENTSYSTEM</i>	One or more of the adjacent systems given in the request parameters are unknown.

8.6.3.2 ExchangePointsResultStructure

Table 91 — Description of *ExchangePointsResultStructure*.

<i>ExchangePointsResultStructure</i>			<i>+Structure</i>	Result structure for an exchange point object.
	<i>Place</i>	1:1	<i>+Place</i>	Place object that describes this exchange point (see 8.4.5).
	<i>TravelDurationEstimate</i>	0:1	<i>xs:duration</i>	Rough estimate of the travel duration from the specified place to this exchange point.
	<i>BorderPoint</i>	0:1	<i>xs:Boolean</i>	Flag if this exchange point is an administrative border point where timetables are cut off while services still may run through and connect the regions. At this kind of point passengers may continue their trip on the same service. Default is FALSE.
	<i>Mode</i>	0:*	<i>+Mode</i>	List of transport modes that call at this location object. This list should only be created in case of stop points or stop places – and only when explicitly requested. (See 8.4.3.1)

8.7 Service Intermodal Trip Information

8.7.1 Description

This service provides intermodal trip information from an origin location to a destination taking various user preferences into account. Most installations will do the public transport journey planning within one system and one data set, but there are environments, of course, where the task of journey planning is split into several sub-tasks. This approach is called distributed journey planning. The OJP API offers support for distributed journey planning. More details of the messages involved in distributed journey planning can be found in section 8.7.2.

8.7.2 Distributed Planning of Intermodal Trips

In distributed environments the complete trip is not calculated within one single system, instead the planning task is split and distributed to several planning engines. Figure 10 shows the general flow of OJP messages within distributed environments. The client system sends a *TripRequest* message (see 8.7.3.1) to its enquirer's home system (EHS). The EHS forwards this request to the distributing system (DS). The DS has implemented all the business logic to split the journey planning request into pieces and to find journey planning engines (responding systems, RS) that are able to calculate partial trip solutions. For planning a trip one, two or more RS may be involved and each of them may get more than one request before the total trip is completely calculated.

The optimisation strategies for the distributed calculation of public transport trips often require the finding of trip solutions from M origin to N destination locations where for each of the N destination locations (or M origin locations respectively) an optimal (partial) trip has to be found. The OJP API provides the *MultiPointTripRequest* (see 8.7.3.2) and *MultiPointTripResponse* (see 8.7.4.2) messages to support this.

Once all partial trip requests have been responded to the DS it combines them into an optimal trip result for the overall origin-destination request. This solution is passed to the client system via the EHS as a *TripResponseStructure* (see 8.7.4.1).

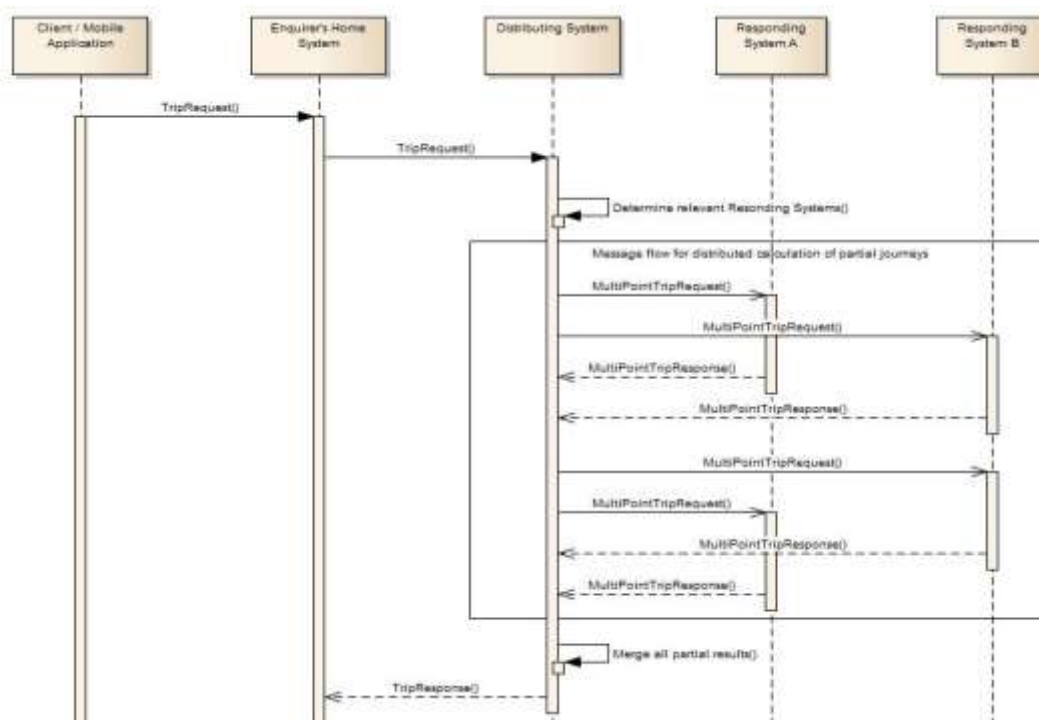


Figure 10 — General flow of OJP messages in distributed journey planning environments

To improve user experience there is the possibility of displaying trip summary information whilst the distributed calculation is still on-going. Depending on the distributed algorithm the existence of an optimal solution and the earliest arrival time at the user's destination can be determined before all trip details are delivered by all RS. In this scenario the *TripRequest* is responded to by the DS with a *TripSummary* (see 8.7.4.7) message (see Figure 11). This allows the client system to display the summary information before the whole distributed calculation process has come to an end. When all trip details finally have been found the DS actively sends a *TripResponse* message (within a *ServiceDelivery*) to the EHS which forwards it to the endpoint's address that was given in the *RequestorEndpointGroup* of the original client request. EHS and client system respond to this delivery by sending a *DataReceivedAcknowledgement* message.

If the Client's *TripRequest* asked for more than one trip solutions the EHS needs to know what the last message (containing a *TripResponse*) of the DS will be. Therefore the DS shall mark all intermediate messages with the *OJP:ServiceDelivery:MoreData* element set to *TRUE*. Only the last *ServiceDelivery* message should have set the *MoreData* element to *FALSE*.

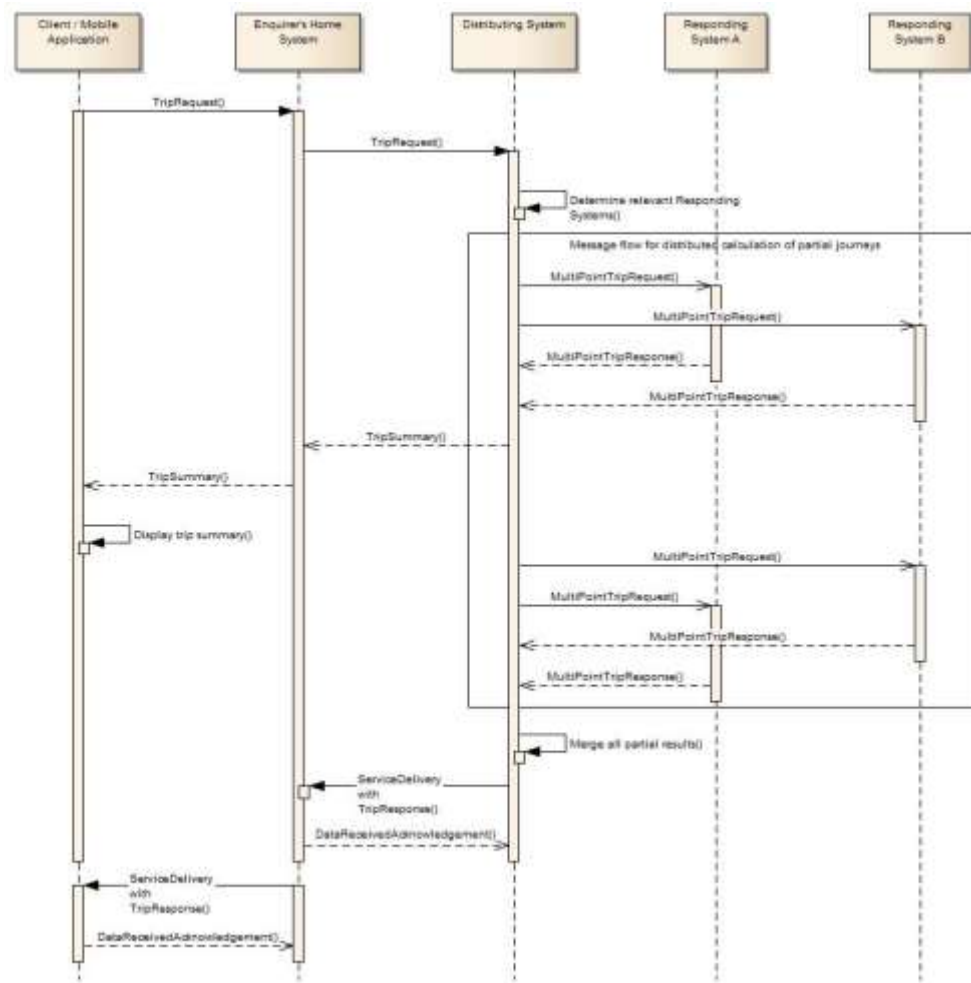


Figure 11 — General flow of OJP messages in distributed journey planning environments with deferred delivery of trip details.

8.7.3 Request Structures

Intermodal trip information can be gathered by using a *TripRequest* element (type *TripRequestStructure*).

8.7.3.1 TripRequestStructure

Table 92 — Description of *TripRequestStructure*.

<i>TripRequestStructure</i>			<i>+Structure</i>	Comprehends the request details for an intermodal trip calculation.
	Origin	1:*	<i>+PlaceContext</i>	Specifies the origin Place from where the user wants to start (see 8.4.5).
	Destination	1:*	<i>+PlaceContext</i>	Specifies the destination Place where the user is heading to (see 8.4.5).
	<i>Via</i>	<i>0:*</i>	<i>+Via</i>	Ordered series of points which the trip has to pass through. If more than one via point is given all of them have to be obeyed - in the correct order. The server is allowed to replace a via stop by equivalent stops (see 8.4.6.2).
	<i>NotVia</i>	<i>0:*</i>	<i>+NotVia</i>	Set of public transport stops which the trip may not pass through. If more than one not via point is given all of them have to be obeyed (see 8.7.3.6).

	<i>NoChangeAt</i>	0:*	<i>+NoChangeAt</i>	Set of public transport stops where the trip is not allowed to interchange (see 8.7.3.7).
	<i>Params</i>	0:1	<i>+TripParam</i>	Options to control the search behaviour and response contents (see 8.7.3.3).

Usually, the elements *Origin* and *Destination* will appear as single child elements. Only in the case of multiple origin (or destination) points where each one has its own departure time (or arrival time) more than one origin (or destination) elements should be used in the *TripRequest*. In this case the responding system will choose the optimal origin-destination point pair that allows for an optimal entire trip. This choice may depend on time of day and could therefore be different with each delivered trip solution.

8.7.3.2 MultiPointTripRequestStructure

Table 93 — Description of *MultiPointTripRequestStructure*.

<i>MultiPointTripRequestStructure</i>			<i>+Structure</i>	The request details for an intermodal multi-point trip calculation.
	<i>Origin</i>	1:*	<i>+PlaceContext</i>	Specifies the origin Place from where the user wants to start (see 8.4.5).
	<i>Destination</i>	1:*	<i>+PlaceContext</i>	Specifies the destination Place to which the user is heading (see 8.4.5).
	<i>Via</i>	0:*	<i>+Via</i>	Ordered series of points which the trip have to pass through. If more than one via point is given all of them have to be obeyed - in the correct order. The server is allowed to replace a via stop by equivalent stops (see 8.4.6.2).
	<i>NotVia</i>	0:*	<i>+NotVia</i>	Set of public transport stops which the trip may not pass through. If more than one not via point is given all of them have to be obeyed (see 8.7.3.6).
	<i>NoChangeAt</i>	0:*	<i>+NoChangeAt</i>	Set of public transport stops where an interchange with the trip is not allowed (see 8.7.3.7).
	<i>Params</i>	0:1	<i>+MultiPointTripParam</i>	Options to control the search behaviour and response contents (see 8.7.3.4).

8.7.3.3 TripParamStructure

Table 94 — Description of *TripParamStructure*.

<i>TripParamStructure</i>			<i>+Structure</i>	Parameters that affect the intermodal trip information.
<i>TripDataFilter</i>	<i>PtModeFilter</i>	0:1	<i>+PtModeFilter</i>	Modes to be considered in trip calculation (see 8.4.3.5).
	<i>LineFilter</i>	0:1	<i>+LineDirectionFilter</i>	Lines/Directions to include/exclude. (See 8.4.4.6).
	<i>OperatorFilter</i>	0:1	<i>+OperatorFilter</i>	Transport operators to include/exclude. (See 8.4.4.4).
	<i>PrivateModeFilter</i>	0:1	<i>+PrivateModeFilter</i>	Private mobility options to include/exclude (see 8.4.3.6).
<i>BaseTripMobilityFilter</i>	<i>NoSingleStep</i>	0:1	<i>xs.boolean</i>	If set to <i>true</i> the user is not able to climb one step. Default is <i>false</i> .
	<i>NoStairs</i>	0:1	<i>xs.boolean</i>	If set to <i>true</i> the user is not able to walk up/down stairs. Default is <i>false</i> .
	<i>NoEscalator</i>	0:1	<i>xs.boolean</i>	If set to <i>true</i> the user is not able to use an escalator. Default is <i>false</i> .

	<i>NoElevator</i>		0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to use an elevator. Default is <i>false</i> .
	<i>NoRamp</i>		0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to use a ramp. Default is <i>false</i> .
<i>TripMobilityFilter</i>	<i>LevelEntrance</i>		0:1	<i>xs:boolean</i>	If set to <i>true</i> the user needs vehicles with level entrance between platform and vehicle, eg: for wheelchair access. Lift-equipped vehicles or stationary lifts at the platform may be sufficient. Default is <i>false</i> .
	<i>BikeTransport</i>		0:1	<i>xs:boolean</i>	If set to <i>true</i> the user wants to carry a bike on public transport. Default is <i>false</i> .
	<i>WalkSpeed</i>		0:1	<i>OpenPercent</i>	Deviation from average walk speed in percent. 100% percent means average speed. Values less than 100 % mean slower, greater than 100% faster. Default is <i>100</i> .
<i>BaseTripPolicy</i>	<i>a</i>	<i>NumberOfResults</i>	- 0:1	<i>xs:positiveInteger</i>	The minimum number of trip results that the user wants to see.
	<i>b</i>	<i>:::</i>	- 0:1	<i>NumberOfResultsGroup</i>	To control the number of trip results before/after a point in time. May not be used when departure time at origin and arrival time at destination are set. (See 8.7.3.5).
	<i>IgnoreRealtimeData</i>		0:1	<i>xs:boolean</i>	If set to <i>true</i> then the trip calculation should not use any realtime or incident data. Default is <i>false</i> .
	<i>ImmediateTripStart</i>		0:1	<i>xs:boolean</i>	Whether the trip calculation should find a solution that starts immediately (eg: because the user is already on the way) instead of finding the latest possible start opportunity. Default is <i>false</i> .
<i>TripPolicy</i>	<i>TransferLimit</i>		0:1	<i>xs:positiveInteger</i>	The maximum number of interchanges the user will accept per trip.
	<i>OptimisationMethod</i>		0:1	<i>fastest / minChanges / leastWalking / leastCost / earliestArrival / latestDeparture / earliestArrivalAndLatestDeparture</i>	The type of the target function that should be applied when searching for optimal trip results.
	<i>ItModesToCover</i>		0:*	<i>IndividualModesEnumeration</i>	For each mode in this list (see 8.4.3.1) a separate monomodal trip shall be found - in addition to inter-modal solutions.
	<i>AcceptDeferredDelivery</i>		0:1	<i>xs:boolean</i>	If yes, then with the first response a summary of the trip solution(s) is enough. Full trip information is sent actively by the server to the Address stated in RequestorEndpointGroup. Default is <i>false</i> .
<i>BaseTripContentFilter</i>	<i>IncludeTrackSections</i>		0:1	<i>xs:boolean</i>	Whether the result should include TrackSection elements (see 8.4.6.16) to describe the geographic route of each TripLeg. Default is <i>false</i> .
	<i>IncludeLegProjection</i>		0:1	<i>xs:boolean</i>	Whether the result should include the geographic projection (coordinates) of each TripLeg. Default is <i>false</i> .
	<i>IncludeTurnDescription</i>		0:1	<i>xs:boolean</i>	Whether the result should include turn-by-turn instructions for each TripLeg. Default is <i>false</i> .
	<i>IncludeAccessibility</i>		0:1	<i>xs:boolean</i>	Whether the result should include accessibility information. Default is <i>false</i> .

<i>TripContentFilter</i>	<i>IncludeIntermediateStops</i>	0:1	<i>xs:boolean</i>	Whether the result should include intermediate stops (between the passenger's board and alight stops). Default is <i>false</i> .
	<i>IncludeFare</i>	0:1	<i>xs:boolean</i>	Whether the result should include fare information. Default is <i>false</i> .
	<i>IncludeOperatingDays</i>	0:1	<i>xs:boolean</i>	Whether the result should include operating day information - as encoded bit string and in natural language. Default is <i>false</i> .
	<i>FareParam</i>	0:1	<i>+FareParam</i>	Parameter for the fare calculation (see 8.4.10.9).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.7.3.4 MultiPointTripParamStructure

Table 95 — Description of *MultiPointTripParamStructure*.

<i>MultiPointTripParamStructure</i>			<i>+Structure</i>	Parameters that affect the intermodal multi-point trip calculation.
<i>TripDataFilter</i>	<i>PtModeFilter</i>	0:1	<i>+PtModeFilter</i>	Modes to be considered in trip calculation (see 8.4.3.5).
	<i>LineFilter</i>	0:1	<i>+LineDirectionFilter</i>	Lines/Directions to include/exclude. (See 8.4.4.6).
	<i>OperatorFilter</i>	0:1	<i>+OperatorFilter</i>	Transport operators to include/exclude. (See 8.4.4.4).
	<i>PrivateModeFilter</i>	0:1	<i>+PrivateModeFilter</i>	Private mobility options to include/exclude (see 8.4.3.6).
<i>BaseTripMobilityFilter</i>	<i>NoSingleStep</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to climb one step. Default is <i>false</i> .
	<i>NoStairs</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to walk up/down stairs. Default is <i>false</i> .
	<i>NoEscalator</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to use an escalator. Default is <i>false</i> .
	<i>NoElevator</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to use an elevator. Default is <i>false</i> .
	<i>NoRamp</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user is not able to use a ramp. Default is <i>false</i> .
<i>TripMobilityFilter</i>	<i>LevelEntrance</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user needs vehicles with level entrance between platform and vehicle, eg: for wheelchair access. Lift-equipped vehicles or stationary lifts at the platform may be sufficient. Default is <i>false</i> .
	<i>BikeTransport</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> the user wants to carry a bike on public transport. Default is <i>false</i> .
	<i>WalkSpeed</i>	0:1	<i>OpenPercent</i>	Deviation from average walk speed in percent. 100% percent means average speed. Values less than 100 % mean slower, greater than 100% faster. Default is <i>100</i> .
<i>MultiPointTripPolicy</i>	<i>IgnoreRealtimeData</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> then the trip calculation should not use any realtime or incident data. Default is <i>false</i> .

	<i>ImmediateTripStart</i>	0:1	<i>xs:boolean</i>	Whether the trip calculation should find a solution that starts immediately (eg: because the user is already on the way) instead of finding the latest possible start opportunity. Default is <i>false</i> .
	<i>TransferLimit</i>	0:1	<i>xs:positiveInteger</i>	The maximum number of interchanges the user will accept per trip.
	<i>OptimisationMethod</i>	0:1	<i>fastest / minChanges / leastWalking / leastCost / earliestArrival / latestDeparture / earliestArrivalAndLatestDeparture</i>	The type of the target function that should be applied when searching for optimal trip results.
	<i>MultiPointType</i>	0:1	<i>anyPoint / eachOrigin / eachDestination</i>	If 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.
<i>BaseTripContentFilter</i>	<i>IncludeTrackSections</i>	0:1	<i>xs:boolean</i>	Whether the result should include TrackSection elements (see 8.4.6.16 to describe the geographic route of each TripLeg. Default is <i>false</i> .
	<i>IncludeLegProjection</i>	0:1	<i>xs:boolean</i>	Whether the result should include the geographic projection (coordinates) of each TripLeg. Default is <i>false</i> .
	<i>IncludeTurnDescription</i>	0:1	<i>xs:boolean</i>	Whether the result should include turn-by-turn instructions for each TripLeg. Default is <i>false</i> .
	<i>IncludeAccessibility</i>	0:1	<i>xs:boolean</i>	Whether the result should include accessibility information. Default is <i>false</i> .
<i>TripContentFilter</i>	<i>IncludeIntermediateStops</i>	0:1	<i>xs:boolean</i>	Whether the result should include intermediate stops (between the passenger's board and alight stops). Default is <i>false</i> .
	<i>IncludeFare</i>	0:1	<i>xs:boolean</i>	Whether the result should include fare information. Default is <i>false</i> .
	<i>IncludeOperatingDays</i>	0:1	<i>xs:boolean</i>	Whether the result should include operating day information - as encoded bit string and in natural language. Default is <i>false</i> .
<i>MultiPointTripContentFilter</i>	<i>IncludeLegs</i>	0:1	<i>xs:boolean</i>	Whether the result should include leg information. Default is <i>false</i> .
	<i>FareParam</i>	0:1	<i>+FareParam</i>	Parameter for the fare calculation (see 8.4.10.9).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.7.3.5 NumberOfResultsGroup

Table 96 — Description of *NumberOfResultsGroup*.

<i>NumberOfResultsGroup</i>			<i>+Group</i>	To control the number of trip results before/after a point in time. May NOT be used when departure time at origin AND arrival time at destination are set.
	<i>NumberOfResultsBefore</i>	1:1	<i>xs:positiveInteger</i>	The desired number of trip results before the given time (at origin or destination).
	<i>NumberOfResultsAfter</i>	1:1	<i>xs:positiveInteger</i>	The desired number of trip results after the given time (at origin or destination).

8.7.3.6 NotViaStructure

Table 97 — Description of *NotViaStructure*.

<i>NotViaStructure</i>				<i>+Structure</i>	Not-via restrictions for a trip, i.e. stops that the trip is not allowed to pass through.
	<i>a</i>	<i>StopPointRef</i>	-1:1	→ <i>StopPoint</i>	Reference to a not-via stop point. (See 8.4.5.1).
	<i>b</i>	<i>StopPlaceRef</i>		→ <i>StopPlace</i>	Reference to a not-via stop place. (See 8.4.5.1).

8.7.3.7 NoChangeAtStructure

Table 98 — Description of *NoChangeAtStructure*.

<i>NoChangeAtStructure</i>				<i>+Structure</i>	No-change-at restrictions for a trip, i.e. stops at which the trip is not allowed to change.
	<i>a</i>	<i>StopPointRef</i>	-1:1	→ <i>StopPoint</i>	Reference to a not-via stop point. (See 8.4.5.1).
	<i>b</i>	<i>StopPlaceRef</i>		→ <i>StopPlace</i>	Reference to a not-via stop place. (See 8.4.5.1).

8.7.4 Response Structures

An element *TripResponse* of the type *TripResponseStructure* is used to respond to an intermodal trip request.

8.7.4.1 TripResponseStructure

Table 99 — Description of *TripResponseStructure*.

<i>TripResponseStructure</i>			<i>+Structure</i>	Contrains the response message data for an intermodal trip request.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
	<i>TripResponseContext</i>	0:1	<i>+TripResponseContext</i>	Context to hold trip response objects that occur frequently. (See 8.7.4.3).
	<i>TripResult</i>	0:*	<i>+TripResult</i>	The trip results found by the server. (See 8.7.4.4).

The following error codes can appear in *ErrorMessage*:

Table 100 — List of possible error codes in *TripResponse*.

<i>TRIP_NOTRIPFOUND</i>	No trip plan could be found that meets all the parameters as they have been set by the user (start and end locations, departure/arrival time and further options possibly set by the user).
<i>TRIP_ORIGINUNKNOWN</i>	The start location (address, stop place, ...) for the requested trip is unknown.
<i>TRIP_DESTINATIONUNKNOWN</i>	The end location (address, stop place, ...) for the requested trip is unknown.
<i>TRIP_VIAUNKNOWN</i>	One of the via points is unknown.
<i>TRIP_NOTVIAUNKNOWN</i>	One of the not-via points is unknown.

TRIP_NOCHANGEATUNKNOWN	One of the no-change-at stations is unknown.
TRIP_NOORIGIN	No start location has been defined for the trip.
TRIP_NODESTINATION	No end location has been defined for the trip.
TRIP_ORIGINDESTINATIONIDENTICAL	Start and end of the trip are identical.
TRIP_DATETIMEERROR	The requested date and/or time do not make sense.
TRIP_DEPARTUREAFTERARRIVAL	The requested departure time at each origin locations is after the requested arrival time at any destination location.
TRIP_DATEOUTOFRANGE	There is no timetable data available for the requested date.

8.7.4.2 MultiPointTripResponseStructure

Table 101 — Description of *MultiPointTripResponseStructure*.

<i>MultiPointTripResponseStructure</i>			+Structure	Contents of the response message data for an intermodal multi-point trip request.
	<i>ErrorMessage</i>	0:*	+ErrorMessage	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
	<i>TripResponseContext</i>	0:1	+TripResponseContext	Context to hold trip response objects that occur frequently. (See 8.7.4.3).
	<i>MultiPointTripResult</i>	0:*	+MultiPointTripResult	The multi-point trip results found by the server. (See 8.7.4.5).

In addition to the error codes defined in table 93 the following codes can appear in *ErrorMessage*:

Table 102 — List of possible error codes in *MultiPointTripResponse*.

MULTIPOINTTRIP_NOTALLPOINTSCOVERED	In case a multi-point request with <i>MultiPointType</i> set to <i>eachDestination</i> could not be responded with a trip solution to each of the destination points. And respectively in case a multi-point request with <i>MultiPointType</i> set to <i>eachOrigin</i> could not be responded with a trip solution for each of the origin points.
MULTIPOINTTRIP_TOOMANYPOINTS	In case a multi-point request use too many points as departure or arrival

8.7.4.3 TripResponseContextStructure

Table 103 — Description of *TripResponseContextStructure*.

<i>TripResponseContextStructure</i>	+Structure (derived from <i>AbstractResponseContextStructure</i>)	Container for data that frequently occurs within the response and is therefore referenced. (See 8.4.6.13).
--	---	--

8.7.4.4 TripResultStructure

Table 104 — Description of *TripResultStructure*.

<i>TripResultStructure</i>			<i>+Structure</i>	Structure for a single trip result and its accompanying error messages.
	ResultId	1:1	<i>xs:NMTOKEN</i>	Id of this trip result for referencing purposes. Unique within trip response.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Result-specific error messages. For possible values refer to the subsequent table. (See also 8.4.4.2).
	a Trip	-1:1	<i>+Trip</i>	Intermodal trip data. (See 8.7.4.6).
	b TripSummary	-1:1	<i>+TripSummary</i>	Summary of an intermodal or partial trip. (See 8.7.4.7).
	<i>TripFare</i>	0:*	<i>+TripFareResult</i>	Fare and fare product information for this trip as a whole or parts of it (see 8.4.10.7).

The following error codes can appear in *ErrorMessage*:

Table 105 — List of possible error codes in *TripResult*.

TRIP_ORIGINEQUIVALENT	The requested origin stop place has been replaced by an equivalent stop place.
TRIP_DESTINATIONEQUIVALENT	The requested destination stop place has been replaced by an equivalent stop place.
TRIP_VIAEQUIVALENT	One of the requested via stop places has been replaced by an equivalent stop place.
TRIP_REALTIMEINCOMPLETE	There is no realtime information available for at least one of the services within this trip result.
TRIP_ITTIMEEXTENDED	The maximum time allowed for using modes of individual transport (mostly walking or cycling) has been extended by the system because otherwise no trip could be found.
TRIP_ITMODECHANGED	The mode of individual transport specified by the user has been replaced by the system because otherwise no trip could be found. Usually this means taking a taxi instead of walking.
TRIP_INCONVENIENTWAITING	The trip plan in this trip result contains a long waiting time.

8.7.4.5 MultiPointTripResultStructure

Table 106 — Description of *MultiPointTripResultStructure*.

<i>MultiPointTripResultStructure</i>			<i>+Structure</i>	Structure for a single multi-point trip result and its accompanying error messages.
	ResultId	1:1	<i>xs:NMTOKEN</i>	Id of this trip result for referencing purposes. Unique within trip response.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Result-specific error messages. For possible values refer to the subsequent table. (See also 8.4.4.2).
	a Trip	-1:1	<i>+Trip</i>	Intermodal trip data. (See 8.7.4.6).
	b TripSummary	-1:1	<i>+TripSummary</i>	Summary of an intermodal or partial trip. (See 8.7.4.7).
	<i>TripFare</i>	0:*	<i>+TripFareResult</i>	Fare and ticket information for this trip as a whole or parts of it (see 8.4.10.7).

In addition to the error codes defined in 8.6.4.4 the following codes can appear in *ErrorMessage*:

Table 107 — List of possible error codes in *MultiPointTripResult*

<i>None defined at present</i>	<i>Placeholder for future use</i>
--------------------------------	-----------------------------------

8.7.4.6 TripStructure

Table 108 — Description of TripStructure

<i>TripStructure</i>			<i>+Structure</i>	Structure for an inter-modal passenger trip.
	<i>TripId</i>	1:1	<i>xs:NMTOKEN</i>	Id of this trip for referencing purposes. Unique within trip response.
	<i>Duration</i>	1:1	<i>xs:duration</i>	Overall duration of the trip.
	<i>StartTime</i>	1:1	<i>xs:dateTime</i>	Departure time from origin.
	<i>EndTime</i>	1:1	<i>xs:dateTime</i>	Arrival time at destination.
	<i>Transfers</i>	1:1	<i>xs:nonNegativeInteger</i>	Number of interchanges.
	<i>Distance</i>	0:1	<i>Distance</i>	Total trip distance.
	<i>TripLeg</i>	1:*	<i>+TripLeg</i>	Legs of this trip. (See 8.7.4.8).
<i>OperatingDays</i>	<i>OperatingDays</i>	0:1	<i>+OperatingDays</i>	Operating days when this trip can be made. (See 8.4.4.8).
	<i>OperatingDaysDescription</i>	0:1	<i>+InternationalText</i>	Human readable description of the operating days, eg: "Monday to Friday" or "Sunday and holidays".
	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>	Reference to situation message. Message details might be found in <i>TripResponseContext</i> (see 8.7.4.1) or through other communication channels (see 8.4.8.2).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.7.4.7 TripSummaryStructure

Table 109 — Description of TripSummaryStructure

<i>TripSummaryStructure</i>			<i>+Structure</i>	Structure for an inter-modal passenger trip.
	<i>TripId</i>	1:1	<i>xs:NMTOKEN</i>	Id of this trip for referencing purposes. Unique within trip response.
	<i>Origin</i>	0:1	<i>+PlaceRef</i>	Reference to origin location (see 8.4.5).
	<i>Destination</i>	0:1	<i>+PlaceRef</i>	Reference to destination location (see 8.4.5).
	<i>Duration</i>	0:1	<i>xs:duration</i>	Overall duration of the trip.
	<i>StartTime</i>	0:1	<i>xs:dateTime</i>	Departure time at origin.
	<i>EndTime</i>	0:1	<i>xs:dateTime</i>	Arrival time at destination.
	<i>PTTripLegs</i>	0:1	<i>xs:nonNegativeInteger</i>	Number of public transport TripLegs
	<i>Distance</i>	0:1	<i>Distance</i>	Total trip distance.
<i>OperatingDays</i>	<i>OperatingDays</i>	0:1	<i>+OperatingDays</i>	Operating days when this trip can be made. (See 8.4.4.8).
	<i>OperatingDaysDescription</i>	0:1	<i>+InternationalText</i>	Human readable description of the operating days, eg: "Monday to Friday" or "Sunday and holidays".

	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>	Reference to situation message. Message details might be found in <i>TripResponseContext</i> (see 8.7.4.1) or through other communication channels (see 8.4.8.2).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.7.4.8 TripLegStructure

Table 110 — Description of *TripLegStructure*.

<i>TripLegStructure</i>			<i>+Structure</i>	Structure for a single leg of a passenger trip.
	LegId	1:1	<i>xs:NMTOKEN</i>	Id of this TripLeg. Unique within <i>TripResult</i> .
	<i>ParticipantRef</i>	0:1	<i>→ParticipantCode</i>	Reference to the participating system from which this piece of information originates. (See 8.4.4.1)
	a TimedLeg	-1:1	<i>+TimedLeg</i>	Timetabled TripLeg. (See 8.7.4.9).
	b TransferLeg		<i>+TransferLeg</i>	TripLeg to connect between different services or modes of transport (interchanges). (See 8.7.4.10).
	c ContinuousLeg		<i>+ContinuousLeg</i>	TripLeg by a continuously available service or mode of transport. (See 8.7.4.110).

8.7.4.9 TimedLegStructure

Table 111 — Description of *TimedLegStructure*.

<i>TimedLegStructure</i>			<i>+Structure</i>	Passenger TripLeg with timetabled schedule.
	LegBoard	1:1	<i>+LegBoard</i>	Stop/Station where boarding is done. (See 8.7.4.12).
	<i>LegIntermediates</i>	0:*	<i>+LegIntermediate</i>	Information on the intermediate stop points passed between <i>LegBoard</i> and <i>LegAlight</i> . (See 8.7.4.14).
	LegAlight	1:1	<i>+LegAlight</i>	Stop/Station to alight. (See 8.7.4.13).
	Service	1:1	<i>+DatedJourney</i>	DATED VEHICLE JOURNEY that is used for this TripLeg. (See 8.4.6.3).
	<i>LegAttribute</i>	0:*	<i>+LegAttribute</i>	Attributes that are not valid on the whole service, but only on parts of the vehicle journey. (See 8.4.6.14).
<i>OperatingDays</i>	<i>OperatingDays</i>	0:1	<i>+OperatingDays</i>	Operating days when this trip can be made. (See 8.4.4.8).
	<i>OperatingDaysDescription</i>	0:1	<i>+InternationalText</i>	Human readable description of the operating days, eg: "Monday to Friday" or "Sunday and holidays".
	<i>LegTrack</i>	0:1	<i>+LegTrack</i>	Geographic embedding of this leg. (See 8.4.6.15).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.7.4.10 TransferLegStructure

Table 112 — Description of *TransferLegStructure*.

<i>TransferLegStructure</i>			<i>+Structure</i>	Leg of a trip that describes the interchange between different services.
	<i>a</i>	TransferMode	-1:1 <i>walk / parkAndRide / bikeAndRide / carHire / bikeHire / protectedConnection / guaranteedConnection / remainInVehicle / changeWithinVehicle / checkIn / checkOut</i>	Classification of interchange types.
	<i>b</i>	ContinuousMode		Continuous transport mode (see 8.4.3.1).
	LegStart		1:1 <i>+PlaceRef</i>	Start location of this leg. (See 8.4.5.11).
	LegEnd		1:1 <i>+PlaceRef</i>	End location of this leg. (See 8.4.5.11.)
<i>TimeWindow</i>	<i>TimeWindowStart</i>		0:1 <i>xs:dateTime</i>	The window of opportunity that the traveller has to perform this leg of the trip. Earliest time when this leg can be started.
	<i>TimeWindowEnd</i>		0:1 <i>xs:dateTime</i>	The window of opportunity that the traveller has to perform this leg of the trip. Latest time when this leg has to be completed.
<i>TransferDuration</i>	Duration		1:1 <i>xs:duration</i>	Overall duration of this connection.
	<i>WalkDuration</i>		0:1 <i>xs:duration</i>	Walk time as part of the overall connection duration.
	<i>BufferTime</i>		0:1 <i>xs:duration</i>	Buffer time as part of the overall connection duration. Buffer times, eg: check in/out times, sometimes are mandatory for using certain services as eg: airplanes, ferries or highspeed trains.
	<i>LegDescription</i>		0:1 <i>+InternationalText</i>	Text that describes this interchange.
	<i>Length</i>		0:1 <i>LengthType</i>	Length of this connection path.
	<i>Attribute</i>		0:* <i>+GeneralAttribute</i>	Notes or service attributes. (See 8.4.4.10).
	<i>PathGuidance</i>		0:1 <i>+PathGuidance</i>	Structured model further describing this interchange, its geographic embedding and accessibility. (See 8.7.4.15).
	<i>SituationFullRef</i>		0:* <i>+SituationFullRef</i>	Reference to situation message. Message details might be found in <i>TripResponseContext</i> (see 8.7.4.3) or through other communication channels (see 8.4.8.2).
	<i>Extension</i>		0:1 <i>xs:anyType</i>	Extensions.

8.7.4.11 ContinuousLegStructure

Table 113 — Description of *ContinuousLegStructure*.

<i>ContinuousLegStructure</i>			<i>+Structure</i>	Leg of a trip that is not bound to a timetable.
	LegStart		1:1 <i>+PlaceRef</i>	Start location of this leg. (See 8.4.5.11).
	LegEnd		1:1 <i>+PlaceRef</i>	End location of this leg. (See 8.4.5.11).
	Service		1:1 <i>+ContinuousService</i>	Service of this leg. May be "walk" in most cases, but also cycling or taxi etc. (See 8.4.6.9).
<i>TimeWindow</i>	<i>TimeWindowStart</i>		0:1 <i>xs:dateTime</i>	Earliest possibility to start this leg.
	<i>TimeWindowEnd</i>		0:1 <i>xs:dateTime</i>	Latest time when this leg has to be completed.
	Duration		1:1 <i>xs:duration</i>	Duration of this leg according to user preferences like walk speed.

	<i>LegDescription</i>	0:1	<i>+InternationalText</i>	Title or summary of this leg for overview.
	<i>Length</i>	0:1	<i>LengthType</i>	Length of the leg.
	<i>LegTrack</i>	0:1	<i>+LegTrack</i>	Detailed description of each element of this leg including geometric projection. (See 8.4.6.15).
	<i>PathGuidance</i>	0:1	<i>+PathGuidance</i>	Structured model further describing this interchange, its geographic embedding and accessibility. (See 8.7.4.15).
	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>	Reference to situation message. Message details might be found in TripResponseContext (see 8.7.4.3) or through other communication channels (see 8.4.8.2).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.7.4.12 LegBoardStructure

Table 114 — Description of *LegBoardStructure*.

<i>LegBoardStructure</i>			<i>+Structure</i>	Describes the boarding situation for a passenger leg.
<i>StopPoint</i>	<i>StopPointRef</i>	1:1	<i>→StopPoint</i>	Reference to a stop point (see 8.4.5.1).
	<i>StopPointName</i>	1:1	<i>+InternationalText</i>	Name or description of stop point for use in passenger information.
	<i>NameSuffix</i>	0:1	<i>+InternationalText</i>	Additional description of the stop point that may be appended to the name if enough space is available. Eg: "opposite main entrance".
	<i>PlannedQuay</i>	0:1	<i>+InternationalText</i>	Name of the bay where to board/alight from the vehicle. According to planned timetable.
	<i>EstimatedQuay</i>	0:1	<i>+InternationalText</i>	Name of the bay where to board/alight from the vehicle. As to the latest realtime status.
<i>ServiceArrival</i>	<i>TimetabledTime</i>	0:1	<i>xs:dateTime</i>	Arrival time according to planned timetable.
	<i>RecordedAtTime</i>	0:1	<i>xs:dateTime</i>	Arrival time at the stop point as it was recorded.
	<i>EstimatedTime</i>	0:1	<i>xs:dateTime</i>	Expected/estimated arrival time at the stop point.
	<i>EstimatedTimeLow</i>	0:1	<i>xs:dateTime</i>	Estimated lower limit for arrival time.
	<i>EstimatedTimeHigh</i>	0:1	<i>xs:dateTime</i>	Estimated upper limit for arrival time.
<i>ServiceDeparture</i>	<i>TimetabledTime</i>	1:1	<i>xs:dateTime</i>	Departure time according to planned timetable.
	<i>RecordedAtTime</i>	0:1	<i>xs:dateTime</i>	Departure time at the stop point as it was recorded.
	<i>EstimatedTime</i>	0:1	<i>xs:dateTime</i>	Expected/estimated departure time at the stop point.
	<i>EstimatedTimeLow</i>	0:1	<i>xs:dateTime</i>	Estimated lower limit for departure time.
	<i>EstimatedTimeHigh</i>	0:1	<i>xs:dateTime</i>	Estimated upper limit for departure time.
	<i>MeetsViaRequest</i>	0:1	<i>xs:boolean</i>	This stop fulfils one of the via requirements stated in the request data. Default is <i>false</i> .
<i>StopCallStatus</i>	<i>Order</i>	0:1	<i>xs:positiveInteger</i>	Sequence number of this stop in the service pattern of the journey.
	<i>RequestStop</i>	0:1	<i>xs:boolean</i>	The vehicle journey calls at this stop only on demand. Default is <i>false</i> .
	<i>UnplannedStop</i>	0:1	<i>xs:boolean</i>	This stop has not been planned by the planning department. Default is <i>false</i> .
	<i>NotServicedStop</i>	0:1	<i>xs:boolean</i>	The vehicle will not call at this stop despite earlier planning. Default is <i>false</i> .

8.7.4.13 LegAlightStructure

Table 115 — Description of *LegAlightStructure*.

<i>LegAlightStructure</i>			<i>+Structure</i>	Describes the alight situation for a passenger leg.
StopPoint	StopPointRef	1:1	→StopPoint	Reference to a stop point (see 8.4.5.1).
	StopPointName	1:1	+InternationalText	Name or description of stop point for use in passenger information.
	NameSuffix	0:1	+InternationalText	Additional description of the stop point that may be appended to the name if enough space is available. Eg: "opposite main entrance".
	PlannedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. According to planned timetable.
	EstimatedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. As to the latest realtime status.
ServiceArrival	TimetabledTime	1:1	xs:dateTime	Arrival time according to planned timetable.
	RecordedAtTime	0:1	xs:dateTime	Arrival time at the stop point as it was recorded.
	EstimatedTime	0:1	xs:dateTime	Expected/estimated arrival time at the stop point.
	EstimatedTimeLow	0:1	xs:dateTime	Estimated lower limit for arrival time.
	EstimatedTimeHigh	0:1	xs:dateTime	Estimated upper limit for arrival time.
ServiceDeparture	TimetabledTime	0:1	xs:dateTime	Departure time according to planned timetable.
	RecordedAtTime	0:1	xs:dateTime	Departure time at the stop point as it was recorded.
	EstimatedTime	0:1	xs:dateTime	Expected/estimated departure time at the stop point.
	EstimatedTimeLow	0:1	xs:dateTime	Estimated lower limit for departure time.
	EstimatedTimeHigh	0:1	xs:dateTime	Estimated upper limit for departure time.
	MeetsViaRequest	0:1	xs:boolean	This stop fulfils one of the via requirements stated in the request data. Default is <i>false</i> .
StopCallStatus	StopSeqNumber	0:1	xs:positiveInteger	Sequence number of this stop in the service pattern of the journey.
	RequestStop	0:1	xs:boolean	The vehicle journey calls at this stop only on demand. Default is <i>false</i> .
	UnplannedStop	0:1	xs:boolean	This stop has not been planned by the planning department. Default is <i>false</i> .
	NotServicedStop	0:1	xs:boolean	The vehicle will not call at this stop despite earlier planning. Default is <i>false</i> .

8.7.4.14 LegIntermediateStructure

Table 116 — Description of *LegIntermediateStructure*.

<i>LegIntermediateStructure</i>			<i>+Structure</i>	Describes the intermediate stops of a trip leg.
StopPoint	StopPointRef	1:1	→StopPoint	Reference to a stop point (see 8.4.5.1).
	StopPointName	1:1	+InternationalText	Name or description of stop point for use in passenger information.
	NameSuffix	0:1	+InternationalText	Additional description of the stop point that may be appended to the name if enough space is available. Eg: "opposite main entrance".
	PlannedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. According to planned timetable.
	EstimatedBay	0:1	+InternationalText	Name of the bay where to board/alight from the vehicle. As to the latest realtime status.

ServiceArrival	TimetabledTime	1:1	<i>xs:dateTime</i>	Arrival time according to planned timetable.
	<i>RecordedAtTime</i>	0:1	<i>xs:dateTime</i>	Arrival time at the stop point as it was recorded.
	<i>EstimatedTime</i>	0:1	<i>xs:dateTime</i>	Expected/estimated arrival time at the stop point.
	<i>EstimatedTimeLow</i>	0:1	<i>xs:dateTime</i>	Estimated lower limit for arrival time.
	<i>EstimatedTimeHigh</i>	0:1	<i>xs:dateTime</i>	Estimated upper limit for arrival time.
ServiceDeparture	TimetabledTime	1:1	<i>xs:dateTime</i>	Departure time according to planned timetable.
	<i>RecordedAtTime</i>	0:1	<i>xs:dateTime</i>	Departure time at the stop point as it was recorded.
	<i>EstimatedTime</i>	0:1	<i>xs:dateTime</i>	Expected/estimated departure time at the stop point.
	<i>EstimatedTimeLow</i>	0:1	<i>xs:dateTime</i>	Estimated lower limit for departure time.
	<i>EstimatedTimeHigh</i>	0:1	<i>xs:dateTime</i>	Estimated upper limit for departure time.
	<i>MeetsViaRequest</i>	0:1	<i>xs:boolean</i>	This stop fulfils one of the via requirements stated in the request data. Default is <i>false</i> .
StopCallStatus	<i>StopSeqNumber</i>	0:1	<i>xs:positiveInteger</i>	Sequence number of this stop in the service pattern of the journey.
	<i>RequestStop</i>	0:1	<i>xs:boolean</i>	The vehicle journey calls at this stop only on demand. Default is <i>false</i> .
	<i>UnplannedStop</i>	0:1	<i>xs:boolean</i>	This stop has not been planned by the planning department. Default is <i>false</i> .
	<i>NotServicedStop</i>	0:1	<i>xs:boolean</i>	The vehicle will not call at this stop despite earlier planning. Default is <i>false</i> .

8.7.4.15 PathGuidanceStructure

Table 117 — Description of *PathGuidanceStructure*.

<i>PathGuidanceStructure</i>			+Structure	Description of a piece of a trip. May include geographic information, turn instructions and accessibility information.
	<i>PathGuidanceSection</i>	1:*	+PathGuidanceSection	One or more navigation sections that build the TripLeg. (See 8.7.4.16).

8.7.4.16 PathGuidanceSectionStructure

Table 118 — Description of *PathGuidanceSectionStructure*.

<i>PathGuidanceSectionStructure</i>			+Structure	Building block of a PathGuidance.
	<i>TrackSection</i>	0:1	+TrackSection	Geographic embedding of this navigation section. (See 8.4.6.16).
	<i>TurnDescription</i>	0:1	+InternationalText	Textual description of a Guidance Advice. This should imply the information from <i>GuidanceAdvice</i> , <i>TurnAction</i> und <i>TrackSection.RoadName</i> .
	<i>GuidanceAdvice</i>	0:1	<i>origin / destination / continue / keep / turn / leave / enter</i>	Code for GuidanceAdvice.
	<i>TurnAction</i>	0:1	<i>sharp left / left / half left / straight on / half right / right / sharp right / uturn</i>	Code for turn action.
	<i>DirectionHint</i>	0:1	+InternationalText	Textual direction hint for better understanding, e.g. "follow signs to Hamburg".

	<i>Bearing</i>	0:1	<i>AbsoluteBearing</i>	Absolute bearing after the described manoeuvre.
	<i>AccessPath</i>	0:1	<i>+AccessPath</i>	Description of the type of accessibility on this navigation section. (See 8.7.4.17).
	<i>SituationFullRef</i>	0:*	<i>+SituationFullRef</i>	Reference to situation message. Message details might be found in <i>TripResponseContext</i> (see 8.7.4.3) or through other communication channels (see 8.4.8.2).

8.7.4.17 PathLinkStructure

Table 119 — Description of *PathLinkStructure*.

<i>PathLinkStructure</i>			<i>+Structure</i>	Description of a path in terms of accessibility.
	<i>Transition</i>	0:1	<i>up / down / level / upAndDown / downAndUp</i>	Whether path is up, down or level.
	<i>AccessFeatureType</i>	0:1	<i>lift / stairs / seriesOfStairs / escalator / ramp / footpath</i>	Type of physical feature of path link
	<i>Count</i>	0:1	<i>xs:positiveInteger</i>	Number of times the access feature occurs in this PathLink.

8.8 Service Stop Events

8.8.1 Description

This service provides information on arrivals and/or departures of public transport services from stops for a requested time or period of time. Restrictions can be set in the request parameters that filter the result contents accordingly. Figure 12 illustrates the exchange of information in a stop event enquiry :

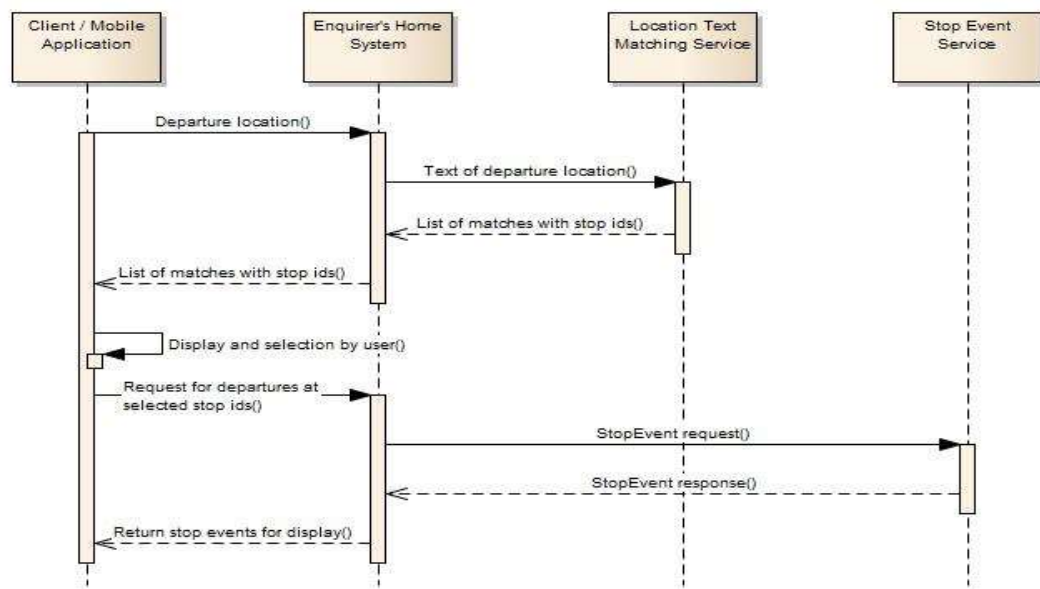


Figure 12 — Stop Event Service

8.8.2 Request Structures

Stop event information (departure or arrival boards) can be requested by sending a ***StopEventRequest*** element (of type *StopEventRequestStructure*).

8.8.2.1 StopEventRequestStructure

Table 120 — Description of *StopEventRequestStructure*.

<i>StopEventRequestStructure</i>			<i>+Structure</i>	Request element for departure and arrival events at stops.
	<i>Place</i>	1:1	<i>+PlaceContext</i>	Place for which to obtain stop event information. (See 8.4.5).
	<i>Params</i>	0:1	<i>+StopEventParam</i>	Specific request parameters. (see 8.8.2.2).

8.8.2.2 StopEventParamStructure

Table 121 — Description of *StopEventParamStructure*.

<i>StopEventParamStructure</i>			<i>+Structure</i>	Contains the request parameters that control the compilation of the departure and arrival events.
<i>StopEventDataFilter</i>	<i>PtModeFilter</i>	0:1	<i>+PtModeFilter</i>	Modes to be considered in stop events. (See 8.4.3.5).
	<i>LineFilter</i>	0:1	<i>+LineDirectionFilter</i>	Lines/Directions to include/exclude. (See 8.4.4.6).
	<i>OperatorFilter</i>	0:1	<i>+OperatorFilter</i>	Transport operators to include/exclude. (See 8.4.4.4).
<i>StopEventPolicy</i>	<i>NumberOfResults</i>	0:1	<i>xs:positiveInteger</i>	Maximum number of events to be returned.
	<i>TimeWindow</i>	0:1	<i>xs:duration</i>	Time window events should lie within. Starting from time given in LocationContext.
	<i>StopEventType</i>	0:1	<i>departure arrival both</i>	Defines if only departures or arrivals or both should be returned.
<i>StopEventContentFilter</i>	<i>IncludePreviousCalls</i>	0:1	<i>xs:boolean</i>	Whether the previous calls of each vehicle journey should be included in the response. Default is <i>false</i> .
	<i>IncludeOnwardCalls</i>	0:1	<i>xs:boolean</i>	Whether the onward calls of each vehicle journey should be included in the response. Default is <i>false</i> .
	<i>IncludeOperatingDays</i>	0:1	<i>xs:boolean</i>	Whether operating day information of this stop event should be included in the response. Default is <i>false</i> .
	<i>IncludeRealtimeData</i>	0:1	<i>xs:boolean</i>	Whether realtime information for each stop event should be included in the response. Default is <i>false</i> .

8.8.3 Response Structures

An element ***StopEventResponse*** of the type *StopEventResponseStructure* is used to respond to a stop events request.

8.8.3.1 StopEventResponseStructure

Table 122 — Description of *StopEventResponseStructure*.

<i>StopEventResponseStructure</i>			<i>+Structure</i>	Response structure for departure and arrival events at stops.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
	<i>StopEventResponseContext</i>	0:1	<i>+StopEventResponseContext</i>	Container for data that is referenced multiple times. (See 8.8.3.2).
	<i>StopEventResult</i>	0:*	<i>+StopEventResult</i>	Enclosing element for stop event data. (See 8.8.3.3).

The following error codes can appear in *ErrorMessage*:

Table 123 — List of possible error codes in *StopEventResponse*.

STOPEVENT_DATEOUTOFRANGE	There are no timetables available for the requested date.
STOPEVENT_LOCATIONUNKNOWN	The location (address, stop etc.) for which stop events have been requested is unknown.
STOPEVENT_LOCATIONUNSERVED	There is no public transport service available at all at the location (address, stop etc.) for which stop events have been requested.
STOPEVENT_NOEVENTFOUND	No departure/arrival could be found within the requested period of time that meets the given restrictions.

8.8.3.2 StopEventResponseContextStructure

Table 124 — Description of *StopEventResponseContextStructure*.

StopEventResponseContextStructure	+Structure (derived from AbstractResponseContextStructure)	Stop event response context. May hold objects that are referenced several times. (See 8.4.6.13).
--	--	--

8.8.3.3 StopEventResultStructure

Table 125 — Description of *StopEventResultStructure*.

StopEventResultStructure			+Structure	Wrapper element for a single stop event result.
	ResultId	1:1	xs:NMTOKEN	ID of this result.
	ErrorMessage	0:*	+ErrorMessage	Result-specific error messages. For possible values refer to the subsequent table. (See also 8.4.4.2).
	StopEvent	1:1	+StopEvent	Stop event data element. (See 8.8.3.4).

The following error codes can appear in *ErrorMessage*:

Table 126 — List of possible error codes in *StopEventResult*.

STOPEVENT_LASTSERVICEOFTHISLINE	This departure/arrival event is the last one of this line for this operating day.
STOPEVENT_NOREALTIME	There is no realtime or forecast data available for this departure/arrival event.

8.8.3.4 StopEventStructure

Table 127 — Description of *StopEventStructure*.

StopEventStructure			+Structure	Details of a single departure/arrival event.
	PreviousCall	0:*	+CallAtNearStop	Calls at stops that happen before this stop event (service pattern of this vehicle journey before this stop event). (See 8.8.3.5).
	ThisCall	1:1	+CallAtNearStop	The call of this vehicle journey at this stop. (See 8.8.3.5).
	OnwardCall	0:*	+CallAtNearStop	Calls at stops that happen after this stop event (rest of the service pattern of the vehicle journey). (See 8.8.3.5).
	Service	1:1	+DatedJourney	DATED VEHICLE JOURNEY that calls at this stop. (See 8.4.6.3).
OperatingDays	OperatingDays	0:1	+OperatingDays	Operating days when this trip can be made. (See 8.4.4.8).

	<i>OperatingDaysDescription</i>	0:1	<i>+InternationalText</i>	Human readable description of the operating days, eg: "Monday to Friday" or "Sunday and holidays".
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.8.3.5 CallAtNearStopStructure

Table 128 — Description of *CallAtNearStopStructure*.

<i>CallAtNearStopStructure</i>			<i>+Structure</i>	A departure/arrival event at a nearby stop.
	<i>CallAtStop</i>	1:1	<i>+CallAtStop</i>	A service that calls at a stop. (See 8.4.6.7).
	<i>WalkDistance</i>	0:1	<i>Distance</i>	Distance from request location (eg: address) to this stop in metres.
	<i>WalkDuration</i>	0:1	<i>xs:duration</i>	Distance from request location (eg: address) to this stop in seconds. All user options taken into account (eg: walk speed).

8.9 Service Trip Information

8.9.1 Description

This service provides information on a single trip (service pattern, real-time status, vehicle facilities etc.).

8.9.2 Request Structures

Trip information can be gathered by using a ***TripInfoRequest*** element (type *TripInfoRequestStructure*).

8.9.2.1 TripInfoRequestStructure

Table 129 — Description of *TripInfoRequestStructure*.

TripInfoRequestStructure			+Structure	Contains the data for a trip information request.	
DatedJourneyRef	a	JourneyRef	-1:1	→Journey	Reference to a Dated Vehicle Journey (see 8.4.4.1).
	a	OperatingDayRef		→OperatingDay	Reference to an Operating Day (see 8.4.4.1).
TimedVehicleRef	b	VehicleRef		→siri:Vehicle	Reference to a vehicle (see 8.4.4.1)..
	b	TimeOfOperation		xs:dateTime	Time stamp when the vehicle is operating. In most use cases equal to "now".
TripInfoRequest	Params		0:1	+ TripInfoParam	Parameter to filter the response contents. (see 8.9.2.2).

8.9.2.2 TripInfoParamStructure

Table 130 — Description of *TripInfoParamStructure*.

<i>TripInfoParamStructure</i>			<i>+Structure</i>	Parameters that filter the trip information response content.
<i>TripInfoPolicy</i>	<i>UseTimetabledDataOnly</i>	0:1	<i>xs:boolean</i>	If set to <i>true</i> then do not return any realtime or incident data. Default is <i>false</i> .
<i>TripInfoContentFilter</i>	<i>IncludeCalls</i>	0:1	<i>xs:boolean</i>	Whether service pattern is to be included. Default is true.
	<i>IncludePosition</i>	0:1	<i>xs:boolean</i>	Whether current position is to be included. Default is true.
	<i>IncludeService</i>	0:1	<i>xs:boolean</i>	Whether service information is to be included. Default is true.
	<i>IncludeTrackSections</i>	0:1	<i>xs:boolean</i>	Whether the result should include TrackSection elements (see 8.4.6.16) to describe the geographic route of each TripLeg. Default is <i>false</i> .
	<i>IncludeTrackProjection</i>	0:1	<i>xs:boolean</i>	Whether the result should include the geographic projection (coordinates) of each TripLeg. Default is <i>false</i> .
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.9.3 Response Structures

An element ***TripInfoResponse*** of the type *TripInfoResponseStructure* is used to respond to a trip information request.

8.9.3.1 TripInfoResponseStructure

Table 131 — Description of *TripInfoResponseStructure*.

<i>TripInfoResponseStructure</i>			<i>+Structure</i>	Response structure for trip information.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
<i>TripInfoResponse</i>	<i>TripInfoResponseContext</i>	0:1	<i>+TripInfoResponseContext</i>	Container for data that is referenced multiple times. (See 8.9.3.2).
	<i>TripInfoResult</i>	0:*	<i>+TripInfoResult</i>	Enclosing element for trip information data. (See 8.9.3.3).

The following error codes can appear in *ErrorMessage* but do not appear in the XSD:

Table 132 — List of possible error codes in *StopEventResponse*.

<i>TRIPINFO_JOURNEYREFUNKNOWN</i>	The journey reference used in the request is unknown.
<i>TRIPINFO_VEHICLEUNKNOWN</i>	The vehicle reference used in the request is unknown.
<i>TRIPINFO_NOJOURNEYFOUND</i>	No matching journey could be found for the requested time and journey/vehicle identifiers.
<i>TRIPINFO_NOGEOINFO</i>	No geographic information available for this vehicle journey.

8.9.3.2 TripInfoResponseContextStructure

Table 133 – Description of *TripInfoResponseContextStructure*

<i>TripInfoResponseContextStructure</i>	<i>+Structure (derived from AbstractResponseContext Structure)</i>	Trip information response context. May hold objects that are referenced several times. (See 8.4.6.13).
--	--	---

8.9.3.3 TripInfoResultStructure

Table 134 – Description of *TripInfoResultStructure*

<i>TripInfoResult Structure</i>	<i>+Structure</i>			Wrapper element for a trip information query result.
	<i>PreviousCall</i>	0:*	<i>+CallAtStop</i>	The stops this service already has called at. Including the current stop if service is currently at stop. (See 8.4.6.7).
	<i>CurrentPosition</i>	0:1	<i>+VehiclePosition</i>	Current position of this service. (See 8.4.6.10).
	<i>OnwardCall</i>	0:*	<i>+CallAtStop</i>	The stops this service still has to call at. (See 8.4.6.7).
	<i>Service</i>	0:1	<i>+DatedJourney</i>	DATED VEHICLE JOURNEY that operates this trip. (See 8.4.6.3).
<i>OperatingDays</i>	<i>OperatingDays</i>	0:1	<i>+OperatingDays</i>	Operating days when this trip can be made. (See 8.4.4.8).
	<i>OperatingDaysDescription</i>	0:1	<i>+InternationalText</i>	Human readable description of the operating days, eg: "Monday to Friday" or "Sunday and holidays".
<i>ServiceFacilityGroup</i>	:::	0:1	<i>siri:ServiceFacilityGroup</i>	Vehicle and service facilities (see 8.4.7.3).
	<i>JourneyTrack</i>	0:1	<i>+LegTrack</i>	Geographic embedding of this journey. (See 8.4.6.15).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.10 Service Tickets and Fare Calculation

8.10.1 Description

This service provides general, stop-specific and trip-specific fare information.

8.10.2 Request Structures

Ticket and fare information can be gathered by using a ***FareRequest*** element (type *FareRequestStructure*).

8.10.2.1 FareRequestStructure

Table 135 — Description of *FareRequestStructure*.

FareRequestStructure			+Structure	Contains the data for a fare request.	
	a	StopFareRequest	-1:1	+StopFareRequest	Request for stop-specific fare information. (See 8.10.2.2).
	b	StaticFareRequest		+StaticFareRequest	Request for general/static fare information. (See 8.10.2.3).
	c	TripFareRequest		+TripFareRequest	Request to calculate the fare information for a single trip. (See 8.10.2.4).
	d	MultiTripFareRequest		+MultiTripFareRequest	Request to calculate aggregated fare information of multiple single trips. (See 8.10.2.5).
	Params		0:1	+ FareParam	Parameter to filter the response contents. (see 8.4.10.9).
	Extension		0:1	xs:anyType	Extensions.

8.10.2.2 StopFareRequestStructure

The stop-specific fare request seeks for fare information that is valid for a specific stop, eg: the tariff zones in which this stop is located.

Table 136 — Description of *StopFareRequestStructure*.

<i>StopFareRequestStructure</i>			<i>+Structure</i>	Contains the data for a stop-specific fare request.
	<i>StopPointRef</i>	1:1	<i>→StopPointCode</i>	References the stop point. (See 8.4.5.1).
	<i>Date</i>	0:1	<i>xs:date</i>	Date for which to retrieve fare information.

8.10.2.3 StaticFareRequestStructure

The static fare request seeks for general fare information as for example a list of available fare products or a URL of pages with further fare content (tariff zone maps, fare terms and conditions etc.).

Table 137 — Description of *StaticFareRequestStructure*.

<i>StaticFareRequestStructure</i>			<i>+Structure</i>	Contains the data for a general (static) fare request.
	<i>Date</i>	0:1	<i>xs:date</i>	Date for which to retrieve fare information.
	<i>FareProductRef</i>	0:*	<i>→FareProductCode</i>	Codes of the fare products for which more information is sought. If no <i>FareProductRef</i> is specified the responding system should reply with information for all available fare products.

8.10.2.4 TripFareRequestStructure

The trip-specific fare request seeks information on fare products and their prices that are valid when making this trip.

Table 138 — Description of *TripFareRequestStructure*.

<i>TripFareRequestStructure</i>			<i>+Structure</i>	Contains the data for a trip-specific fare request.
	<i>Trip</i>	1:1	<i>+Trip</i>	Contains a complete trip from origin to destination. (See 8.7.4.6).

8.10.2.5 MultiTripFareRequestStructure

The difference between *MultiTripFareRequestStructure* and *TripFareRequestStructure* lies in the point that the server – in case of a *MultiTripFareRequestStructure* – is requested to find a fare product combination covering all trips that is as economic as possible, for example, a day fare product if enough trips are on the same day.

Table 139 — Description of *MultiTripFareRequestStructure*.

<i>MultiTripFareRequestStructure</i>			<i>+Structure</i>	Contains the data for a fare request for multiple trips.
	<i>Trip</i>	1:*	<i>+Trip</i>	Contains the trips for which fare information is to be retrieved. (See 8.7.4.6).

8.10.3 Response Structures

An element *FareResponse* of the type *FareResponseStructure* is used to respond to a fare information request.

8.10.3.1 FareResponseStructure

Table 140 — Description of *FareResponseStructure*.

<i>FareResponseStructure</i>			<i>+Structure</i>	Contains the response data for a fare request.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Error messages related to the processing of the entire request. For possible values refer to the subsequent table. (See also 8.4.4.2).
	<i>FareResult</i>	0:*	<i>+FareResult</i>	Fare result choice element. (See 8.10.3.2).

The following error codes can appear in *ErrorMessage*:

Table 141 — List of possible error codes in *FareResponse*.

<i>FARES_DATEOUTOFRANGE</i>	The fare request cannot be processed because there is no information available for the requested date.
<i>FARES_STOPPOINTUNKNOWN</i>	The fare request cannot be processed because the requested stop is unknown.

8.10.3.2 FareResultStructure

Table 142 — Description of *FareResultStructure*.

<i>FareResultStructure</i>			<i>+Structure</i>	Wrapper element for fare results.
	<i>ResultId</i>	1:1	<i>xs:NMTOKEN</i>	ID of this result.
<i>a</i>	<i>StopFareResult</i>	-1:1	<i>+StopFareResult</i>	Response to stop-specific fare request. (See 8.10.3.3).
<i>b</i>	<i>StaticFareResult</i>		<i>+StaticFareResult</i>	Response to general (static) fare request. (See 8.10.3.4).
<i>c</i>	<i>TripFareResult</i>		<i>+TripFareResult</i>	Response to trip-specific fare request. (See 8.4.10.7).
<i>d</i>	<i>MultiTripFareResult</i>		<i>+MultiTripFareResult</i>	Response to fare request for multiple trips. (See 8.10.3.6).

8.10.3.3 StopFareResultStructure

Table 143 — Description of *StopFareResultStructure*.

<i>StopFareResultStructure</i>			<i>+Structure</i>	Result structure for stop-specific fare information.
	<i>TariffZoneListInArea</i>	1:*	<i>+TariffZoneListInArea</i>	One or more lists of tariff zones that are associated with the stop. (See 8.4.10.3).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.10.3.4 StaticFareResultStructure

Table 144 — Description of *StaticFareResultStructure*.

<i>StaticFareResultStructure</i>			<i>+Structure</i>	Result structure for general (static) fare information.
	<i>FareProduct</i>	0:*	<i>+FareProduct</i>	One or more fare products s that are available for this area. (See 8.4.10.6).
	<i>StaticInfoUrl</i>	0:1	<i>+WebLink</i>	URL of information page on the web (see 8.4.2.4).
	<i>Extension</i>	0:1	<i>xs:anyType</i>	Extensions.

8.10.3.5 TripFareProductReferenceStructure

Table 145 — Description of *TripFareProductReferenceStructure*.

<i>TripFareProductReferenceStructure</i>			<i>+Structure</i>	Structure that connects fare products to trips (or parts of them).
	<i>FareProductRef</i>	1:1	<i>→FareProductCode</i>	Reference to a fare product.
	<i>FromTripIdRef</i>	1:1	<i>xs:NMTOKEN</i>	Identifies the "valid from" trip which is the first trip of a set of trips for which the fare product is valid.
	<i>FromTripLegIdRef</i>	0:1	<i>xs:NMTOKEN</i>	Identifies the "valid from" tripLeg. If missing, then valid from the first leg.
	<i>ToTripIdRef</i>	1:1	<i>xs:NMTOKEN</i>	Identifies the "valid to" trip which is the last trip of a set of trips for which the fare product is valid.
	<i>ToTripLegIdRef</i>	0:1	<i>xs:NMTOKEN</i>	Identifies the "valid to" tripLeg. If missing, then valid to the last leg.

8.10.3.6 MultiTripFareResultStructure

Table 146 — Description of *MultiTripFareResultStructure*.

<i>MultiTripFareResultStructure</i>			<i>+Structure</i>	Contains the result data of fare information for multiple trips.
	<i>ErrorMessage</i>	0:*	<i>+ErrorMessage</i>	Result-specific error messages. For possible values refer to the subsequent table. (See also 8.4.4.2).
	<i>TripFareProductReference</i>	1:*	<i>+TripFareProductReference</i>	Non-empty list of trip references in the corresponding <i>MultiTripFareRequestStructure</i> . (See 8.10.3.5).
	<i>FareProduct</i>	0:*	<i>+FareProduct</i>	Zero, one or more fare product s that are valid for the referenced trips / part of trips (see 8.4.10.6).
	<i>PassedZones</i>	0:1	<i>+TariffZoneListInArea</i>	The set of passed zones – counted over all trips (see 8.4.10.3).
	<i>StaticInfoURL</i>	0:*	<i>+WebLink</i>	URL of fare information pages on the web (see 8.4.2.4).

The following error codes can appear in *ErrorMessage* but do not appear in the XSD:

Table 147 — Error codes in *ErrorMessage*

<i>FARES_OUTOFAREA</i>	The trip found by the journey planning leaves the area where the fare are valid.
<i>FARES_JOURNEYNOTPERMITTED</i>	A service used in the trip result is not permitted by this fare scheme.
<i>FARES_ADDITIONALCHARGES</i>	Additional fees (eg: tolls or booking fees) are to be expected.
<i>FARES_ADDITIONALTICKETS</i>	Additional tickets are necessary because suitable tickets could not be found for all services or trips in the request.
<i>FARES_ROUTENOTFEASIBLE</i>	No ticket can be suggested because the route of the trip does not comply with the fare terms (eg: in case of round-trips, dead-end trips or exceeding the permitted total duration).

8.11 Discovery and Capabilities

It is important to note that the OJP fully relies on SIRI's communication protocol (fully described in EN 15531-2:2015 - Part 2: Communications). The following paragraphs therefore show how the SIRI approach has been copied or modified in respect of the OJP requirements.

Once there are many OJP systems covering a large number of areas, it will be necessary to allow systems to obtain the reference data that needs to be used. In the SIRI model Service Discovery has been broken down into three steps: (i) Universal Server Discovery; (ii) SIRI Capability Discovery; and (iii) SIRI Functional Service Coverage Discovery.

The first stage in Universal Server Discovery is to find the servers and sites supporting the SIRI protocols. This was not identified as a SIRI requirement, and the same choice has been made accordingly for OJP. If needed, general purpose web service discovery protocols such as WSDL are likely to be sufficient. The expectation is that OJP systems will reach bilateral or multi-lateral agreements to exchange data between each other – and this process will involve identification of all relevant servers and access arrangements for each of them.

Once a SIRI server is known, a client needs to know which SIRI Functional Services and features it supports, and at which version level. SIRI defines a simple Capability Discovery message, which allows a client to obtain this information from each SIRI Producer Service. The service returns a fixed list of the capabilities supported by the service, and the configuration details. But in SIRI implementing a CapabilityDiscoveryResponse is not mandatory: in the context of OJP it was decided not to implement such a Capability Discovery Service, but to systematically rely on the definition of local agreements or profiles (see EN 15531-1:2015 - Part 1: Context and framework – Annex A for more information about how to set up such an agreement).

Once a client knows what services a SIRI server can provide, it may wish to know what data it holds, and what access rights it has to the data and services. Application data discovery services are specific to the application content of the SIRI Functional Services. In the context of OJP, some service are already designed to provide such kinds of discovery (for example Object Information Service or Location Text Matching), and when more information is necessary (such as a detailed physical description of stops, or line, route and journey pattern structures, etc.), then the OJP recommendation is to use the alternative solution proposed by NeTeX to exchange any type of reference data, including stops, lines, etc. NeTeX data objects may also be exchanged using the SIRI protocol, using a NeTeX ***DataObjectRequest*** & ***DataObjectDelivery*** (see the NeTeX specification). A SERVICE FRAME is used to group SCHEDULED

STOP POINTs and LINEs. A RESOURCE FRAME is used to group a set of Product categories or other codes.

Annex A (informative)

A distributed approach to journey planning across Europe

This Annex outlines how one or more **distributed journey planning systems** might link local, regional or national journey planning services into one or more networks of journey planners across Europe.

This approach will enable **incremental development of journey planning capabilities across Europe** as each nation or region creates local journey planning systems that are able to participate in such a network or networks. Users of these local systems would be able to seek information about trips to, from or wholly beyond their own boundaries and **receive that information in their own languages and in familiar ways on the systems that they use for local journey planning**.

Schematically the arrangement for each partnership of journey planning systems can be seen in the following simple diagram in Figure A.1 — Typical configuration of distributed journey planning system resources.

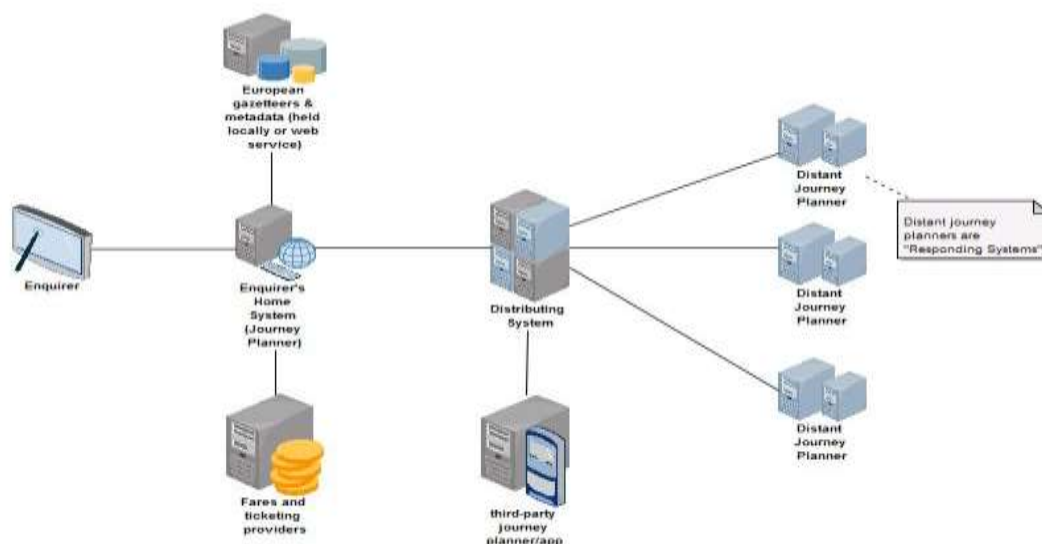


Figure A.1 — Typical configuration of distributed journey planning system resources

*"Distant journey planners" in this diagram are referred to as **responding systems** in this document, whilst the Enquirer's **home system** and **distributing system** can be integrated in a single system or co-located.* Each enquirer would use their local journey planner – and this in turn will call for information from one or more distant journey planners as necessary (the three shown in the diagram are indicative of a large number of distant systems participating in the network). The communication between the individual journey planners would use a standardised open journey planner API.

The local journey planners that would be relevant to participate in such a network are those which themselves hold and process stop and timetable data to create itineraries. So in areas where there is an existing distributed journey planning system, the participating systems would be those which themselves contribute to their national information service. The process can be simplified if several regions have already merged their data into a shared journey planning system. In this way any

European network of journey planners does not face the overheads of two distributed journey planning systems for some enquiries – each component section of an enquiry (as described below) should be directed to a system which itself can calculate and return the required information in response.

The network and timetable information for trunk (long-distance) modes (air, ferry, rail and coach) should be made available to local journey planning systems wherever possible so that these local systems can plan trips on all modes within their own territories – and to and from stations, air and ferry ports across the whole of Europe. Some, if not all, of the trunk journey planning information might be collated centrally and made available to be implemented on all local systems – so that local systems themselves can plan the “local plus trunk” elements of any trip without reference to distant journey planners. If this is the case then the distant journey planners would only be requested to provide the “distant local” element of each trip plan – and this also simplifies the distributed journey planning process.

The proposed approach will also establish a basis for other journey planning providers to create user interfaces to the service which may focus on the special needs of particular types of enquirer – so adding specialist channels to the information which might not otherwise be available. For instance an organisation concerned with the interests of those with impaired vision might create a user interface that is particularly suited to its own clients’ requirements – and then tap into the network of journey planners to provide an information service for those clients in a way they would find relatively easy to access.

The journey planning process involves certain key stages :

- The enquirer specifying the start and end points of a trip (along with the required time and date of departure or arrival) in a free-format using the enquirer’s own native language
- The local planning system matching the enquirer’s requested locations with locations understood by the relevant journey planning systems – including asking the enquirer to resolve any ambiguity in the possible matches
- The local planning system establishing how the requested journey can be planned where one end or both ends are outside the direct scope of the local journey planner being used by the enquirer
- The local planning system (the distribution system) sending elements of the trip request to one or more relevant “distant” journey planners and receiving a response from each of those planners (the responding systems)
- The local planning system integrating those responses with locally generated information if relevant to create seamless trip plans that meet the enquirer’s request
- The local planning system presenting the results to the enquirer in a format that is familiar to the enquirer and in the native language of the enquirer.

To facilitate this process some common data resources will be required to support the network of journey planners. In particular there will be a need for comprehensive gazetteers for all of the areas served by the network – ideally including localities (cities, towns, districts and villages), road names and street addresses, postcodes, Points of Interest, and the names of public transport stops and stations. Such data could be held either as a centrally collated and distributable data resource that can be searched locally by each participating system – or it could be provided as a centralised web service enabling each participating system to interrogate a continually updated set of gazetteers that cover all areas currently participating in the linked services. As new regions join the network of participating systems, so additional gazetteer data would be added for those new areas. Powerful searching and matching routines (such as those which already exist for services such as Google or Bing maps) will be

necessary to pinpoint rapidly the enquirer's intentions and match these to the most appropriate one or more items held in the gazetteers.

Although the User Interface for each of the local journey planners would remain substantially unchanged from their current layouts, it may be necessary or at least beneficial to ask the enquirer to indicate whether their trip request is for a trip wholly within the area covered by the local planner, or for a trip to, from or between points which are elsewhere in Europe. This would be no more than a single tick-box added to existing layouts.

Once the enquirer's trip's start and end points are understood by the journey planning system – which may require the user to select from potential matches found in the gazetteer search – the system will require a mechanism to identify how best to plan the requested trip. Metadata held with the gazetteers should indicate which journey planning systems are aware of each location in the gazetteer – and should indicate whether the information held for that location on a particular system is comprehensive or only selective (as might be the case for a location just outside the formal border of a particular local journey planner). System designers will need to consider how to handle situations where more than one source claims to hold comprehensive information to meet a particular requirement. Some process (shown in the diagram as the Distributed Journey Planning Controller (the distribution system) probably implemented within each Local Journey Planner) will then need to be followed to allow the requested trip to be split into component parts which can be planned by individual local journey planners (responding systems) in a way that will allow the separate legs of the trip to be linked together into a seamless itinerary. Procedures for doing this have already been demonstrated in the DELFI, EU-Spirit and JourneyWeb distributed journey planning systems (see Annex B). This Technical Specification for an Open API for distributed journey planning builds on the knowledge and experience of the systems that have already used a distributed approach. If a collation of Europe-wide trunk timetable information can be made available for implementation on all local journey planners, then each local planner will be able to plan “local + trunk” legs of a trip, and the distributed journey planning task would be simplified to integrate only the “distant local” element of any trip.

Once the method of splitting the enquiry has been determined, then each part of the enquiry will need to be sent to the relevant distant journey planner. Work in Germany has shown that both DELFI and EU-Spirit journey planning systems can (and have already) shared a single API for the question-and-answer process involved. The proposed Open API will ensure that each participating system is required to implement only one API for enquiries by external journey planners – thereby keeping the barriers to entry for each system to a minimum. That one API might be used not only for a European distributed journey planning system, but might equally be used to provide access to the local journey planner for third party information services which seek to offer journey planning advice – such as apps or third-party web sites.

Where systems already have their own APIs those existing APIs could co-exist with the proposed Open API. The existing APIs in these cases may already be optimised for their existing functions and there is no need to make changes to them. Instead the Open API can sit alongside any such existing APIs, and be used for exchanges with other European journey planners.

The proposed Open API will lead to information being collected by the enquirer's journey planning system from other distant systems – so the final two stages of the process are to bring the information together from the different sources in order to create seamless itineraries that meet the enquirer's requirements – and then to present those itineraries in the method used by that host journey planner for all its enquiries ... offering familiarity to the enquirer, and using the enquirer's native language

Maps are a key component of modern journey planning systems, and there are now several options for incorporating maps within both the enquiry and the results parts of this process, either using multi-national services such as Google, Bing or Open Street Map, or by drawing on local map resources from

each area and establishing a data transfer protocol to enable such local maps to be used seamlessly in the presentation of enquiries and results on distant systems. Ideally it should be possible for a user to establish the point of origin and/or destination of a requested trip by input of a topographic place name – and then refining that by zooming into a relevant area map and selecting the points by clicking on the relevant map(s). When itineraries are presented, maps appropriate to each segment of the itinerary should be made available – including map data from the journey planning systems which should be overlaid onto relevant local base maps for the area concerned. This should be able to display both an overview of the entire itinerary down to the detail at each interchange location (showing stop positions and any connecting walk legs between arrival and departure points at the interchange). It may be that local interchange maps will be needed to supplement those of a universal mapping system – in which case these should be drawn down from the relevant local journey planning system through a hyperlink.

Fares and ticketing. Integrating fares and ticketing may be a second stage of development of a Europe-wide service, as there are few examples of such information services existing, at least in areas where fares are commercially determined (such as in the UK). Even then it is probably best to think in terms of creating simple and efficient links between each network of journey planning services and a complementary set of ticketing agencies (which may be transport operators or third parties). This is because the necessary commercial arrangements and security considerations would make it very difficult and expensive to set up and maintain ticketing facilities for every participating system associated with a network of journey planning services. By providing almost seamless links between journey planners and a multitude of ticketing agencies, this avoids such problems and provides an open market for ticket purchase to the extent that relevant transport operators can offer ticketing for their services either directly or through third party agents. Users of the journey planning services will need information about the capabilities of each sales channel – some may only offer ticketing for specific operators' services, whilst others might be able to offer a wider-scope of seamless ticketing. For door-to-door trips, however, there are many circumstances in which the enquirer will not want to buy a ticket for the complete trip that they have just planned – for instance, they may already have a ticket that is valid for part of the trip. And there will be many situations where no on-line ticket options exist for certain legs of a trip. So separation of journey planning from ticketing is a key and positive feature of the suggested arrangements. Although at first sight this may fail to meet the aspiration for a totally seamless journey planning and ticketing service, in practice it offers a route that has been seen to work already. It is one that can be almost seamless to the enquirer, but still leave the enquirer in control of their ticket purchase requirements. It should be possible to offer seamless links between all the journey planning services and relevant ticketing services using a standard hand-off API from the journey planners to the ticketing services – but this API is not part of the proposed Open API for distributed journey planning.

Other information. Many local journey planning systems offer information in addition to journey planning itineraries – for instance timetables for individual PT services, or detailed diagrams illustrating major interchanges – which may also be of interest to those using one of the participating partner journey planning services. It is suggested that access to such additional information should be made available by drawing the relevant document down using a hyperlink from the third party system – thereby ensuring that the information offered to the enquirer is the most up to date that is available. Each contributing travel information system would have its own range of available information that could be offered to users of the journey planning network – so the links to end-users on each journey planner should reflect the information that can be obtained from a specific source. Some information – such as real-time departure information – may not be relevant because the latency of that information is such that it may not be relevant some distant journey planning enquiries. However users may be making an enquiry from their own home system when they are in a different journey planning system's area, when immediate real-time information would be useful to them. In any event, longer-term incident information affecting journey planning, if available, should be built into journey planning responses if possible.

Summary

The vision in this Annex is for an Open API to support an incrementally developing network of local journey planners which can provide distributed journey planning across Europe and, where possible, to offer seamless links to third-party ticketing services. By adopting this approach, the central overheads can be kept to a minimum, whilst an initial distributed journey planning network could be operational very quickly, thereby creating immediate impact and providing the foundations for extension to achieve complete coverage of Europe as and when more local journey planning systems are available to support this.

Annex B (informative)

Lessons from experiences of Distributed Journey Planning to date

B.1 Introduction

To date (before 2015) there have been three significant European examples of Distributed Journey Planning:

- EU-Spirit – covering parts of Germany, France, Luxembourg, Sweden, Denmark, Poland
- DELFI – covering the whole of Germany
- JourneyWeb – covering Great Britain (England, Wales and Scotland)

Each has operated for around 10 years between 2003 and 2015, evolving over that period as circumstances have changed. The following paragraphs seek to highlight some of the essential differences in the ways that each of these has worked. The descriptions are necessarily at a high level and describe the broad principles that were followed in each case. Each of the systems included many more detailed processes that are not described here to ensure that the results were credible and appropriate to the needs of the enquirer.

B.2 JourneyWeb

JourneyWeb was developed through a British research project and was the XML schema adopted by the British Transport Direct service. It had been developed originally to allow collaboration between journey planning systems of individual local authorities – but was seen to be very appropriate for the national Transport Direct service based around 11 regional information services covering the whole of Great Britain (England, Scotland and Wales). The schema provided a range of functions including distributed journey planning which (like the Open API of this Technical Specification) could be used in various architectures. For Transport Direct the architecture chosen was to have a central Coordinated Journey Planner (CJP) which offered the public enquiry and response interface. An enquiry received by the CJP was analysed to determine the traveline region(s) in which the requested trip's origin and destination were located – and this then determined which region would receive requests for components of the required trip plan.

The architecture was supported by two key national databases that were compiled from the data of the 140 or so local transport authorities :

- NPTG – National Public Transport Gazetteer : a database of all publicly recognisable localities (cities, suburbs, towns, villages and hamlets), with a geocoded centroid, and each associated with its relevant administrative area
- NaPTAN – National Public Transport Access Nodes : a database of individual public transport stop points (including clustering of stop points) with each location geocoded and attached to a specific topographic place and administrative area

These two databases were the bedrock of the national information system, to which each public transport schedule was related through its stops and their respective localities. A third database was

then required for the distributed journey planning system itself – comprising two sets of Exchange Points :

- Trunk Exchange Points, being the points in each region where long-distance passengers are likely to change from trunk (long-distance) services (train or coach) from or to local services (which could include all forms of public transport, including trains and coaches)
- Adjacent Region Exchange Points being points along regional boundaries which were used to deal with trips that, whilst crossing into another region, were so "local" that they did not need to use a trunk public transport leg.

Each exchange point was nominated to serve a set of localities (with overlaps between these sets where relevant).

During Transport Direct's operational life of more than 10 yrs the basic method of planning a trip remained broadly constant. A trip that was between points that were both within a single region simply required an enquiry to be sent to that region – and the results from that enquiry were displayed to the enquirer. For a trip between points in two different regions, however, the CJP used the Exchange Points database to determine how to split the enquiry into three components – a local leg, a trunk leg and a distant local leg ... and then, recognising the need to reduce complexity of calculation, it then asked the region with the greater number of exchange points to calculate the relevant local and trunk legs of the trip – and asked the other region to calculate the other local leg of the trip. The two sets of results were then combined to show a set of optimised seamless trip plans.

When Transport Direct launched it was making enquiries of 10 regional traveline information systems – one of which covered both London and the surrounding South East of England region. Over time it became possible to reduce the number of separate regional information systems by sharing data (facilitated by the fact that several adjacent regions had the same system supplier). By the time that Transport Direct ceased in 2014 the number of regions had been reduced to just three through this organic process – and Transport Direct's own demise came about because it became possible to collate all national timetable data into a single dataset (the traveline national dataset, TNDS) – which has been made available as open data and allowed others to deliver public transport information for the whole of Great Britain through several different nationwide systems.

Transport Direct also ran its own rail journey planner which was updated every day (rather than every week, as was the case with the traveline services). This was used to re-plan rail legs of trips to ensure that any short-term plan changes in timetables were reflected in solutions offered. Where necessary a replacement rail leg in a trip would trigger a re-plan of subsequent legs of the overall trip before the solution could be offered.

Although there is no limitation in principle on the modes of transport that could be covered by the JourneyWeb protocol, in practice internal air services generally were not included in the Transport Direct offering (other than trips wholly within Scotland, reflecting the importance of air services to get to and from some of the Scottish islands).

Overall the system was a hybrid between a centralised system and a peer-to-peer one. The system had an identity of its own (Transport Direct) and enquiries were made directly to that system, which then managed the distributed journey planning process by calling for information from the regional information systems of traveline. The traveline information systems themselves were passive in this process in that they delivered information to Transport Direct whilst directly providing information for their own regions to their own enquirers (and nationally only to and from rail stations and coach stops in the other regions). This was not the original vision for JourneyWeb – where it had been envisaged as a way of allowing each local information system to interact with others on a peer-to-peer basis – but

the protocol in the schema was capable of supporting both approaches, and the one adopted by Transport Direct had the benefits of being a single implementation driving the whole process.

B.3 EU-Spirit

EU-Spirit came out of a European Project and was being developed at much the same time as JourneyWeb. For much of its life it was a CORBA-based protocol, and only converted to being XML-based in 2013 (at the same time as DELFI, which allowed the schema for the two protocols to be merged).

EU-Spirit relies on a central hub to plan the long-distance part of a trip initially, and it then seeks plans for the local legs at each end of the overall trip from the respective local systems before returning the seamless trip plans to the enquiring system (which in turn presents them to the enquirer).

Participation in EU-Spirit has varied over the years, with a constant hub involving north Germany and Sweden and Denmark – but with some other German Bundesländer, Austria, Poland, Luxembourg, parts of France having been involved at different times.

The central hub journey planner contained timetable data for air, rail and coach services, or could access such information from separate journey planners. In part this reflects the reluctance of some transport operators to share their timetable data at the time the service was developed.

As with JourneyWeb the most important special set of data is that which defines the "transition points" between each of the various journey planning systems. Other data requirements are also very similar to those of JourneyWeb. Unlike Transport Direct's implementation of JourneyWeb, however, EU-Spirit relies on its contributing local systems to provide the public interface ... so enquiries are made using the same journey planner as a user would go to for a local public transport trip plan ... and that local system calls on the central resources of EU-Spirit to create the response for the trips which go into the areas of other planning systems.

The central computer in EU-Spirit (referred to as the RING) offers three key services :

- RODI – Ring Origin Destination Identifier. The RODI tries to match the user input for origin and destination locations. In order to do so, the RODI contacts the appropriate local (passive) servers.
- RCC – Ring Connection Composer. This RCC acts as a super connection composer and retrieves and combines the partial information by open interfaces to the formerly isolated information systems that now provide information through their local (passive) servers.
- RRDB – Ring Reference Database. In this database – among others – all transition locations are stored. Transition locations are points (typically stations or airports) where an interchange between two local systems or between a local system and an inter-regional system (e.g. national railway) is possible.

B.4 DELFI

DELFI is a more recent Germany-wide distributed journey planning system, linking the services of all 16 federal states. Originally implemented using CORBA in 2004 it was converted in 2009 to a SOAP Web Service based on a XSD/WSDL specification that has also been used by EU-Spirit since then.

Like EU-Spirit, DELFI is integrated within the participating local journey planning systems – so enquirers use a system they are familiar with to trigger enquiries for trips that extend beyond the geographical scope of their local planning systems. A central set of metadata (which is distributed to

each participating system for use in its "search controller") identifies the journey planning systems which cover each topographic place in Germany, so that each local system can identify the appropriate source of information for the long-distance and "the other end" legs of the trip plan. Unlike Transport Direct, the trunk (mainly rail) trip information is planned separately by a German rail planner – so each trip plan typically comprises three sections that need to be integrated (local – trunk – distant local), with that integration being undertaken by the local journey planner being used by the enquirer.

B.5 APII-SIM

The APII-SIM research project in France aims to:

- develop an advanced distributed journey planner, learning from the existing ones, and benchmark it as a proof of concept (POC)
- design a solution's API : enabling any local journey planner (responding system) to communicate with each other according to a shared and open protocol

APII-SIM is an innovative distributed journey planner solution in many ways:

- its architecture is designed in such way that APII-SIM is fairly agnostic about local journey planners: the concept of a "long distance journey planner" is no longer relevant
- its architecture enables several responses for a single request. For a long distance trip (e.g. from Paris to Nice), therefore it is possible to compare the proposed trip and then get a rail or flight or even car-sharing solutions;
- its API splits search service and localization service as two independent services (that could be delivered by distributed architecture as well as centralised architecture)
- its API enables a more responsive feedback to requesting clients who receive a partial result quickly and then a fully detailed trip

From April to June 2015, a Proof Of Concept (POC) has been deployed for an experiment. The POC has connected several journey planners :

- the Oise County
- the Alsace region
- the Champagne-Ardenne region
- the Ile de France region
- the SNCF trains

These existing journey planners have been requested through the common APII-SIM Generic Journey Planner API. An acceptance test plan has focused on trip search service (and not on localization service). The quality and performance measured led the project to validate the relevance of a distributed journey planner architecture. Test analysis feedback has defined the milestones to be followed in order to launch an industrialization phase before opening a service at a national scale.

B.6 Shared journey planning methodology

The DELFI Documentation (v5.00) illustrates the methodology used by all of the distributed journey planners. The planners use a graphical approach to searching for the optimum trip plan applying Dijkstra's algorithm. The figures below illustrate

- Figure B.1 shows how a trip can take many different routes, from which the optimum options need to be found
- Figure B.2 shows how various options can be shown graphically to assist the processing of finding the optimum combination
- Figure B.3 presents a sequence of three diagrams that show the forwards, backwards and repeated forwards calculations that are required to be certain that the solution is optimal

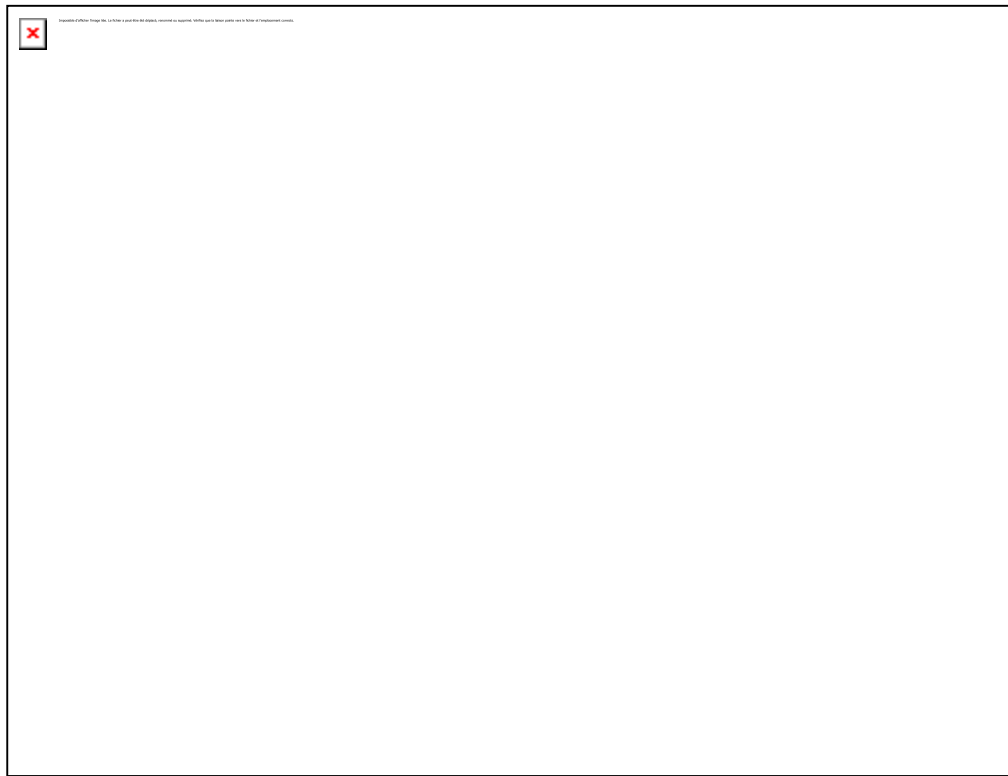


Figure B.1 — The local – trunk – local elements of a trip plan (from DELFI Documentation, v5.00)

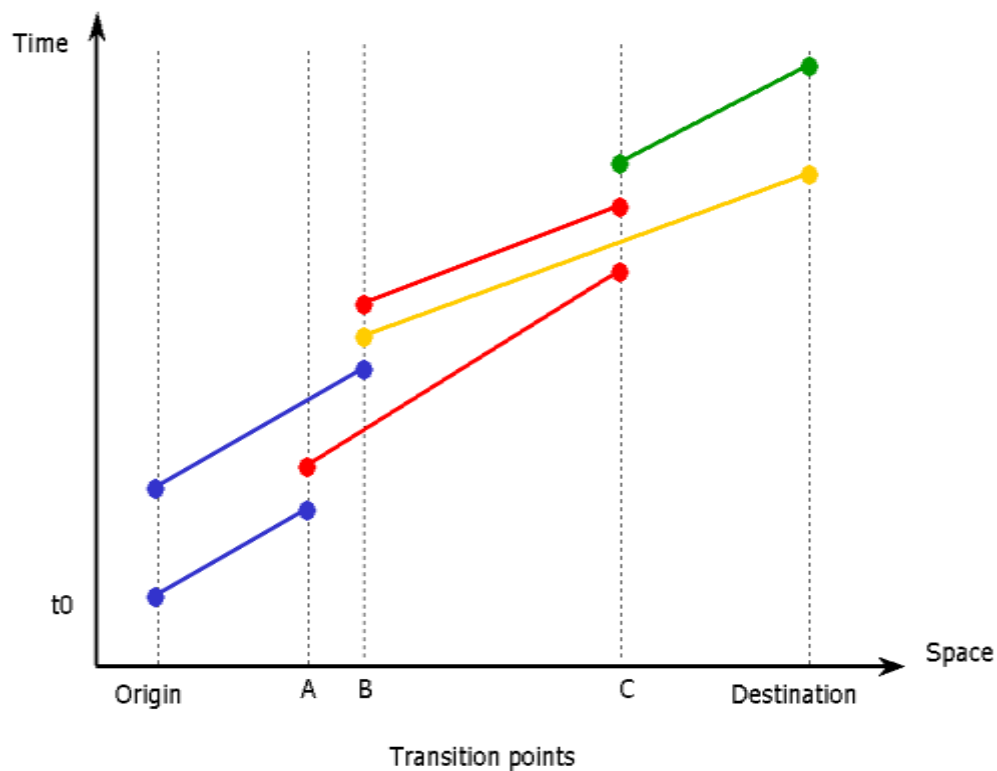


Figure B.2 — Alternative routes with different travel times showing why forward and backward planning is necessary to optimise the solution (from DELFI Documentation, v5.00)

"Transition points" in the above Figure are referred to as "Exchange Points" in this document.

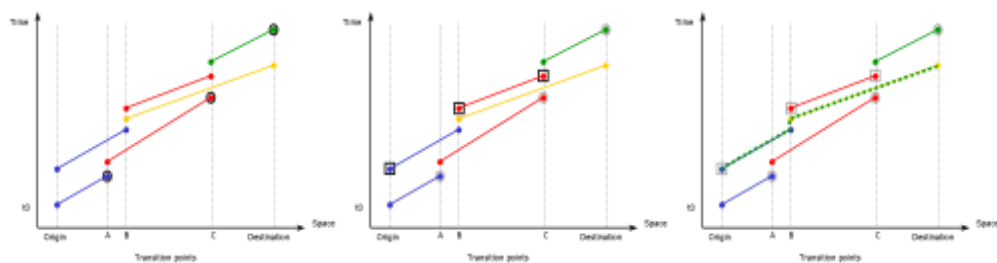


Figure B.3— The three stages of journey planning – forwards, backwards and forwards again (from DELFI Documentation v5.00)

Annex C : Bibliography

- [1] ISO/IEC 19501-1:2002, Unified Modelling Language (UML) – Part 1: Specification
- [2] T Kämpke and M Schaal: Distributed Generation of Fastest Paths, Proceedings of the 10th IASTED International Conference “Parallel and Distributed Computing and Systems”, October 28-31, 1998, Las Vegas, Nevada, USA, p172-177
- [3] DELFI
<http://www.delfi.de>
http://www.delfi.de/sites/default/files/library_entries/Delfi5Doc_v1_0.pdf
- [4] EU-Spirit
<http://eu-spirit.eu/>
- [5] JourneyWeb : Downloads and Schema
https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/365243/journeyweb-downloads.pdf
- [6] TRIAS
<https://www.vdv.de/ip-kom-oev.aspx>
<https://www.vdv.de/trias-xsd-v1.0.zipx?forced=true>
<https://www.vdv.de/vdv-430-mobile-kundeninformation-im-oev.pdf?forced=true>
<https://www.vdv.de/vdv-431-1-ekap-systemarchitektur.pdf?forced=true>
<https://www.vdv.de/vdv-431-2-ekap-schnittstellenbeschreibung.pdf?forced=true>
- [7] OJP : Schema Download
<http://www.vdv.de/ojp>