IXSI - Interface for X-Sharing Information Version 4

Authors:
Markus Beutel, Sevket Gökay, Peter von Grumbkow, Dirk Hillbrecht, Karl-Heinz Krempels, Christian Samsel, Christoph Terwelp, Heike Twele, Gerhard Wagner

Date: January 31, 2017 Document Version: 4.7d

Gefördert durch:

Bundesministerium für Wirtschaft und Technologie

aufgrund eines Beschlusses des Deutschen Bundestages

Contents

1 Overview

Aim of this interface specification is the information exchange of information systems for rentals vehicles with travel information systems. The reason for the information exchange is the requirement of creating intermodal travel chains combining rental services and public transport.

The specification in-hand contains:

- A Role Model,
- A Service hierachy of different qualities of information exchange,
- Interaction sequences to explain the order of messages between the partners based on the different service levels,
- The data model,
- Recommendations for the use of specific technologies for the data exchange and parsing,
- Tables of allowed values for enumerations.

2 Role Model

The role model describes the occuring roles in the information exchange.

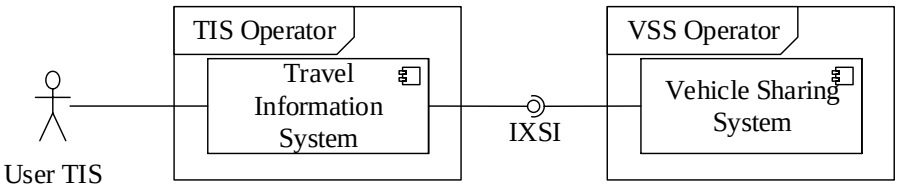


Figure 2.1: Overview Roles.

Travel Information System

The Travel Information System (TIS) is a information system, responsible for travel inquiries and covers the unification of travel options, the construction of travel chains, the calculation of a total price for a travel chain, the reservation of travel options for separated elements of the travel chain, the processing of data for representations in user interfaces and finally the presentation of information.

Use Cases

- User inquiries the TIS concerning mobility options, using search filters and preferences. Search filters and preferneces might be starting- and traget- location, departure- and arrival- time, transportation modes, amount of (mode-) switchings, price range, etc. results are provided via user-interfaces of the TIS in the form of travel chains.

Vehicle Sharing System

The Vehicle Sharing System (VRS) is a information system, responsible for managing and booking of sharing vehicles. Vehicles may vary in type or might be attached or unattached to sharing stations.

Use Cases

- A user books a vehicle via VRS to specific prices, times and stations and uses it.
- A user inquiries the availability of a vehicle via VRS.

User TIS

User TIS – represents a legal person, which is authorized to book and utilize a travel chain under use of selected modes of transportation.

Use Cases

- User sets up a travel inquiry towards the TIS.
- User books a travel via TIS.

User VRS

User VRS - represents a legal person, which is authorized to rent and use a vehicle.

Use Cases

- User sets up a travel inquiry towards the VRS.
- User books a travel via VRS.

Operator TIS

Operator TIS – provides the TIS as a service for transportation service providers.

Operator VRS

Operator VRS - provides the VRS as a service for vehicle rental companies.

3 Hierarchical Model

The hierarchical model describes different qualities of information coupling, based on service groups and serves as a recommendation concerning possible stages of implementation. To realize a coupling between VRS and TIS, at least service 1 (static data) of both interacting parties has to be supported. In fig. 3.1, dependencies between different services are depicted.

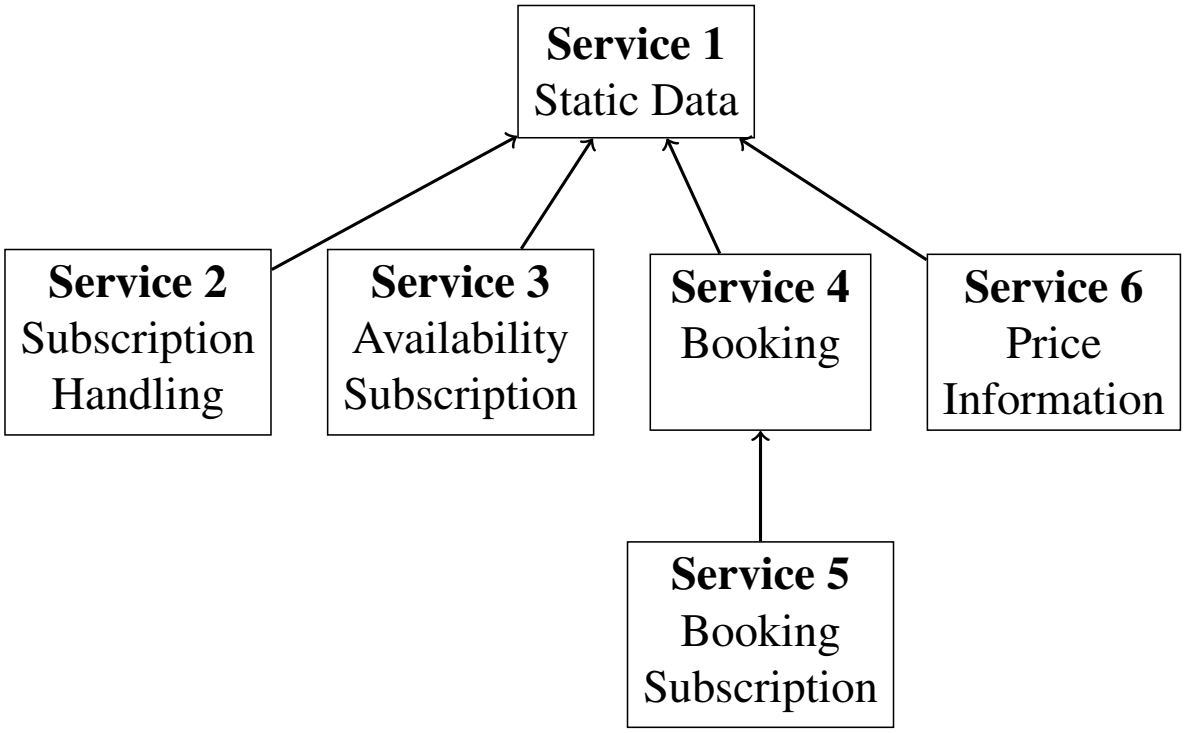


Figure 3.1: IXSI Service Groups.

3.1 Base Service A – Session Handling

Service A enables the authentication of end-customers towards the VRS.

Functions

- Open / Close Session

Dependencies

none

3.2 Base Service B – Subscription Handling)

Service B contains a function to check the status of a subscription connection (heartbeat)

Functions

- Heartbeat

Dependencies

none

3.3 Base Service C – tokens

Service C contains a function to create authentication tokens for users, which can be saved/ transferred instead of plaintext passwords.

Functions

- Creation of tokens

Dependencies

none

3.4 Service 1 – Static Data

Service 1 serves the information exchange across vehicle rental companies and static data of booking targets. These include provider-, position-, and vehicle-data.

For example, service 1 can be used, to display messages solely for existing locations of an VRS-provider in a TIS.

Functions

- Call of booking targets and provider information.

Dependencies

none

3.5 Service 2 – Availability Query

Service 2 serves for asynchronous calls of availability information.

The actual availability times of booking targets are called during the travel inquiry by the TIS at the VRS.

Functions

- Calls of availabilities of booking targets
- Calls of location capacities (Service 2a)

Dependencies

- Service 1

3.6 Service 3 – Availability Subscription

Services 2 serves the asynchronous exchange of availability information.

To accelerate the travel inquiry, the TIS can subscribe availability timescales of booking targets to avoid a query during the travel inquiry. After subscribing for an amount of booking targets, the VRS informs continuously about changes in availability timescales.

Functions

- Availability subscription
- Availability information (push)
- Subscription of location capacities (Service 3a)

Dependencies

- Service 1
- Base Service B

3.7 Service 4 – Booking

Service 4 serves booking, booking changes and canceling of vehicles via the TIS on behalf of the customer of a VRS.

The booking of a vehicle requires that a customer authenticates himself towards the VRS. Therefore, authentication information is forwarded from the TIS to the VRS. The secret (Password, key, etc.) of the customer is not allowed to be saved due to security reasons. The TIS receives an authentication-token in case of a successful authentication of a user. It can either be saved on the TIS or the user's device. Using the authentication-token, the TIS can proceed queries concerning bookings and canceling. To allow booking changes, a booking can be replaced by another booking via a booking change request. In case of an impossible booking, is has to be secured that the initial booking retains it's validity.

Functions

- Authentication of users towards the VHR
- Booking Query
- Query concerning booking changes/ canceling

Dependencies

- Service 1

3.8 Service 5 – Booking subscription

Service 5 serves the subscription of booking changes.

The TIS is able to subscribe conducted bookings at the VRS to inform users in case of changes, e.g., damaged vehicles.

Functions

- Booking subscriptions
- Booking alert (Push)

Dependencies

- Service 4
- Base Service B

3.9 Service 6 – Price information

Service 6 serves price information of rental services.

Through the transmission of starting and target location as well as departure and arrival time of the travel, the TIS is able to query price information at the VRS to inform the user. The VRS responds with an overall price and eventually individual price parts.

Functions

- Query of Prices

Dependencies

- Service 1

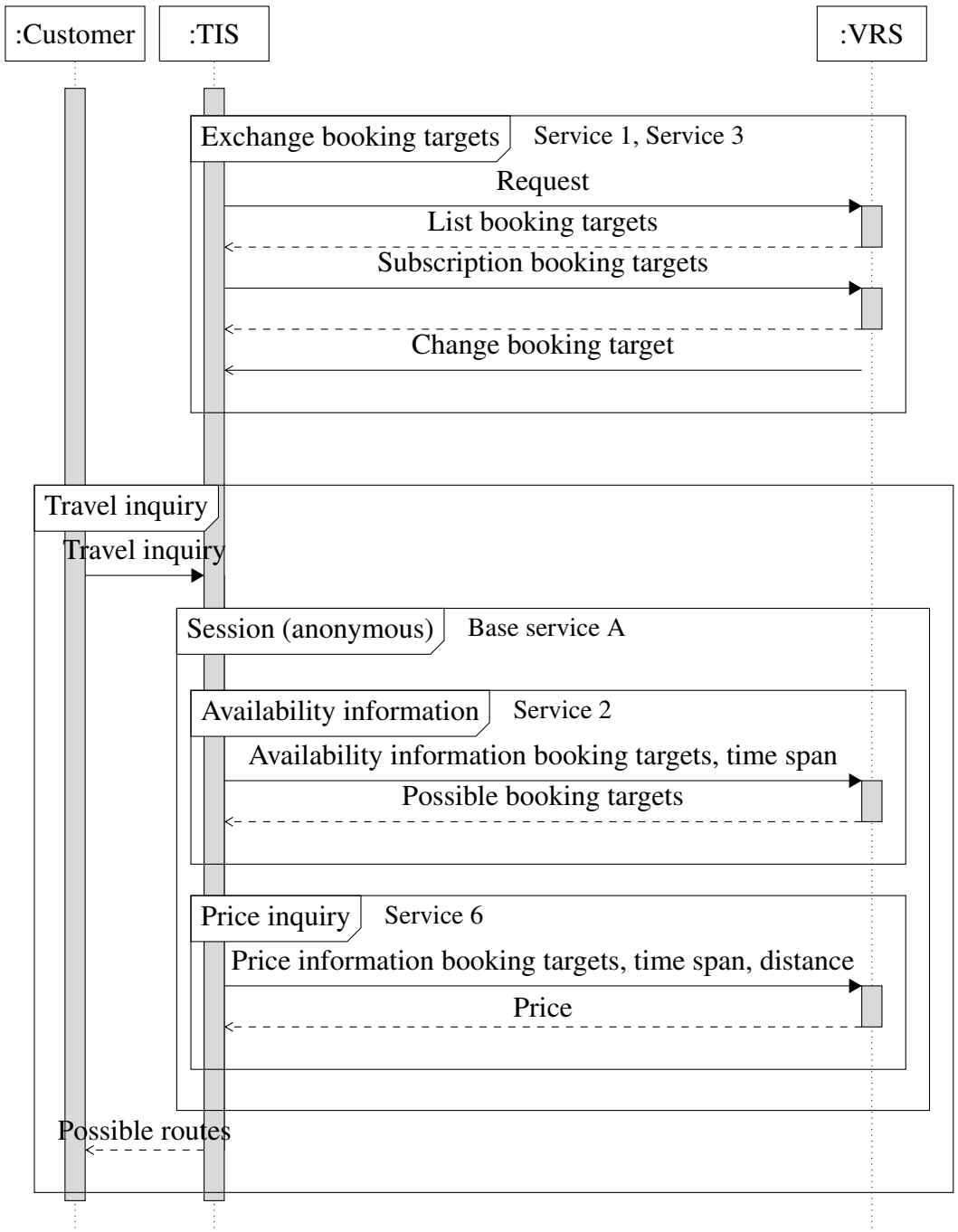
4 Interaction Sequences

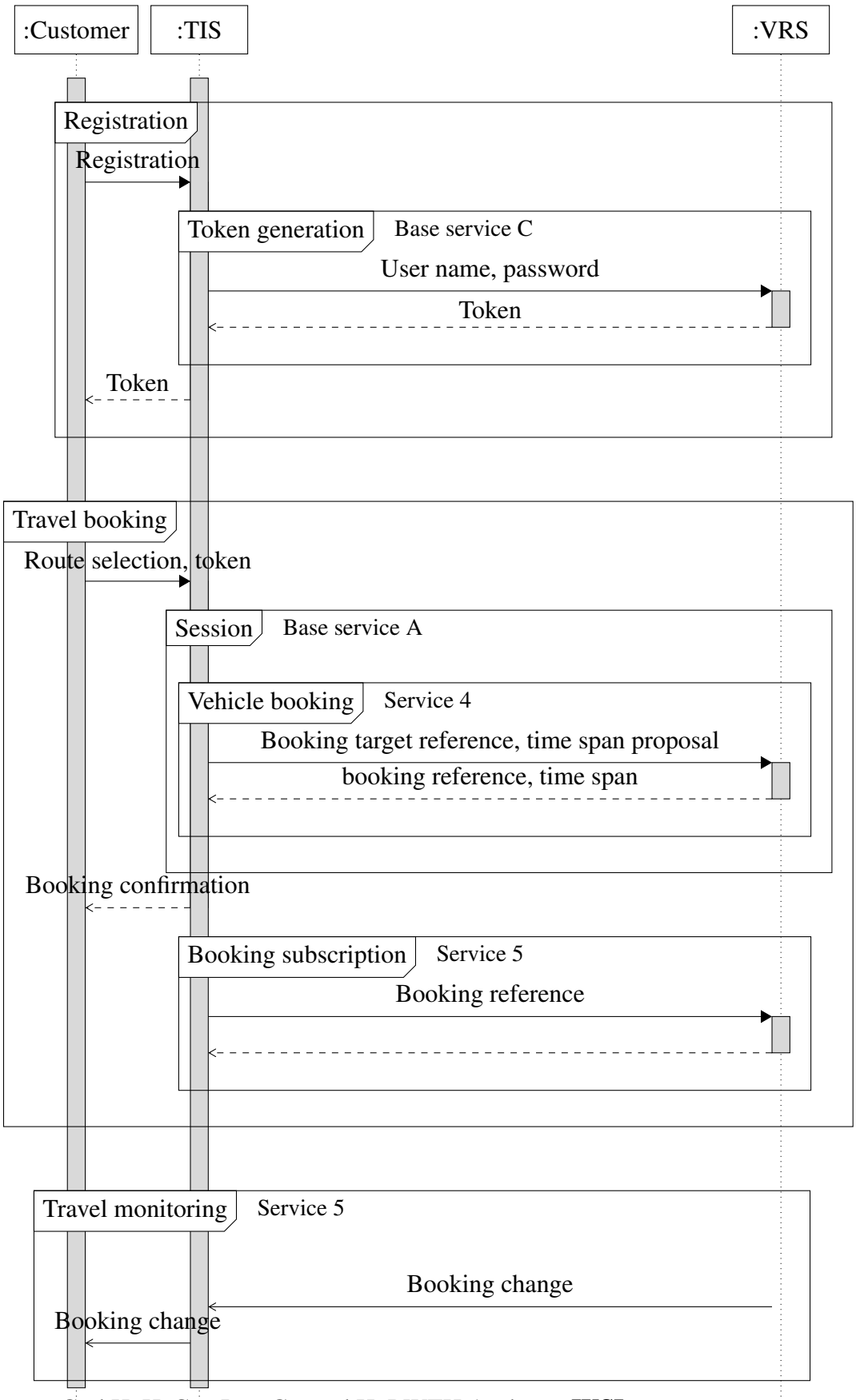
This section provides an overview about the interaction schemes, used by IXSI. For simplification, these interaction schemes are described in interaction sequence diagrams without using the technical terms of function calls. In principal, two types of interactions are used: The simple and well-known request/response- or. query- interaction scheme, whereby every call of the client (in this case TIS) is followed by exactly one response of the server (VRS). Furthermore, the subscription-scheme, whereby one object is subscribed by the client once and updated continuously by the server. Hereby, the communication channel is opened all the time.

4.1 Overview

The following sequence diagram gives an exemplary overview of information coupling, possibly enabled by IXSI. The used services are going to be described in detail in the following chapters. In this use case, a customer sets up a travel inquiry, books a respective travel and gets informed concerning relevant booking changes.

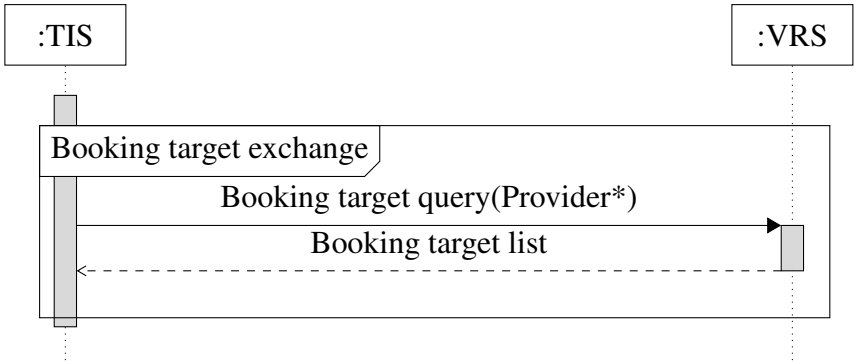
In the first block *Exchange booking targets*, the booking targets, provided by the VRS are exchanged with the TIS and relevant booking targets are subscribed (see. sections 4.2 and 4.4). This happens proactive, without involving the customer. In block *Travel Inquiry*, a customer sets up a travel inquiry with, e.g. his mobile device, on the TIS. Thereby, various rental vehicles are suitable for booking, whose availabilities are requested synchronously towards the VRS. For availabilities, the TIS additionally requests price information. As a result, the TIS provides a selection of possible routes/ connections to the customer. Because this is a communication triggered by the customer, a session which proceeds the queries is created implicitly. An anonymous session is used, because the customer did not register on his device beforehand. In block *Tarvel booking*, a customer has chosen a travel route and intends to book it. For this purpose, he registers on his mobile device, whereby a token is generated(Block *Registration*). With this token, a (not anonymous) session is created, which proceeds the booking operation. For that, the TIS forwards booking reference and a time proposal to the VRS. This sends a booking confirmation, which is forwarded to the customer. In addition, the TIS subscribes the respective booking on the VRS. In the last block *Travel monitoring*, the customers gets informed concerning booking changes, which are received by the TIS, coming from the VRS.


:Customer
:TIS
:VRS
Exchange booking targets
Service 1, Service 3
Request
List booking targets
Subscription booking targets
Change booking target
Travel inquiry
Travel inquiry
Session (anonymous)
Base service A
Availability information
Service 2
Availability information booking targets, time span
Possible booking targets
Price inquiry
Service 6
Price information booking targets, time span, distance
Price
Possible routes


:Customer
:TIS
:VRS
Registration
Registration
Token generation
Base service C
User name, password
Token
Token
Travel booking
Route selection, token
Session
Base service A
Vehicle booking
Service 4
Booking target reference, time span proposal
booking reference, time span
Booking confirmation
Booking subscription
Service 5
Booking reference
Travel monitoring
Service 5
Booking change
Booking change

4.2 Service 1 – Static Data

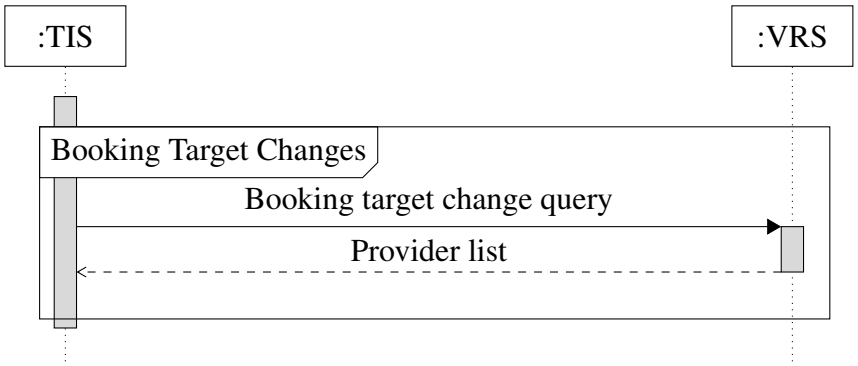
Booking Target Query



** optional*

Basis for information coupling is the exchange of booking targets. Booking targets are a logical representation of one or several vehicles with common characteristics, e.g., provided by the same provider, same vehicle type or same rental station. These characteristics are statically. To receive booking target information concerning a certain provider solely, it is possible to filter concerning the respective provider. The transmission is initiated by the TIS.

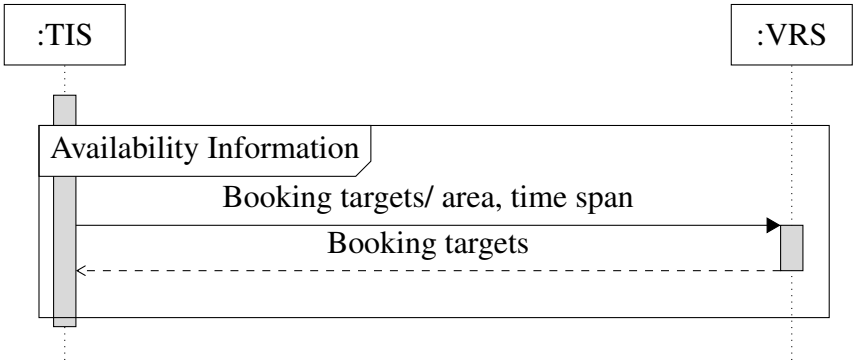
Booking Target Change Query



To avoid the transmission of all booking information per interval, with the help of the query `ChangedProviders` it is possible to request the provider, related to changes since a specific point in time, set by the parameter `timestamp`. A provider reference is returned, which in turn can be transferred as a parameter by the function call `BookingTargetsInfo`.

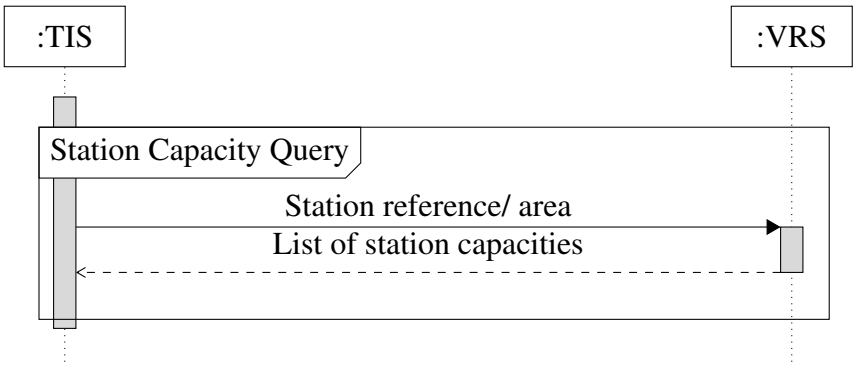
4.3 Service 2 – Availability Information

Availability Query



To request specific availabilities, the TIS sends a query, which either contains a list of booking targets or a geographic area in form of a proximity search or as a rectangle and a required time period. Without specification, the availabilities of all booking targets are returned. As a response, the VRS returns a list of booking targets and their availabilities.

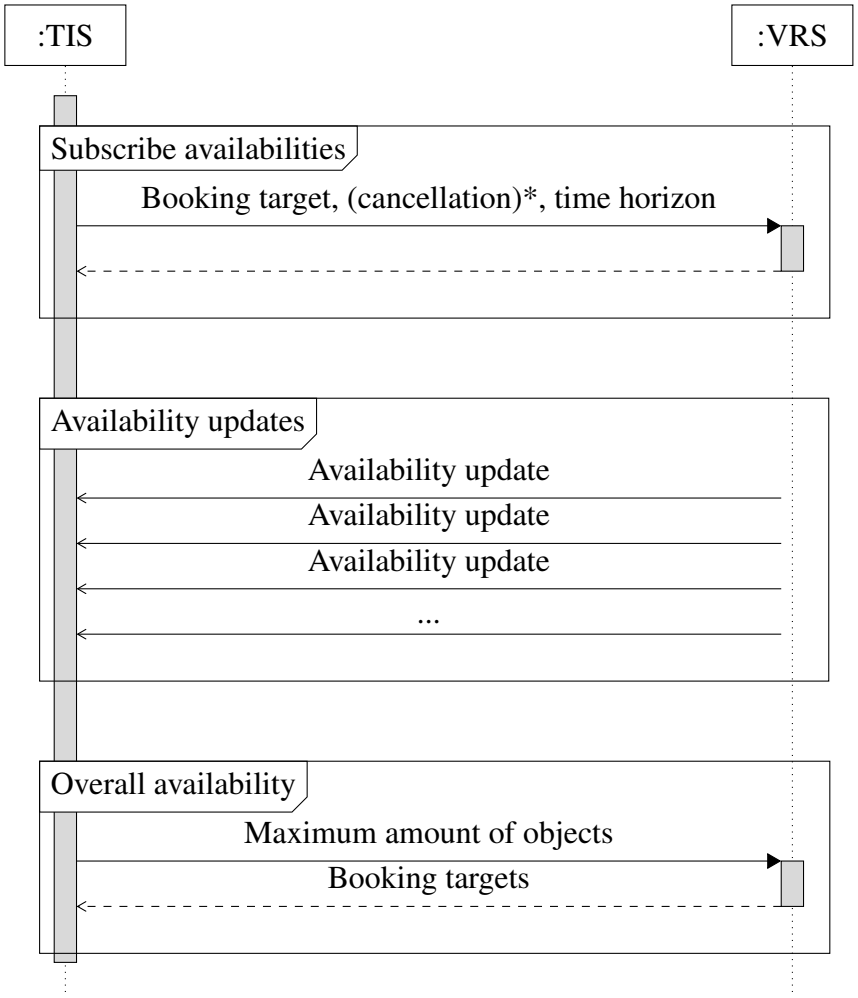
Current Station Capacity Query (Service 2a)



The TIS is able to request current capacities from rental stations, e.g., for map illustration. For this purpose, a list containing station IDs or an area has to be transferred. As a response, a list of locations and their current amount of available vehicles is returned.

4.4 Service 3 – Availability Subscription

Availability Subscription

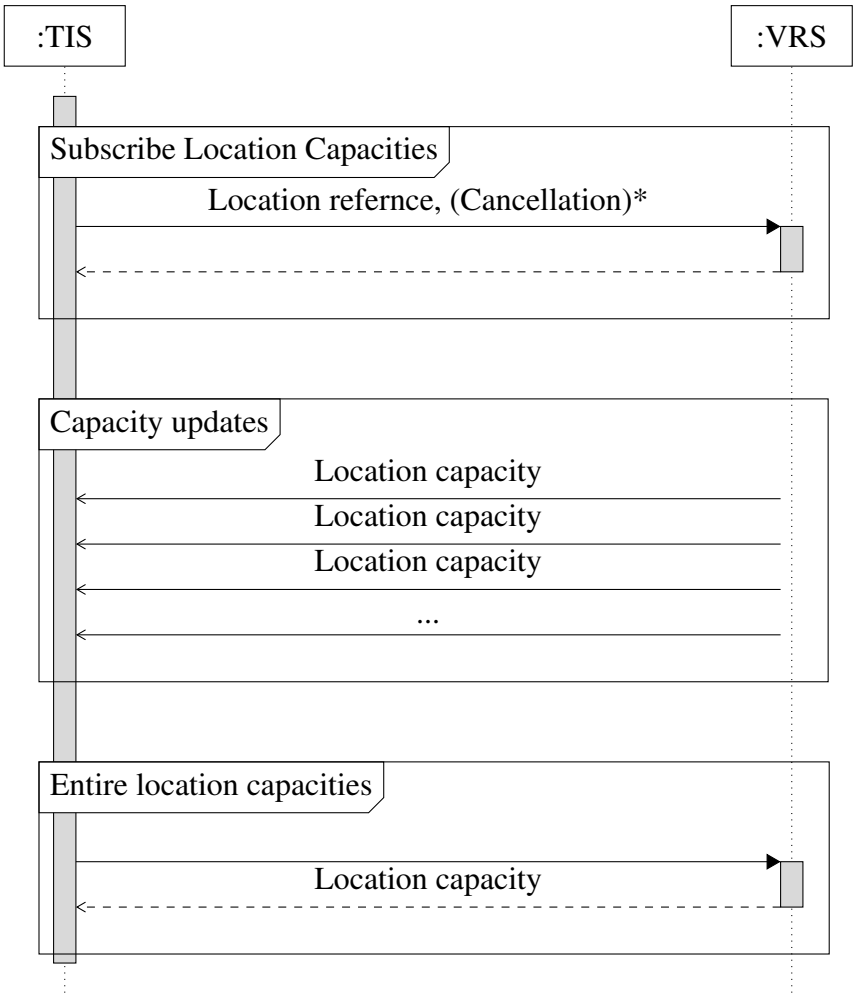


The TIS can subscribe to information concerning booking targets to get informed about availability changes immediately. In Principal, this serves to enable responses to travel inquiries without additional (synchronous) requests towards the VRS.

Through the initial query `AvailabilitySubscriptionRequest` a subscription is started. For this purpose, the TIS forwards the related booking reference. By setting the flag "Cancellation", a subscription can be canceled. In case of changes concerning availabilities, the VRS forwards asynchronously `AvailabilityPushMessage`. These a delivered via the same communication channel as upon the subscription was created beforehand. By cancellation of the communication channel, all subscriptions turn to be obsolete.

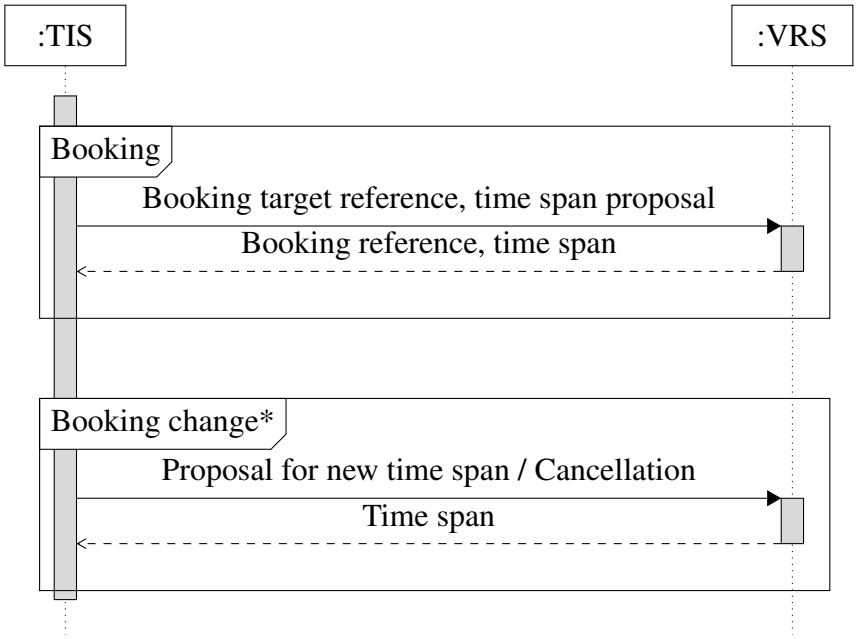
For initial synchronization of overall availabilities, the TIS is able to call the function `CompleteAvailabilityRequest`.

Location Capacity Subscription (Service 3a)



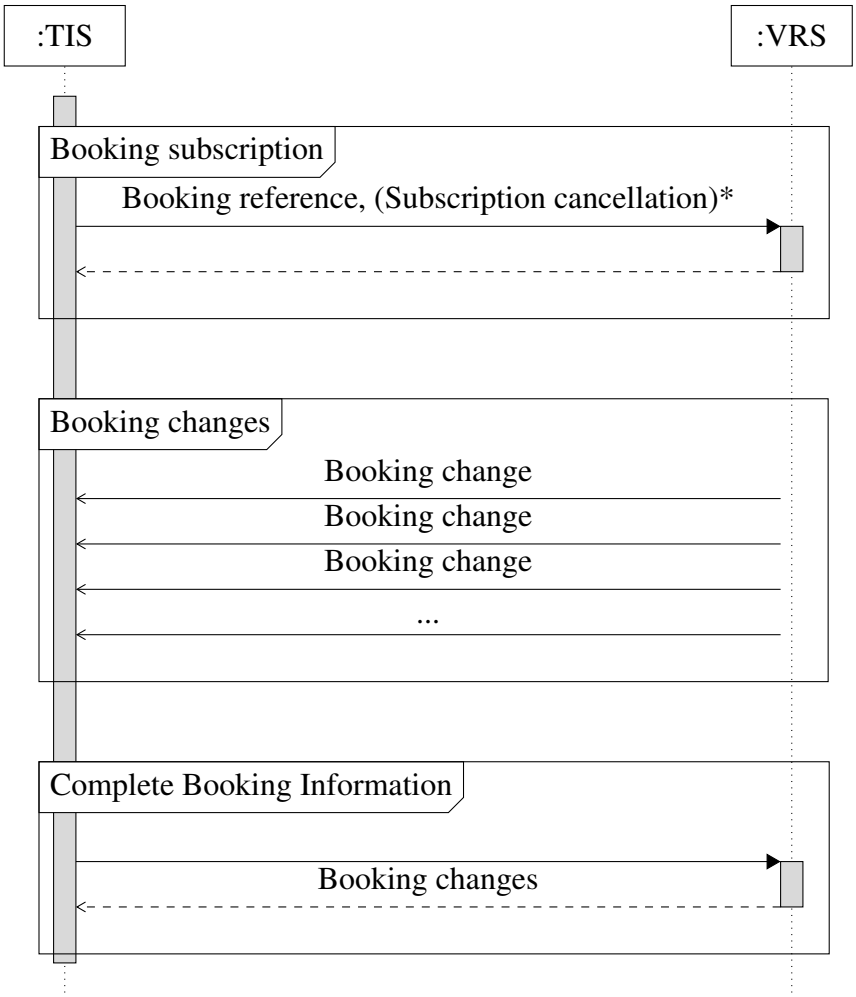
The TIS can subscribe capacity information of locations. The interaction procedure is analogous to section 4.4.

4.5 Service 4 – Booking/ Booking Change



To book a vehicle in customer order, it is necessary that the TIS authenticates the customers towards the VRS. For this purpose, there are three possibilities, further specified in section 5.8 In this example, a session is opened explicitly and closed after the transaction. After that, a booking can utilized by calling `Booking` with provision of the respective booking target ID and a time span proposal. “Vorschlag” because the VRS is able to change the time span to the used booking grid. As response, the used booking reference and the actual time span is returned. The booking reference can be used for monitoring of the booking (see section 4.6). For changes of the booking time span or for cancellation, `ChangeBooking` can be called. In case of changes of the booking target, a cancellation or a new booking becomes necessary.

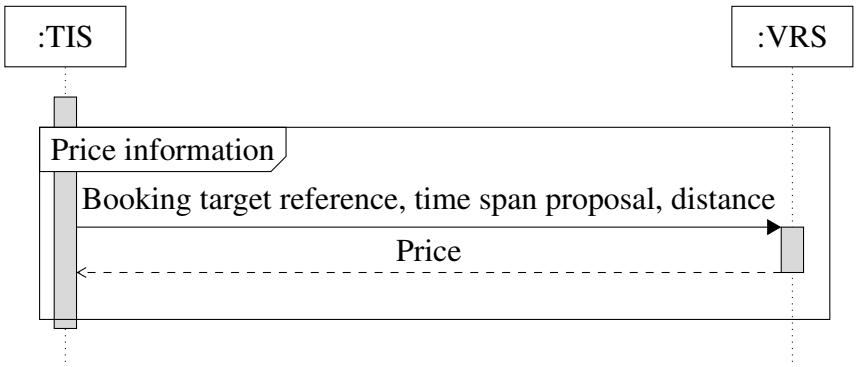
4.6 Service 5 – Booking Subscription



The TIS can subscribe to changes concerning bookings to provide this information to customers and probably propose alternatives. For example in case of an technical failure of a vehicle, the VRS can inform the TIS that a booking is not longer possible. In addition, it is possible to set a booking to “possible again”. The booking can be canceled ultimately only by the customer himself.

The interaction procedure is analogous to section 4.4.

4.7 Service 6 – Price Information



With the query `PriceInformationRequest`, the TIS can request price information towards the VRS on the basis of booking target ID, time span and traveled distance. In case of a customer authentication through `OpenSession` took place beforehand, the price request has to be replied-to accordingly to the customer contract.

5 Data Model

This chapter describes the underlying data model.

Legend of Symbols:
○ Choice (`xs:choice`)
□ Optional (`minOccurs=0`)
⧈ Multivalent (`maxOccurs=0`)

5.1 Base Data Types

Simple Base Types

Simple base types are aliases for existing data types to allow a semantic differentiation.

Name	Basetype	Comment
AreaIDType	xs:token	Type for area IDs.
AttributeClassType	xs:token	Enumeration for attribute classes.
AttributeIDType	xs:token	Enumeration for attribute codes.
BookeeIDType	xs:token	Type for bookee IDs.
BookingIDType	xs:token	Type for booking IDs.
ClassType	xs:token	Enumeration for car classes.
EngineType	xs:token	Enumeration for engines.
ErrorCodeType	xs:token	Enumeration for error codes.
EuroPriceType	xs:nonNegativeInteger	Type for storing prices in Euro cents.
GlobalIDType	xs:token	Type for globally unique IDs for identifiying clone entities.
PercentType	xs:nonNegativeInteger	Type for percent between 0 and 100.
PlaceGroupIDType	xs:token	Type for place group IDs.
PlaceIDType	xs:token	Type for place IDs.
ProviderIDType	xs:token	Type for provider IDs.
SessionIDType	xs:token	Type for session IDs.
SystemIDType	xs:token	Type for system IDs.
UserIDType	xs:token	Type for user IDs.

```
1  <xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
       ↪ name="AreaIDType">
2    <xs:restriction base="xs:token"/>
3  </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="PlaceIDType">
4    <xs:restriction base="xs:token"/>
5  </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="PlaceGroupIDType">
6    <xs:restriction base="xs:token"/>
7  </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="BookeeIDType">
8    <xs:restriction base="xs:token"/>
9  </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="ProviderIDType">
10   <xs:restriction base="xs:token"/>
11 </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="SessionIDType">
12   <xs:restriction base="xs:token"/>
13 </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="BookingIDType">
14   <xs:restriction base="xs:token"/>
15 </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="UserIDType">
16   <xs:restriction base="xs:token"/>
17 </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="SystemIDType">
18   <xs:restriction base="xs:token"/>
19 </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="GlobalIDType">
20   <xs:restriction base="xs:token"/>
21 </xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
       ↪ /2001/XMLSchema" name="EuroPriceType">
22   <xs:restriction base="xs:nonNegativeInteger"/>
```

```
</xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
    ↪ /2001/XMLSchema" name="PercentType">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
    ↪ /2001/XMLSchema" name="ErrorCodeType">
  <xs:restriction base="xs:token"/>
</xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
    ↪ /2001/XMLSchema" name="ClassType">
  <xs:restriction base="xs:token"/>
</xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
    ↪ /2001/XMLSchema" name="EngineType">
  <xs:restriction base="xs:token"/>
</xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
    ↪ /2001/XMLSchema" name="AttributeClassType">
  <xs:restriction base="xs:token"/>
</xs:simpleType><xs:simpleType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org
    ↪ /2001/XMLSchema" name="AttributeIDType">
  <xs:restriction base="xs:token"/>
</xs:simpleType>
```

XML source code 5.1: Baisdatentypen

Allowed values for enumerations *ClassType*, *EngineType*, *AttributeClassType* and *ErrorCodeType* are described in chapter 8.

Text

TextType: Type for a text.

	Element	Type	Comment
	Text	`xs:string`	
□	*Language*	`xs:language`	Language identifier according to BCP 47, which is based in ISO 639.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="TextType">
  <xs:sequence>
    <xs:element name="Text" type="xs:string"/>
    <xs:element name="Language" type="xs:language" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.2: TextType

Geographical Coordinates

CoordType: Type for geodecimal coordinates. The reference ellipsoid is always WGS84 (GPS).

	Element	Type	Comment
	Longitude	`simpleType`	
	Latitude	`simpleType`	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CoordType">
  <xs:sequence>
    <xs:element name="Longitude">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="-180.0"/>
          <xs:maxInclusive value="180.0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Latitude">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="-90.0"/>
          <xs:maxInclusive value="90.0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.3: CoordType

Address

AddressType: Type for storing an address.

	Element	Type	Comment
□	*Country*	`xs:string`	Name of the country.
□	*PostalCode*	`xs:string`	Complete postal code of the address.
	City	`xs:string`	Name of the city.
□	*StreetHouseNr*	`xs:string`	Street and house number in one string.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="AddressType">
  <xs:sequence>
    <xs:element name="Country" type="xs:string" minOccurs="0"/>
    <xs:element name="PostalCode" type="xs:string" minOccurs="0"/>
    <xs:element name="City" type="xs:string"/>
    <xs:element name="StreetHouseNr" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.4: AddressType

Location

GeoPositionType: Type for a geopraphical position.

	Element	Type	Comment
	Coord	`CoordType`	Geographical coordinates.
□	*Address*	`AddressType`	Optional address of the geographical position.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="GeoPositionType">
  <xs:sequence>
    <xs:element name="Coord" type="CoordType"/>
    <xs:element name="Address" type="AddressType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.5: GeoPositionType

Cycle

GeoCircleType: Type for a geographic circle.

Element	Type	Comment
Center	`CoordType`	Coordinates of the circle center.
Radius	`xs:nonNegativeInteger`	Radius in meters of the circle.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="GeoCircleType">
  <xs:sequence>
    <xs:element name="Center" type="CoordType"/>
    <xs:element name="Radius" type="xs:nonNegativeInteger"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.6: GeoCircleType

Rectangle

GeoRectangleType: Type for a geographic rectangle.

	Element	Type	Comment
	UpperLeft	`CoordType`	Upper left corner of the geographic rectangle.
	LowerRight	`CoordType`	Lower right corner of the geographic rectangle.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="GeoRectangleType">
  <xs:sequence>
    <xs:element name="UpperLeft" type="CoordType"/>
    <xs:element name="LowerRight" type="CoordType"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.7: GeoRectangleType

Area

GeoAreaType: Type for geographic areas.

	Element	Type	Comment
⊡	*PolyPoint*	`CoordType`	Geo positions of the closed border polygon of the area.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="GeoAreaType">
  <xs:sequence>
    <xs:element name="PolyPoint" type="CoordType" minOccurs="3" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.8: GeoAreaType

Area (Inclusion / Exclusion)

IncExcGeoAreaType: Type for geographic areas with exclusion possibility.
Basetype: *GeoAreaType*

	Element	Type	Comment
□	*Exclude*	`xs:boolean`	Only if set to true, this area definition defines an exclusion.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="IncExcGeoAreaType">
  <xs:complexContent>
    <xs:extension base="GeoAreaType">
      <xs:sequence>
        <xs:element name="Exclude" type="xs:boolean" default="false" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 5.9: IncExcGeoAreaType

Time Period

TimePeriodType: Type for a time period.

Element	Type	Comment
Begin	`xs:dateTime`	
End	`xs:dateTime`	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="TimePeriodType">
  <xs:sequence>
    <xs:element name="Begin" type="xs:dateTime"/>
    <xs:element name="End" type="xs:dateTime"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.10: TimePeriodType

Time Period (Proposal)

TimePeriodProposalType: Type for a time period proposal.
Basetype: *TimePeriodType*

	Element	Type	Comment
□	*MaxWait*	`xs:duration`	Maximal waiting time until the booking actually may begin in order to fulfil the booking time grid.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="TimePeriodProposalType">
  <xs:complexContent>
    <xs:extension base="TimePeriodType">
      <xs:sequence>
        <xs:element name="MaxWait" type="xs:duration" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 5.11: TimePeriodProposalType

Origin/Target

OriginDestType: Type for an origin or a destination location.

	Element	Type	Comment
○	*PlaceID*	`PlaceIDType`	Place ID. If given, the location refers to the place with this ID. Used for place based booking targets.
○	*GeoPosition*	`CoordType`	If given, the location refers to this geo position. Used for free floating booking targets.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="OriginDestType">
  <xs:choice>
    <xs:element name="PlaceID" type="PlaceIDType"/>
    <xs:element name="GeoPosition" type="CoordType"/>
  </xs:choice>
</xs:complexType>
```

XML source code 5.12: OriginDestType

Structured Information

InfoType: Type for structured information classification with associated text.

	Element	Type	Comment
□ ⊡	*Text*	`xs:string`	Text of the informaton to be shown to the user.
	WithText	`xs:boolean`	Defines, whether this information has a text meant for the user or not.
□	*Class*	`AttributeClassType`	Class of the information, taken from the list of allowed values.
□	*Mandatory*	`xs:boolean`	Defines whether the attribute has to be shown to the user in any case.
□	*Importance*	`PercentType`	Importance of the attribute in percent.
□	*URL*	`xs:anyURI`	URL to additional information on this general attribute. If available, the whole attribute text has to be used as the marked link.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="InfoType">
  <xs:sequence>
    <xs:element name="Text" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="WithText" type="xs:boolean"/>
    <xs:element name="Class" type="AttributeClassType" minOccurs="0"/>
    <xs:element name="Mandatory" type="xs:boolean" default="false" minOccurs="0"/>
    <xs:element name="Importance" type="PercentType" minOccurs="0"/>
    <xs:element name="URL" type="xs:anyURI" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.13: InfoType

Connection to Public Transportation

StopLinkType: Type for link from a car-sharing place to a stop.
group membership: *DurationGroup*

	Element	Type	Comment
	StopID	`xs:token`	
□	*Distance*	`xs:nonNegativeInteger`	Distance for walking this link in meters. In case of areas, this is the probable distance to walk to get a vehicle.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="StopLinkType">
2   <xs:sequence>
3     <xs:element name="StopID" type="xs:token"/>
4     <xs:group ref="DurationGroup"/>
5     <xs:element name="Distance" type="xs:nonNegativeInteger" minOccurs="0"/>
6   </xs:sequence>
7 </xs:complexType>
```

XML source code 5.14: StopLinkType

5.2 Base Groups

Position or Area

PlaceOrAreaGroup: Group for information which is used for places and free floating areas.

	Element	Type	Comment
⊡	*Name*	`TextType`	Printable name(s) of the place or area.
	ProviderID	`ProviderIDType`	Provider ID, to which the place or area belongs to.
□ ⊡	*Description*	`TextType`	
□ ⊡	*StopLink*	`StopLinkType`	Walking links to stops.
□ ⊡	*AttributeID*	`AttributeIDType`	Additional information on this place or area.

```
<xs:group xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
    ↪ PlaceOrAreaGroup">
  <xs:sequence>
    <xs:element name="Name" type="TextType" maxOccurs="unbounded"/>
    <xs:element name="ProviderID" type="ProviderIDType"/>
    <xs:element name="Description" type="TextType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="StopLink" type="StopLinkType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="AttributeID" type="AttributeIDType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
```

XML source code 5.15: PlaceOrAreaGroup

Duration

DurationGroup: Group for the probable duration and its variance for getting a booking target.

	Element	Type	Comment
	Duration	`xs:duration`	Probable duration for finding a booking target.
□	*Variance*	`xs:duration`	Variance of the probable duration for finding a booking target. The probable duration can vary by this value up or down.

```
<xs:group xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
    ↪ DurationGroup">
  <xs:sequence>
    <xs:element name="Duration" type="xs:duration"/>
    <xs:element name="Variance" type="xs:duration" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

XML source code 5.16: DurationGroup

5.3 Service 1 - Static Data

Booking Target ID

BookingTargetIDType: Type for booking target IDs.

Element	Type	Comment
BookeeID	`BookeeIDType`	
ProviderID	`ProviderIDType`	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="BookingTargetIDType">
2   <xs:sequence>
3     <xs:element name="BookeeID" type="BookeeIDType"/>
4     <xs:element name="ProviderID" type="ProviderIDType"/>
5   </xs:sequence>
6 </xs:complexType>
```

XML source code 5.17: BookingTargetIDType

Rental Station ID

ProviderPlaceIDType: Type for provider/place combinations.

Element	Type	Comment
ProviderID	`ProviderIDType`	
PlaceID	`PlaceIDType`	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="ProviderPlaceIDType">
2   <xs:sequence>
3     <xs:element name="ProviderID" type="ProviderIDType"/>
4     <xs:element name="PlaceID" type="PlaceIDType"/>
5   </xs:sequence>
6 </xs:complexType>
```

XML source code 5.18: ProviderPlaceIDType

Vehicle Attribute

AttributeType: Type for structured attribute classification with associated text.

	Element	Type	Comment
□ ⊡	*Text*	`TextType`	Text of the attribute to be shown to the user.
	WithText	`xs:boolean`	Defines, whether this attribute has a text meant for the user or not.
	ID	`AttributeIDType`	Internal code of the attribute for referencing.
□	*Class*	`AttributeClassType`	Class of the attribute, taken from the list of allowed values.
□	*Separate*	`xs:boolean`	If set to true, this attribute describes accessory equipment, which has to be booked separately.
□	*Mandatory*	`xs:boolean`	Defines whether the attribute has to be shown to the user in any case.
□	*Importance*	`PercentType`	Importance of the attribute in percent.
□	*URL*	`xs:anyURI`	URL to additional information on this general attribute. If available, the whole attribute text has to be used as the marked link.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="AttributeType">
2   <xs:sequence>
3     <xs:element name="Text" type="TextType" minOccurs="0" maxOccurs="unbounded"/>
4     <xs:element name="WithText" type="xs:boolean"/>
5     <xs:element name="ID" type="AttributeIDType"/>
6     <xs:element name="Class" type="AttributeClassType" minOccurs="0"/>
7     <xs:element name="Separate" type="xs:boolean" default="false" minOccurs="0"/>
8     <xs:element name="Mandatory" type="xs:boolean" default="false" minOccurs="0"/>
9     <xs:element name="Importance" type="PercentType" minOccurs="0"/>
10    <xs:element name="URL" type="xs:anyURI" minOccurs="0"/>
11  </xs:sequence>
12 </xs:complexType>
```

XML source code 5.19: AttributeType

Rental Station

PlaceType: Type for place information.
group membership: *PlaceOrAreaGroup*

	Element	Type	Comment
	ID	`PlaceIDType`	Place ID. For one physical place this ID has to remain constant for subsequent responses.
□	*GlobalID*	`GlobalIDType`	Globallly unique ID of this object. Used for identifying clones of this object, if they are delivered from different systems. Such ID should be formed using unique elements like DNS domain names.
	GeoPosition	`GeoPositionType`	Geo position of the place.
□	*Capacity*	`xs:nonNegativeInteger`	Capacity of the place for returning booking targets.
□	*OnPremisesTime*	`xs:duration`	Additional time needed for finding the place on the premises. Can be used for large parking areas or parking houses. Defaults to 0.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="PlaceType">
  <xs:sequence>
    <xs:element name="ID" type="PlaceIDType"/>
    <xs:element name="GlobalID" type="GlobalIDType" minOccurs="0"/>
    <xs:element name="GeoPosition" type="GeoPositionType"/>
    <xs:element name="Capacity" type="xs:nonNegativeInteger" minOccurs="0"/>
    <xs:group ref="PlaceOrAreaGroup"/>
    <xs:element name="OnPremisesTime" type="xs:duration" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.20: PlaceType

Area with Dense Indication

DensityAreaType: Type for geographic areas with probable time to get a vehicle.
group membership: *DurationGroup*

	Element	Type	Comment
	Area	`GeoAreaType`	Geographic area definition.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="DensityAreaType">
  <xs:sequence>
    <xs:element name="Area" type="GeoAreaType"/>
    <xs:group ref="DurationGroup"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.21: DensityAreaType

Freefloating Area

FloatingAreaType: Type for free floating areas.
group membership: *PlaceOrAreaGroup DurationGroup*

	Element	Type	Comment
	ID	`AreaIDType`	Area ID. For one physical free floating area this ID has to remain constant for subsequent responses.
⊡	*Area*	`IncExcGeoAreaType`	Geographic areas covered by the free floaters. A point belongs to the free floating area, if it is within at least one of the given inclusion areas, and not in one of the exclusion areas.
□ ⊡	*SubArea*	`DensityAreaType`	Sub areas of the free floating area. For each sub area a probable time (and its variance) for getting a free floating booking target is given. For stops covered by more than one sub area, the minimum duration is used.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="FloatingAreaType">
  <xs:sequence>
    <xs:element name="ID" type="AreaIDType"/>
    <xs:element name="Area" type="IncExcGeoAreaType" maxOccurs="unbounded"/>
    <xs:group ref="PlaceOrAreaGroup"/>
    <xs:group ref="DurationGroup"/>
    <xs:element name="SubArea" type="DensityAreaType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.22: FloatingAreaType

Group of Rental Stations

PlaceGroupType: Type for place groups.

	Element	Type	Comment
	ID	`PlaceGroupIDType`	Place group ID. For one physical place group this ID has to remain constant for subsequent responses.
⊡	*PlaceID*	`ProbabilityPlaceIDType`	
	Probability	`PercentType`	Average probability for getting a booking target at a specific place in this place group. It only applies for those places, for which no specific value is provided.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="PlaceGroupType">
2   <xs:sequence>
3     <xs:element name="ID" type="PlaceGroupIDType"/>
4     <xs:element name="PlaceID" type="ProbabilityPlaceIDType" maxOccurs="unbounded"/>
5     <xs:element name="Probability" type="PercentType"/>
6   </xs:sequence>
7 </xs:complexType>
```

XML source code 5.23: PlaceGroupType

Provider

ProviderType: Type for provider information.

	Element	Type	Comment
	ID	`ProviderIDType`	Provider ID. For one provider this ID has to remain constant for subsequent responses.
	Name	`xs:normalizedString`	Printable full name of the provider.
	CustomerChoice	`xs:boolean`	If set, this provider is presented to the user as a "home provider". This way this provider becomes a "customer provider".
□	*ShortName*	`xs:normalizedString`	Printable short name of the provider.
□ ⧈	*AttributeID*	`AttributeIDType`	Additional information on this provider (e.g. URLs, logo, inter-app-URLs).

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="ProviderType">
2   <xs:sequence>
3     <xs:element name="ID" type="ProviderIDType"/>
4     <xs:element name="Name" type="xs:normalizedString"/>
5     <xs:element name="CustomerChoice" type="xs:boolean"/>
6     <xs:element name="ShortName" type="xs:normalizedString" minOccurs="0"/>
7     <xs:element name="AttributeID" type="AttributeIDType" minOccurs="0" maxOccurs="unbounded"/>
8   </xs:sequence>
9 </xs:complexType>
```

XML source code 5.24: ProviderType

Booking Target

BookingTargetType: Type for booking target information.

	Element	Type	Comment
	ID	`BookingTargetIDType`	BookingTargetID. For one physical booking target this ID has to remain constant for subsequent responses.
□	*GlobalID*	`GlobalIDType`	Globallly unique ID of this object. Used for identifying clones of this object, if they are delivered from different systems. Such ID should be formed using unique elements like DNS domain names.
⧈	*Name*	`TextType`	Printable name of the bookee.
○	*PlaceID*	`PlaceIDType`	Place ID, where the bookee is placed. This has to be filled for station based booking targets.
○	*PlaceGroupID*	`PlaceGroupIDType`	PLace group ID of this bookee, if it does not belong to a fixed place.
○	*AreaID*	`AreaIDType`	Area ID of this bookee. This has to be filled for free floating booking targets.
	Class	`ClassType`	Class of the bookee.
□	*BookingHorizon*	`xs:duration`	Maximum duration in future, when this target can be booked. This is relative to the current time.
□	*BookingGrid*	`xs:nonNegativeInteger`	Grid for start and end time of bookings in minutes. Should be a divisor of 60. Examples: 1, 5, 10, 15, 30, 60.
□	*OpeningTime*	`xs:duration`	Time needed for actually opening the booking target. If not given, the client system will assume some reasonable time.
□	*Engine*	`EngineType`	
□	*CO2Factor*	`xs:nonNegativeInteger`	CO2 emmision in gram per kilometer.
□	*MaxDistance*	`xs:nonNegativeInteger`	Maximal distance in meters, this bookee can go without longer stops. If not given, no limit is assumed.
□ ⧈	*AttributeID*	`AttributeIDType`	Additional information on this bookee, e.g. air-conditioning, navigational system, etc.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingTargetType">
  <xs:sequence>
    <xs:element name="ID" type="BookingTargetIDType"/>
    <xs:element name="GlobalID" type="GlobalIDType" minOccurs="0"/>
    <xs:element name="Name" type="TextType" maxOccurs="unbounded"/>
    <xs:choice>
      <xs:element name="PlaceID" type="PlaceIDType"/>
      <xs:element name="PlaceGroupID" type="PlaceGroupIDType"/>
      <xs:element name="AreaID" type="AreaIDType"/>
    </xs:choice>
    <xs:element name="Class" type="ClassType"/>
    <xs:element name="BookingHorizon" type="xs:duration" minOccurs="0"/>
    <xs:element name="BookingGrid" type="xs:nonNegativeInteger" minOccurs="0"/>
    <xs:element name="OpeningTime" type="xs:duration" minOccurs="0"/>
    <xs:element name="Engine" type="EngineType" minOccurs="0"/>
    <xs:element name="CO2Factor" type="xs:nonNegativeInteger" minOccurs="0"/>
    <xs:element name="MaxDistance" type="xs:nonNegativeInteger" minOccurs="0"/>
    <xs:element name="AttributeID" type="AttributeIDType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.25: BookingTargetType

Booking

BookingType: Type for a booking.

	Element	Type	Comment
	ID	`BookingIDType`	
□	*TimePeriod*	`TimePeriodType`	Actual time period of the booking, eventually adapted to the booking time grid.
□ ⊡	*Info*	`InfoType`	Additional information on this booking.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingType">
  <xs:sequence>
    <xs:element name="ID" type="BookingIDType"/>
    <xs:element name="TimePeriod" type="TimePeriodType" minOccurs="0"/>
    <xs:element name="Info" type="InfoType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.26: BookingType

5.4 Service 2 - Availability Information

Booking Characteristics

BookingTargetPropertiesType: Type for needed properties of a booking target for synchronous availability.

	Element	Type	Comment
	ID	`BookingTargetIDType`	Booking target ID.
□	*Distance*	`xs:nonNegativeInteger`	Distance to drive in meters. The car-sharing system should return only those booking targets as available, which are able to drive that distance (according to state of charge).
□	*SeverityFactor*	`xs:nonNegativeInteger`	Severity factor of the proposed route in percentage. 100 represents a nearly flat route. The higher the value, the more energy is cosumed for driving the route. E.g. a value of 200 indecates, that twice as much energy will be consumed as would be for a nearly flat route of the same length.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="BookingTargetPropertiesType">
2   <xs:sequence>
3     <xs:element name="ID" type="BookingTargetIDType"/>
4     <xs:element name="Distance" type="xs:nonNegativeInteger" minOccurs="0"/>
5     <xs:element name="SeverityFactor" type="xs:nonNegativeInteger" minOccurs="0"/>
6   </xs:sequence>
7 </xs:complexType>
```

XML source code 5.27: BookingTargetPropertiesType

Booking Target Availability

BookingTargetAvailabilityType: Type for availability of booking targets.

	Element	Type	Comment
	ID	`BookingTargetIDType`	
□	*PlaceID*	`PlaceIDType`	For currently available booking targets, which are not bound to a specific place, the current place should be returned.
□	*GeoPosition*	`GeoPositionType`	For currently available booking targets, which are not place bounded, the current position should be returned.
□ ⊡	*Inavailability*	`TimePeriodType`	Periods of inavailability of the booking target.
□	*CurrentStateOfCharge*	`PercentType`	Current state of charge iof booking target in percent.
□	*CurrentDrivingRange*	`xs:nonNegativeInteger`	Current driving range of booking target in meters, given a nearly flat route. If not given, the vehice is considered to have an infinite driving range.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingTargetAvailabilityType">
  <xs:sequence>
    <xs:element name="ID" type="BookingTargetIDType"/>
    <xs:element name="PlaceID" type="PlaceIDType" minOccurs="0"/>
    <xs:element name="GeoPosition" type="GeoPositionType" minOccurs="0"/>
    <xs:element name="Inavailability" type="TimePeriodType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="CurrentStateOfCharge" type="PercentType" minOccurs="0"/>
    <xs:element name="CurrentDrivingRange" type="xs:nonNegativeInteger" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.28: BookingTargetAvailabilityType

5.5 Service 4 - Booking

Vehicle Availability

BookingTargetChangeAvailabilityType: Type for changes in availability of booking targets.

	Element	Type	Comment
	ID	`BookingTargetIDType`	Booking target ID.
□	*PlaceID*	`PlaceIDType`	For currently available booking targets, which are not bound to a specific place, the current place should be returned.
□	*GeoPosition*	`GeoPositionType`	For newly available vehicles, which are not place bounded, the position should be returned.
○	*Inavailability*	`TimePeriodType`	New inavailability period
○	*Availability*	`TimePeriodType`	New availability period.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingTargetChangeAvailabilityType">
  <xs:sequence>
    <xs:element name="ID" type="BookingTargetIDType"/>
    <xs:element name="PlaceID" type="PlaceIDType" minOccurs="0"/>
    <xs:element name="GeoPosition" type="GeoPositionType" minOccurs="0"/>
    <xs:choice>
      <xs:element name="Inavailability" type="TimePeriodType"/>
      <xs:element name="Availability" type="TimePeriodType"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.29: BookingTargetChangeAvailabilityType

Location Capacity

PlaceAvailabilityType: Type for the available capacity of a place.

	Element	Type	Comment
	ID	`ProviderPlaceIDType`	
	Availability	`xs:nonNegativeInteger`	Currently available capacity for returning booking targets. This is the number of booking targets which could be returned now, without exceeding the overall capacity of the place.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="PlaceAvailabilityType">
  <xs:sequence>
    <xs:element name="ID" type="ProviderPlaceIDType"/>
    <xs:element name="Availability" type="xs:nonNegativeInteger"/>
```

```
5   </xs:sequence>
6 </xs:complexType>
```

XML source code 5.30: PlaceAvailabilityType

5.6 Service 5 - Booking Subscription

Booking Availability

BookingChangeType: Type for changes of a booking.

	Element	Type	Comment
	BookingID	`BookingIDType`	
○	*Cancelled*	`xs:boolean`	Booking has been cancelled by user, eventually through different channels (e.g. native app, phone, etc). Afterwards the booking ID is not valid anymore.
○	*NewPeriod*	`TimePeriodType`	Booking has been given a new period by user. eventually through different channels (e.g. antive app, phone, etc.).
○	*Notification*	`xs:boolean`	The booking is still valid, but there is some infromation for the user concerning the booking.
○	*Impossible*	`xs:boolean`	The booking has been rendered impossible by the car sharing system (e.g. due to a car failure). However, the booking ID is still valid, as the booking might be re-possiblized later.
○	*Repossiblized*	`xs:boolean`	The booking has been rendered possible again (e.g. by a replacement car).
□ ⊡	*Reason*	`TextType`	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="BookingChangeType">
2   <xs:sequence>
3     <xs:element name="BookingID" type="BookingIDType"/>
4     <xs:choice>
5       <xs:element name="Cancelled" type="xs:boolean"/>
6       <xs:element name="NewPeriod" type="TimePeriodType"/>
7       <xs:element name="Notification" type="xs:boolean"/>
8       <xs:element name="Impossible" type="xs:boolean"/>
9       <xs:element name="Repossiblized" type="xs:boolean"/>
```

```
    </xs:choice>
    <xs:element name="Reason" type="TextType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.31: BookingChangeType

5.7 Service 6 - Price Information

Price Information (Details)

TariffDetailType: Type for a component of a traiff including the price.

	Element	Type	Comment
	Name	`xs:string`	Printable name of the tariff detail.
	Price	`EuroPriceType`	Price in Euro-Cent.
□	*Description*	`xs:string`	Printable description of the tariff detail.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="TariffDetailType">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Price" type="EuroPriceType"/>
    <xs:element name="Description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.32: TariffDetailType

Price Information

TariffType: Type for a tariff including the price.

	Element	Type	Comment
	Name	`xs:string`	Printable name of the tariff.
	Price	`EuroPriceType`	Price in Euro-Cent.
□	*Description*	`xs:string`	Printable description of the tariff.
□ ⧈	*Detail*	`TariffDetailType`	Details for the overall price. If provided, it should contain a set of component prices, that sums up to the overall price.
□ ⧈	*Info*	`InfoType`	Additional information for this tariff.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="TariffType">
2   <xs:sequence>
3     <xs:element name="Name" type="xs:string"/>
4     <xs:element name="Price" type="EuroPriceType"/>
5     <xs:element name="Description" type="xs:string" minOccurs="0"/>
6     <xs:element name="Detail" type="TariffDetailType" minOccurs="0" maxOccurs="unbounded"/>
7     <xs:element name="Info" type="InfoType" minOccurs="0" maxOccurs="unbounded"/>
8   </xs:sequence>
9 </xs:complexType>
```

XML source code 5.33: TariffType

5.8 Authentication

User Identification

UserInfoType: Type for a user identification.

	Element	Type	Comment
	ProviderID	`ProviderIDType`	ID of the provider of the user. Without the user ID is not unique.
	UserID	`UserIDType`	User ID.
○	*Password*	`xs:normalizedString`	Password of the user.
○	*Token*	`xs:token`	Authentication token for the user, which replaces the password for data security reasons.

```xml
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="UserInfoType">
  <xs:sequence>
    <xs:element name="ProviderID" type="ProviderIDType"/>
    <xs:element name="UserID" type="UserIDType"/>
    <xs:choice>
      <xs:element name="Password" type="xs:normalizedString"/>
      <xs:element name="Token" type="xs:token"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.34: UserInfoType

Authentication

AuthType: Type for an authentication within a request.

	Element	Type	Comment
○	*SessionID*	`SessionIDType`	Valid session ID. If a session is reused, fill this one.
⊡	*UserInfo*	`UserInfoType`	If no session is available, the user has to authenticate.
○	*Anonymous*	`xs:boolean`	If neither an existing session, nor a use information is available, an anonyous session can be opened.
□	*SessionTimeout*	`xs:duration`	If set, the client wishes to get a session with the given inactivity timeout.

```xml
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="AuthType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="SessionID" type="SessionIDType"/>
      <xs:sequence>
        <xs:element name="UserInfo" type="UserInfoType" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:element name="Anonymous" type="xs:boolean"/>
    </xs:choice>
    <xs:element name="SessionTimeout" type="xs:duration" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 5.35: AuthType

5.9 Error Handling

Errors

ErrorType: Type for errors.

	Element	Type	Comment
	Code	`ErrorCodeType`	Code of the error taken from the list of allowed error codes.
□	*NonFatal*	`xs:boolean`	If set, the corresponding operation was in general successful and operations can continue.
□	*SystemMessage*	`xs:string`	Text for system logging, explaining the error more detailed. Should be in English.
□	*UserMessage*	`xs:string`	Text for user information in requested language.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="ErrorType">
2   <xs:sequence>
3     <xs:element name="Code" type="ErrorCodeType"/>
4     <xs:element name="NonFatal" type="xs:boolean" default="false" minOccurs="0"/>
5     <xs:element name="SystemMessage" type="xs:string" minOccurs="0"/>
6     <xs:element name="UserMessage" type="xs:string" minOccurs="0"/>
7   </xs:sequence>
8 </xs:complexType>
```

XML source code 5.36: ErrorType

6 Technische Realisierung

6.1 Nachrichtenkodierung

Die Nachrichten zwischen den beiden System werden als XML übertragen. Eine präzise Typdefinition wird durch das zur Schnittstelle gehörige XML Schema vorgegeben.

Falls sich der Overhead, der durch die Einführung von XML, entsteht als problematisch herausstellen sollte, besteht die Möglichkeit das Efficient XML Interchange (EXI) Protokoll einzusetzen. Der Einsatz von EXI würde die Größe der Nachrichten erheblich verringern, ohne die Vorteile der Verwendung von XML zu verlieren.

6.2 Kommunikationskanal

As the interface allows asynchronous subscription besides the usual request/response scheme, usage of the WebSocket protocol instead of plain HTTP is recommended. WebSockets allow persistent connections between both systems and a bidirectional message exchange. Da die Schnittstelle neben dem Anfrage- und Antwort-Schema auch ein asynchrones Abonnementmodell vorsieht und eine geringe Antwortzeit wünschenswert ist, wird für den Kommunikationskanal das WebSocket-Protokoll empfohlen. WebSockets erlauben es, eine bestehende Verbindung der beiden Systeme herzustellen und über diese bidirektional Nachrichten auszutauschen. Das FVS stellt den Server und das RIS den Client (im HTTP Kontext) dar. Es können im Prinzip beliebig viele Kommunikationskanäle geöffnet werden. Aktualisierungen von abonnierten Objekten werden über die gleiche Verbindung geliefert, über die sie abonniert wurden. Bei Unterbrechung der Verbindung endet das Abonnement.

6.3 Authentifizierung

IXSI is a B2B interface and therefore does not include an internal authentication mechanism. Instead the communication partners are advised to use existing authentication mechanisms such as SSL certificates, a virtual private network (VPN) or HTTP authentication.

Endkunde

Da per IXSI auch nutzergesteuerte Requests von System zu System geschickt werden, ist es nötig, dass sich ein Nutzer gegenüber dem FVS authentifizieren kann. Dies geschieht normalerweise über das Tripel Anbieter Referenz/Nutzer Referenz/Passwort. Um das Passwort nicht im Klartext speichern zu müssen, kann alternativ zum Passwort ein Token verwendet werden, welches über die Funktion TokenGeneration (vgl. section 7.5) vom FVS generiert werden kann. So muss der Nutzer zwar initial sein Passwort einmalig eingeben, anschließend kann er sich aber über das daraus generierte, gespeicherte Token authentifizieren. Dieses kann z. B. auf dem Endgerät des Nutzers gespeichert werden.

Bei aufeinanderfolgenden Anfragen, die vom selben Nutzer ausgelöst werden, soll nicht in jedem Request eine Authentifizierung durchgeführt werden. Daher wird mit dem ersten Request eines Nutzers eine Session eröffnet (explizit durch OpenSession oder implizit). Für Folgerequests desselben Nutzers kann dann anstatt einer Authentifizierung die ID der eröffneten Session mitgeschickt werden. Diese Sessions besitzen nur eine beschränkte zeitliche Gültigkeit. Nach Ablauf dieser Gültigkeit muss der Nutzer neu identifiziert und eine neue Session eröffnet werden. Eine Session kann auch explizit durch CloseSession geschlossen werden.

6.4 Verbindungssicherheit

To ensure confidentiality of the transferred data, encryption is required. Therefore, the SSL/TLS-protocol is suitable. This should be used in case that the connection is not already secured through other measures (e.g., by using VPN).

7 Nachrichten

Die zwischen den Interaktionspartnern ausgetauschten Nachrichten basieren auf fünf Nachrichtentypen der Art Request / Response und Push. Alle Nachrichten vom RIS an das FVS stellen Anfragen dar und sind daher vom Typ Request. Der Typ Response wird für direkte Antworten des FVS auf Anfragen des RIS verwendet. Um eine Zuordnung der Response-Nachrichten zu den zugehörigen Request-Nachrichten zu erlauben, wird jede Request-Nachricht mit einer eindeutigen Transaction-ID markiert, die in der zugehörigen Response-Nachricht wieder mitgegeben werden muss. Eine Response-Nachricht kann anstelle ihres normalen Inhalts auch einen Fehler enthalten. Der letzte Nachrichtentyp Push findet in den Fällen Verwendung, in denen das RIS fortlaufende Aktualisierungen (Abonnement) vom FVS angefordert hat.

Für einige Nachrichten sind Beispiele zur illustration vorhanden, siehe z.B. listing 7.24.

7.1 Basisnachrichten

Basisklasse

IxsiMessageType: Type for all IXSI messages.

	Element	Type	Comment
○ ⧈	*Request*	QueryRequestType	Independent requests in query scheme.
○ ⧈	*Response*	QueryResponseType	Independent responses in query scheme.
○	*SubscriptionRequest*	SubscriptionRequestType	Request in subscription scheme. Each request leads to at least one response.
○	*SubscriptionResponse*	SubscriptionResponseType	Response in subscription scheme. Each response corresponds to exactly one request.
○	*SubscriptionMessage*	SubscriptionMessageType	Push message in subscription scheme.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="IxsiMessageType">
2   <xs:choice>
3     <xs:element name="Request" type="QueryRequestType" maxOccurs="unbounded"/>
4     <xs:element name="Response" type="QueryResponseType" maxOccurs="unbounded"/>
5     <xs:element name="SubscriptionRequest" type="SubscriptionRequestType"/>
6     <xs:element name="SubscriptionResponse" type="SubscriptionResponseType"/>
7     <xs:element name="SubscriptionMessage" type="SubscriptionMessageType"/>
8   </xs:choice>
9 </xs:complexType>
```

XML source code 7.1: IxsiMessageType

Basisantwort

AbstractBaseResponseType: Abstract type as a base for responses.

	Element	Type	Comment
□ ⧈	*Error*	ErrorType	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="AbstractBaseResponseType" abstract="true">
2   <xs:sequence>
3     <xs:element name="Error" type="ErrorType" minOccurs="0" maxOccurs="unbounded"/>
4   </xs:sequence>
5 </xs:complexType>
```

XML source code 7.2: AbstractBaseResponseType

Eine abstrakte Basisklasse für einen Request ist nicht vorhanden, da nicht erforderlich.

Transaktionsdaten

TransactionType: Type for an transaction ID within requests and responses.

Element	Type	Comment
TimeStamp	`xs:dateTime`	
MessageID	`xs:nonNegativeInteger`	Unique ID.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="TransactionType">
2   <xs:sequence>
3     <xs:element name="TimeStamp" type="xs:dateTime"/>
4     <xs:element name="MessageID" type="xs:nonNegativeInteger"/>
5   </xs:sequence>
6 </xs:complexType>
```

XML source code 7.3: TransactionType

Query Request

QueryRequestType: Request messages in query scheme.
group membership: *StaticDataRequestGroup UserTriggeredRequestGroup*

Element	Type	Comment
Transaction	`TransactionType`	Transaction identifier, set by the requestor.
SystemID	`SystemIDType`	ID of the requesting system.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="QueryRequestType">
2   <xs:sequence>
3     <xs:element name="Transaction" type="TransactionType"/>
4     <xs:element name="SystemID" type="SystemIDType"/>
5     <xs:choice>
6       <xs:group ref="StaticDataRequestGroup"/>
7       <xs:group ref="UserTriggeredRequestGroup"/>
8     </xs:choice>
9   </xs:sequence>
10 </xs:complexType>
```

XML source code 7.4: QueryRequestType

Query Response

QueryResponseType: Response messages in query scheme.
group membership: *StaticDataResponseGroup UserTriggeredResponseGroup*

Element	Type	Comment
Transaction	`TransactionType`	Transaction identifier, copied from the corresponding request.
CalcTime	`xs:duration`	Real time needed for calculating the contained results.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="QueryResponseType">
  <xs:sequence>
    <xs:element name="Transaction" type="TransactionType"/>
    <xs:element name="CalcTime" type="xs:duration"/>
    <xs:choice>
      <xs:group ref="StaticDataResponseGroup"/>
      <xs:group ref="UserTriggeredResponseGroup"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.5: QueryResponseType

Subscription Request

SubscriptionRequestType: Request messages in subscription scheme.
group membership: *SubscriptionAdministrationRequestGroup SubscriptionRequestGroup RequestMessageGroup*

Element	Type	Comment
Transaction	`TransactionType`	Transaction identifier, set by the requestor.
SystemID	`SystemIDType`	ID of the requesting system.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="SubscriptionRequestType">
  <xs:sequence>
    <xs:element name="Transaction" type="TransactionType"/>
    <xs:element name="SystemID" type="SystemIDType"/>
    <xs:choice>
      <xs:group ref="SubscriptionAdministrationRequestGroup"/>
      <xs:group ref="SubscriptionRequestGroup"/>
      <xs:group ref="RequestMessageGroup"/>
```

```
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.6: SubscriptionRequestType

Subscription Response

SubscriptionResponseType: Response messages in subscription scheme.
group membership: *SubscriptionAdministrationResponseGroup SubscriptionResponseGroup ResponseMessageGroup*

Element	Type	Comment
Transaction	`TransactionType`	Transaction identifier, copied from the corresponding request.
CalcTime	`xs:duration`	Real time needed for calculating the contained results.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="SubscriptionResponseType">
  <xs:sequence>
    <xs:element name="Transaction" type="TransactionType"/>
    <xs:element name="CalcTime" type="xs:duration"/>
    <xs:choice>
      <xs:group ref="SubscriptionAdministrationResponseGroup"/>
      <xs:group ref="SubscriptionResponseGroup"/>
      <xs:group ref="ResponseMessageGroup"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.7: SubscriptionResponseType

7.2 Basisgruppen

Benutzer-initiiert

UserTriggeredRequestGroup: Requests which are directly triggered through a user interaction.

	Element	Type	Comment
□	*Language*	`xs:language`	Requested language for the results. All texts returned in the respective responses should be in this language.
	Auth	`AuthType`	Authentication information.
○	*OpenSession*	`OpenSessionRequestType`	Open a session for a user.
○	*CloseSession*	`CloseSessionRequestType`	Explicitely close a session.
○	*TokenGeneration*	`TokenGenerationRequestType`	Generate a token for a user which replaces his password.
○	*Availability*	`AvailabilityRequestType`	Get the availability for the given booking targets.
○	*PlaceAvailability*	`PlaceAvailabilityRequestType`	Get the availabile capacity for the given places.
○	*PriceInformation*	`PriceInformationRequestType`	Get a price information.
○	*Booking*	`BookingRequestType`	Book a booking target.
○	*ChangeBooking*	`ChangeBookingRequestType`	Change a booking.

```
1 <xs:group xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
      ↪ UserTriggeredRequestGroup">
2   <xs:sequence>
3     <xs:element name="Language" type="xs:language" minOccurs="0"/>
4     <xs:element name="Auth" type="AuthType"/>
5     <xs:choice>
6       <xs:element name="OpenSession" type="OpenSessionRequestType"/>
7       <xs:element name="CloseSession" type="CloseSessionRequestType"/>
8       <xs:element name="TokenGeneration" type="TokenGenerationRequestType"/>
9       <xs:element name="Availability" type="AvailabilityRequestType"/>
10      <xs:element name="PlaceAvailability" type="PlaceAvailabilityRequestType"/>
11      <xs:element name="PriceInformation" type="PriceInformationRequestType"/>
12      <xs:element name="Booking" type="BookingRequestType"/>
13      <xs:element name="ChangeBooking" type="ChangeBookingRequestType"/>
14    </xs:choice>
15  </xs:sequence>
16 </xs:group>
```

XML source code 7.8: UserTriggeredRequestGroup

UserTriggeredResponseGroup: Responses, which are directly triggered through a user interaction.

	Element	Type	Comment
□	*SessionID*	`SessionIDType`	If returned, a session for the user was opened. This session ID can be reused for authentication in the next request.
□	*SessionTimeout*	`xs:duration`	Inactivity timeout of the session. If a period of inactivity on this session exceeds this value, the session timeouts.
○	*OpenSession*	`OpenSessionResponseType`	Open a session for the user.
○	*CloseSession*	`CloseSessionResponseType`	Explicitely close a session.
○	*TokenGeneration*	`TokenGenerationResponseType`	Generate a token for a user which replaces his password.
○	*Availability*	`AvailabilityResponseType`	Get the availability for the given booking targets.
○	*PlaceAvailability*	`PlaceAvailabilityResponseType`	Get the availabile capacity for the given places.
○	*PriceInformation*	`PriceInformationResponseType`	Get a price information.
○	*Booking*	`BookingResponseType`	Book a booking target.
○	*ChangeBooking*	`ChangeBookingResponseType`	Change a booking.

```
1 <xs:group xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
      ↪ UserTriggeredResponseGroup">
2   <xs:sequence>
3     <xs:element name="SessionID" type="SessionIDType" minOccurs="0"/>
4     <xs:element name="SessionTimeout" type="xs:duration" minOccurs="0"/>
5     <xs:choice>
6       <xs:element name="OpenSession" type="OpenSessionResponseType"/>
7       <xs:element name="CloseSession" type="CloseSessionResponseType"/>
8       <xs:element name="TokenGeneration" type="TokenGenerationResponseType"/>
9       <xs:element name="Availability" type="AvailabilityResponseType"/>
10      <xs:element name="PlaceAvailability" type="PlaceAvailabilityResponseType"/>
11      <xs:element name="PriceInformation" type="PriceInformationResponseType"/>
12      <xs:element name="Booking" type="BookingResponseType"/>
13      <xs:element name="ChangeBooking" type="ChangeBookingResponseType"/>
14    </xs:choice>
15  </xs:sequence>
16 </xs:group>
```

XML source code 7.9: UserTriggeredResponseGroup

Herkunft/Zielangabe

OriginDestGroup: Group for origin and destination location.

	Element	Type	Comment
□	*Origin*	`OriginDestType`	Location where the booking target is to be picked up.
□	*Dest*	`OriginDestType`	Location where the booking target is to be returned.

```
1 <xs:group xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema" name="
      ↪ OriginDestGroup">
2   <xs:sequence>
3     <xs:element name="Origin" type="OriginDestType" minOccurs="0"/>
4     <xs:element name="Dest" type="OriginDestType" minOccurs="0"/>
5   </xs:sequence>
6 </xs:group>
```

XML source code 7.10: OriginDestGroup

Gruppen rein zur Organisation

Anmerkung: Im XML Schema sind weitere Gruppen definiert, die jedoch nur der Organisation dienen und keinen Einfluss auf das Nachrichtenformat haben und deshalb hier nicht aufgelistet sind.

Name	Kommentar
`StaticDataRequestGroup`	
`StaticDataResponseGroup`	
`SubscriptionAdministrationRequestGroup`	
`SubscriptionAdministrationResponseGroup`	
`SubscriptionRequestGroup`	
`SubscriptionResponseGroup`	
`RequestMessageGroup`	
`ResponseMessageGroup`	
`PushMessageGroup`	

7.3 Dienst A - Sitzungen

Session öffnen

OpenSessionRequestType: Request for opening a session for a user. Can be used for Function 2, 4, and 6
group membership: *UserTriggeredRequestGroup*

	Element	Type	Comment
	(empty)		

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="OpenSessionRequestType"/>
```

XML source code 7.11: OpenSessionRequestType

OpenSessionResponseType: Response for for opening a session for a user.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
	(empty)		

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="OpenSessionResponseType">
2   <xs:complexContent>
3     <xs:extension base="AbstractBaseResponseType"/>
4   </xs:complexContent>
5 </xs:complexType>
```

XML source code 7.12: OpenSessionResponseType

Session schließen

CloseSessionRequestType: Request for closing an existing session.
group membership: *UserTriggeredRequestGroup*

Element	Type	Comment
(empty)		

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CloseSessionRequestType"/>
```

XML source code 7.13: CloseSessionRequestType

CloseSessionResponseType: Response for clsoing an existing session.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

Element	Type	Comment
(empty)		

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CloseSessionResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType"/>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.14: CloseSessionResponseType

7.4 Dienst B - Abonnements

Heartbeat

HeartBeatRequestType: Request for checking a connection.
group membership: *SubscriptionAdministrationRequestGroup*

Element	Type	Comment
(empty)		

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="HeartBeatRequestType"/>
```

XML source code 7.15: HeartBeatRequestType

HeartBeatResponseType: Response for checking a connection.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionAdministrationResponseGroup*

Element	Type	Comment
(empty)		

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="HeartBeatResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType"/>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.16: HeartBeatResponseType

7.5 Dienst C - Tokens

Tokengenerierung

TokenGenerationRequestType: Request for generating a token for a user.
group membership: *UserTriggeredRequestGroup*

Element	Type	Comment
(empty)		

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="TokenGenerationRequestType"/>
```

XML source code 7.17: TokenGenerationRequestType

TokenGenerationResponseType: Response for generating a token for a user.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
□	*Token*	`xs:token`	Token for the user. Can be used in future instead of the password for authentication.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="TokenGenerationResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="Token" type="xs:token" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.18: TokenGenerationResponseType

7.6 Dienst 1 - Statische Daten

Abfrage Buchungsziele

BookingTargetsInfoRequestType: Request for getting all infos for all booking targets.
group membership: *StaticDataRequestGroup*

	Element	Type	Comment
□ ⧈	*ProviderFilter*	`ProviderIDType`	If set, the filter contains those providers, data is requested for. If not set, data for all providers is requested.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingTargetsInfoRequestType">
  <xs:sequence>
    <xs:element name="ProviderFilter" type="ProviderIDType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.19: BookingTargetsInfoRequestType

BookingTargetsInfoResponseType: Response for getting all infos for all booking targets.
Basetype: *AbstractBaseResponseType*
group membership: *StaticDataResponseGroup*

	Element	Type	Comment
□	*Timestamp*	`xs:dateTime`	Timestamp of this data delivery. The delivering system should be able to identify changes compared to this delivery by this timestamp.
□ ⊡	*Bookee*	`BookingTargetType`	
□ ⊡	*Place*	`PlaceType`	
□ ⊡	*PlaceGroup*	`PlaceGroupType`	
□ ⊡	*FreeFloatingArea*	`FloatingAreaType`	
□ ⊡	*Provider*	`ProviderType`	
□ ⊡	*Attributes*	`AttributeType`	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingTargetsInfoResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="Timestamp" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="Bookee" type="BookingTargetType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Place" type="PlaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="PlaceGroup" type="PlaceGroupType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="FreeFloatingArea" type="FloatingAreaType" minOccurs="0" maxOccurs="unbounded
            ↪ "/>
        <xs:element name="Provider" type="ProviderType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Attributes" type="AttributeType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.20: BookingTargetsInfoResponseType

Abfrage Änderungen Buchungsziele

ChangedProvidersRequestType: Request for getting those providers with changed static data.
group membership: *StaticDataRequestGroup*

Element	Type	Comment
Timestamp	`xs:dateTime`	Timestamp of the static data delivery, the changes are requested for.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="ChangedProvidersRequestType">
  <xs:sequence>
    <xs:element name="Timestamp" type="xs:dateTime"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.21: ChangedProvidersRequestType

ChangedProvidersResponseType: Repsonse for getting those providers with changed static data.
Basetype: *AbstractBaseResponseType*
group membership: *StaticDataResponseGroup*

	Element	Type	Comment
□ ⊡	*Provider*	`ProviderIDType`	List of providers, which have changes in their static data.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="ChangedProvidersResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="Provider" type="ProviderIDType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.22: ChangedProvidersResponseType

7.7 Dienst 2 - Verfügbarkeitsauskunft

Abfrage Verfügbarkeiten

AvailabilityRequestType: Synchronous request for availabilities of booking targets.
group membership: *UserTriggeredRequestGroup*

	Element	Type	Comment
○ ⊡	*BookingTarget*	BookingTargetPropertiesType	
○	*Circle*	GeoCircleType	
○	*GeoRectangle*	GeoRectangleType	
□	*TimePeriod*	TimePeriodType	Interesting time period, for which availabilities of the given booking targets shall be returned. If no given, currently available booking targets should be returned.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="AvailabilityRequestType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="BookingTarget" type="BookingTargetPropertiesType" maxOccurs="unbounded"/>
      <xs:element name="Circle" type="GeoCircleType"/>
      <xs:element name="GeoRectangle" type="GeoRectangleType"/>
    </xs:choice>
    <xs:element name="TimePeriod" type="TimePeriodType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.23: AvailabilityRequestType

```
<Ixsi>
    <Request>
        <Transaction>
            <TimeStamp>2014-11-03T11:19:01.976+01:00</TimeStamp>
            <MessageID>100</MessageID>
        </Transaction>
        <Auth>
            <UserInfo>
                <ProviderID>2</ProviderID>
                <UserID>40</UserID>
                <Password>x</Password>
            </UserInfo>
        </Auth>
        <Availability>
            <BookingTarget>
                <ID>
                    <BookeeID>10</BookeeID>
                    <ProviderID>2</ProviderID>
                </ID>
            </BookingTarget>
            <TimePeriod>
                <Begin>2014-11-04T15:01:00.000+01:00</Begin>
                <End>2014-11-09T17:14:00.000+01:00</End>
            </TimePeriod>
        </Availability>
    </Request>
</Ixsi>
```

XML source code 7.24: AvailabilityRequest Beispiel

AvailabilityResponseType: Synchronous response for availabilites of booking targets.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
□ ⊡	*BookingTarget*	`BookingTargetAvailabilityType`	List of booking targets and their availabilities. Not listed booking targets are considered to be either unknown to the car sharing system or not available at all in the requested time period.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="AvailabilityResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="BookingTarget" type="BookingTargetAvailabilityType" minOccurs="0" maxOccurs=
            ↪ "unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.25: AvailabilityResponseType

```
<Ixsi>
    <Response>
        <Transaction>
            <TimeStamp>2014-11-03T11:19:01.976+01:00</TimeStamp>
            <MessageID>100</MessageID>
        </Transaction>
        <CalcTime>PT0.090S</CalcTime>
        <Availability>
            <BookingTarget>
                <ID>
                    <BookeeID>10</BookeeID>
                    <ProviderID>2</ProviderID>
                </ID>
                <Inavailability>
                    <Begin>2014-11-09T17:00:00.000+01:00</Begin>
                    <End>2014-11-09T20:00:00.000+01:00</End>
                </Inavailability>
                <CurrentStateOfCharge>57</CurrentStateOfCharge>
            </BookingTarget>
        </Availability>
    </Response>
</Ixsi>
```

XML source code 7.26: AvailabilityResponse Beispiel

Abfrage Standortkapazitäten (Dienst 2a)

PlaceAvailabilityRequestType: Synchronous request for the available capacity of places.
group membership: *UserTriggeredRequestGroup*

	Element	Type	Comment
○ ⊡	*PlaceID*	ProviderPlaceIDType	
○	*Circle*	GeoCircleType	
○	*GeoRectangle*	GeoRectangleType	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="PlaceAvailabilityRequestType">
2   <xs:sequence>
3     <xs:choice>
4       <xs:element name="PlaceID" type="ProviderPlaceIDType" maxOccurs="unbounded"/>
5       <xs:element name="Circle" type="GeoCircleType"/>
6       <xs:element name="GeoRectangle" type="GeoRectangleType"/>
7     </xs:choice>
8   </xs:sequence>
9 </xs:complexType>
```

XML source code 7.27: PlaceAvailabilityRequestType

PlaceAvailabilityResponseType: Synchronous response for the available capacity of places..
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
□ ⊡	*Place*	PlaceAvailabilityType	List of places and their available capacity. Not listed places are considered to be either unknown to the car sharing system or not available at all.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="PlaceAvailabilityResponseType">
2   <xs:complexContent>
3     <xs:extension base="AbstractBaseResponseType">
4       <xs:sequence>
5         <xs:element name="Place" type="PlaceAvailabilityType" minOccurs="0" maxOccurs="unbounded"/>
6       </xs:sequence>
7     </xs:extension>
8   </xs:complexContent>
9 </xs:complexType>
```

XML source code 7.28: PlaceAvailabilityResponseType

7.8 Dienst 3 - Verfügbarkeitsabonnement

Verfügbarkeitsabonnement

AvailabilitySubscriptionRequestType: Request for subscribing to availabilities of given booking targets.
group membership: *SubscriptionRequestGroup*

	Element	Type	Comment
⊡	*BookingTargetID*	`BookingTargetIDType`	Booking tragets to subscribe/unsubscribe.
□	*Unsubscription*	`xs:boolean`	If set, unsubcribe from the given booking targets. Otherwise subscribe to them.
□	*EventHorizon*	`xs:duration`	If set, the event horizon defines the time in future, for wich the subscription is valid. This value is used for all subscribed booking targets. This duration is relative to the current time.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="AvailabilitySubscriptionRequestType">
2   <xs:sequence>
3     <xs:element name="BookingTargetID" type="BookingTargetIDType" maxOccurs="unbounded"/>
4     <xs:element name="Unsubscription" type="xs:boolean" default="false" minOccurs="0"/>
5     <xs:element name="EventHorizon" type="xs:duration" minOccurs="0"/>
6   </xs:sequence>
7 </xs:complexType>
```

XML source code 7.29: AvailabilitySubscriptionRequestType

```
1  <Ixsi>
2      <SubscriptionRequest>
3          <Transaction>
4              <TimeStamp>2014-11-03T11:23:47.309+01:00</TimeStamp>
5              <MessageID>100</MessageID>
6          </Transaction>
7          <AvailabilitySubscription>
8              <BookingTargetID>
9                  <BookeeID>7</BookeeID>
10                 <ProviderID>2</ProviderID>
11             </BookingTargetID>
12         </AvailabilitySubscription>
13     </SubscriptionRequest>
14 </Ixsi>
```

XML source code 7.30: AvailabilitySubscriptionRequest Beispiel

AvailabilitySubscriptionResponseType: Response for subscribing to availabilities of given booking targets.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionResponseGroup*

	Element	Type	Comment
	(empty)		

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="AvailabilitySubscriptionResponseType">
2   <xs:complexContent>
3     <xs:extension base="AbstractBaseResponseType"/>
4   </xs:complexContent>
5 </xs:complexType>
```

XML source code 7.31: AvailabilitySubscriptionResponseType

AvailabilitySubscriptionStatusRequestType: Request for getting all subsribed booking targets.
group membership: *SubscriptionRequestGroup*

	Element	Type	Comment
	(empty)		

AvailabilitySubscriptionStatusResponseType: Response for getting all subscribed booking targets.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionResponseGroup*

	Element	Type	Comment
□ ⊡	*BookingTargetID*	`BookingTargetIDType`	Subscribed booking targets.

AvailabilityPushMessageType: Push message containing changes in availability of subscribed booking targets.
group membership: *PushMessageGroup*

	Element	Type	Comment
⊡	*AvailabilityChange*	`BookingTargetChangeAvailabilityType`	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="AvailabilityPushMessageType">
  <xs:sequence>
    <xs:element name="AvailabilityChange" type="BookingTargetChangeAvailabilityType" maxOccurs="
        ↪ unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.32: AvailabilityPushMessageType

```
<Ixsi>
    <SubscriptionMessage>
        <AvailabilityPush>
            <AvailabilityChange>
                <ID>
                    <BookeeID>7</BookeeID>
                    <ProviderID>2</ProviderID>
                </ID>
                <Inavailability>
                    <Begin>2014-11-04T15:00:00.000+01:00</Begin>
                    <End>2014-11-04T17:30:00.000+01:00</End>
                </Inavailability>
            </AvailabilityChange>
        </AvailabilityPush>
    </SubscriptionMessage>
</Ixsi>
```

XML source code 7.33: AvailabilityPush Beispiel

Vollständige Verfügbarkeitsabonnementinformation

CompleteAvailabilityRequestType: Request for getting the complete current status of subscribed availabilities.
group membership: *RequestMessageGroup*

	Element	Type	Comment
□	*MaxTargets*	`xs:integer`	Defines the maximum number of targets to be returned in one message.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CompleteAvailabilityRequestType">
  <xs:sequence>
    <xs:element name="MaxTargets" type="xs:integer" minOccurs="0"/>
  </xs:sequence>
```

```
</xs:complexType>
```

XML source code 7.34: CompleteAvailabilityRequestType

```
<Ixsi>
    <SubscriptionRequest>
        <Transaction>
            <TimeStamp>2014-11-03T11:23:30.059+01:00</TimeStamp>
            <MessageID>100</MessageID>
        </Transaction>
        <CompleteAvailability>
            <MaxTargets>5</MaxTargets>
        </CompleteAvailability>
    </SubscriptionRequest>
</Ixsi>
```

XML source code 7.35: CompleteAvailabilityRequest Beispiel

CompleteAvailabilityResponseType: Response for getting the complete current status of subscribed abailabilities. Several such responses may belong to one single request.
Basetype: *AbstractBaseResponseType*
group membership: *ResponseMessageGroup*

	Element	Type	Comment
	MessageBlockID	`xs:token`	ID of the block, this message belongs to. All messages of one block have the same ID.
□	*Last*	`xs:boolean`	If set, this message is the last one in the corresponding block. Otherwise more messages of the same block will follow.
□ ⊡	*BookingTarget*	`BookingTargetAvailabilityType`	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CompleteAvailabilityResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="MessageBlockID" type="xs:token"/>
        <xs:element name="Last" type="xs:boolean" default="false" minOccurs="0"/>
        <xs:element name="BookingTarget" type="BookingTargetAvailabilityType" minOccurs="0" maxOccurs=
            ↪ "unbounded"/>
      </xs:sequence>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.36: CompleteAvailabilityResponseType

```
<Ixsi>
    <SubscriptionResponse>
        <Transaction>
            <TimeStamp>2014-11-03T11:23:30.059+01:00</TimeStamp>
            <MessageID>100</MessageID>
        </Transaction>
        <CalcTime>PT0.000S</CalcTime>
        <CompleteAvailability>
            <MessageBlockID>100</MessageBlockID>
            <Last>true</Last>
            <BookingTarget>
                <ID>
                    <BookeeID>14</BookeeID>
                    <ProviderID>2</ProviderID>
                </ID>
                <Inavailability>
                    <Begin>2014-11-04T15:00:00.000+01:00</Begin>
                    <End>2014-11-04T18:00:00.000+01:00</End>
                </Inavailability>
            </BookingTarget>
            <BookingTarget>
                <ID>
                    <BookeeID>15</BookeeID>
                    <ProviderID>2</ProviderID>
                </ID>
            </BookingTarget>
            <BookingTarget>
                <ID>
                    <BookeeID>16</BookeeID>
                    <ProviderID>2</ProviderID>
                </ID>
            </BookingTarget>
        </CompleteAvailability>
    </SubscriptionResponse>
</Ixsi>
```

XML source code 7.37: CompleteAvailabilityResponse Beispiel

Standortkapazitätsabonnement (Dienst 3a)

PlaceAvailabilitySubscriptionRequestType: Request for subscribing to available capacity of places.
group membership: *SubscriptionRequestGroup*

	Element	Type	Comment
⊡	*PlaceID*	`ProviderPlaceIDType`	Places to subscribe/unsubscribe.
□	*Unsubscription*	`xs:boolean`	If set, unsubcribe from the given booking targets. Otherwise subscribe to them.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="PlaceAvailabilitySubscriptionRequestType">
2   <xs:sequence>
3     <xs:element name="PlaceID" type="ProviderPlaceIDType" maxOccurs="unbounded"/>
4     <xs:element name="Unsubscription" type="xs:boolean" default="false" minOccurs="0"/>
5   </xs:sequence>
6 </xs:complexType>
```

XML source code 7.38: PlaceAvailabilitySubscriptionRequestType

PlaceAvailabilitySubscriptionResponseType: Response for subscribing to available capacity of places.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionResponseGroup*

Element	Type	Comment
(empty)		

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="PlaceAvailabilitySubscriptionResponseType">
2   <xs:complexContent>
3     <xs:extension base="AbstractBaseResponseType"/>
4   </xs:complexContent>
5 </xs:complexType>
```

XML source code 7.39: PlaceAvailabilitySubscriptionResponseType

PlaceAvailabilitySubscriptionStatusRequestType: Request for getting all subsribed places.
group membership: *SubscriptionRequestGroup*

Element	Type	Comment
(empty)		

PlaceAvailabilitySubscriptionStatusResponseType: Response for getting all subscribed places.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionResponseGroup*

	Element	Type	Comment
□ ⊡	*PlaceID*	`ProviderPlaceIDType`	Subscribed places.

PlaceAvailabilityPushMessageType: Push message containing changes in availability of subscribed places.
group membership: *PushMessageGroup*

	Element	Type	Comment
⊡	*PlaceAvailability*	`PlaceAvailabilityType`	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="PlaceAvailabilityPushMessageType">
2   <xs:sequence>
3     <xs:element name="PlaceAvailability" type="PlaceAvailabilityType" maxOccurs="unbounded"/>
4   </xs:sequence>
5 </xs:complexType>
```

XML source code 7.40: PlaceAvailabilityPushMessageType

Vollständige Standortkapazitätsinformation (Dienst 3a)

CompletePlaceAvailabilityRequestType: Request for getting the complete current status of subscribed places.
group membership: *RequestMessageGroup*

	Element	Type	Comment
□	*MaxPlaces*	`xs:integer`	Defines the maximum number of places to be returned in one message.

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="CompletePlaceAvailabilityRequestType">
2   <xs:sequence>
3     <xs:element name="MaxPlaces" type="xs:integer" minOccurs="0"/>
4   </xs:sequence>
5 </xs:complexType>
```

XML source code 7.41: CompletePlaceAvailabilityRequestType

CompletePlaceAvailabilityResponseType: Response for getting the complete current status of subscribed places. Several such responses may belong to one single request.
Basetype: *AbstractBaseResponseType*
group membership: *ResponseMessageGroup*

	Element	Type	Comment
	MessageBlockID	`xs:token`	ID of the block, this message belongs to. All messages of one block have the same ID.
□	*Last*	`xs:boolean`	If set, this message is the last one in the corresponding block. Otherwise more messages of the same block will follow.
□ ⊡	*PlaceAvailability*	`PlaceAvailabilityType`	

```
1 <xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      ↪ name="CompletePlaceAvailabilityResponseType">
2   <xs:complexContent>
3     <xs:extension base="AbstractBaseResponseType">
4       <xs:sequence>
5         <xs:element name="MessageBlockID" type="xs:token"/>
6         <xs:element name="Last" type="xs:boolean" default="false" minOccurs="0"/>
7         <xs:element name="PlaceAvailability" type="PlaceAvailabilityType" minOccurs="0" maxOccurs="
              ↪ unbounded"/>
8       </xs:sequence>
9     </xs:extension>
10  </xs:complexContent>
11 </xs:complexType>
```

XML source code 7.42: CompletePlaceAvailabilityResponseType

7.9 Dienst 4 - Buchung / Buchungsänderung

Buchung

BookingRequestType: Request for booking a booking target. A booking does not imply a inavailability.
group membership: *OriginDestGroup UserTriggeredRequestGroup*

	Element	Type	Comment
	BookingTargetID	`BookingTargetIDType`	
	TimePeriodProposal	`TimePeriodProposalType`	Proposed time period for the usage of the booking target.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingRequestType">
  <xs:sequence>
    <xs:element name="BookingTargetID" type="BookingTargetIDType"/>
    <xs:element name="TimePeriodProposal" type="TimePeriodProposalType"/>
    <xs:group ref="OriginDestGroup"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.43: BookingRequestType

```
<Ixsi>
    <Request>
        <Transaction>
            <TimeStamp>2014-11-03T11:19:02.258+01:00</TimeStamp>
            <MessageID>100</MessageID>
        </Transaction>
        <Auth>
            <UserInfo>
                <ProviderID>2</ProviderID>
                <UserID>40</UserID>
                <Password>x</Password>
            </UserInfo>
        </Auth>
        <Booking>
            <BookingTargetID>
                <BookeeID>14</BookeeID>
                <ProviderID>2</ProviderID>
            </BookingTargetID>
            <TimePeriodProposal>
                <Begin>2014-11-04T15:21:00.000+01:00</Begin>
                <End>2014-11-04T17:18:00.000+01:00</End>
            </TimePeriodProposal>
        </Booking>
    </Request>
</Ixsi>
```

XML source code 7.44: BookingRequest Beispiel

BookingResponseType: Response for booking a booking target.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
□	*Booking*	`BookingType`	Information on a successful booking.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="Booking" type="BookingType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.45: BookingResponseType

```
<Ixsi>
    <Response>
        <Transaction>
            <TimeStamp>2014-11-03T11:19:02.258+01:00</TimeStamp>
            <MessageID>100</MessageID>
        </Transaction>
        <CalcTime>PT0.325S</CalcTime>
        <Booking>
            <BookingID>2-9</BookingID>
            <TimePeriod>
                <Begin>2014-11-04T15:00:00.000+01:00</Begin>
                <End>2014-11-04T17:30:00.000+01:00</End>
            </TimePeriod>
        </Booking>
    </Response>
</Ixsi>
```

XML source code 7.46: BookingResponse Beispiel

Buchungsänderung

ChangeBookingRequestType: Request for changing a booking.
group membership: *UserTriggeredRequestGroup*

	Element	Type	Comment
	BookingID	`BookingIDType`	Booking to change.
○	*NewTimePeriodProposal*	`TimePeriodProposalType`	New time period proposal for changing the booking.
○	*Cancel*	`xs:boolean`	The booking shall be cancelled.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="ChangeBookingRequestType">
  <xs:sequence>
    <xs:element name="BookingID" type="BookingIDType"/>
```

```
    <xs:choice>
      <xs:element name="NewTimePeriodProposal" type="TimePeriodProposalType"/>
      <xs:element name="Cancel" type="xs:boolean"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.47: ChangeBookingRequestType

ChangeBookingResponseType: Response for changing a booking.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
□	*Booking*	`BookingType`	Information on a successful re-booking.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="ChangeBookingResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="Booking" type="BookingType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.48: ChangeBookingResponseType

7.10 Dienst 5 - Buchungsabonnement

Buchungsabonnement

BookingAlertSubscriptionRequestType: Request for subscribing to alerts for given bookings.
group membership: *SubscriptionRequestGroup*

	Element	Type	Comment
⊡	*BookingID*	`BookingIDType`	Bookings to subscribe/unsubscribe.
□	*Unsubscription*	`xs:boolean`	If set, unsubcribe from the given bookings. Otherwise subscribe to them.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingAlertSubscriptionRequestType">
  <xs:sequence>
    <xs:element name="BookingID" type="BookingIDType" maxOccurs="unbounded"/>
    <xs:element name="Unsubscription" type="xs:boolean" default="false" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.49: BookingAlertSubscriptionRequestType

BookingAlertSubscriptionResponseType: Response for subscribing to alerts for given bookings.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionResponseGroup*

	Element	Type	Comment
	(empty)		

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingAlertSubscriptionResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType"/>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.50: BookingAlertSubscriptionResponseType

BookingAlertSubscriptionStatusRequestType: Request for getting all subscribed bookings.
group membership: *SubscriptionRequestGroup*

	Element	Type	Comment
	(empty)		

BookingAlertSubscriptionStatusResponseType: Response for getting all subscribed bookings.
Basetype: *AbstractBaseResponseType*
group membership: *SubscriptionResponseGroup*

	Element	Type	Comment
□ ⊡	*BookingID*	`BookingIDType`	Subscribed bookings.

BookingAlertPushMessageType: Push message containing alerts for subscribed bookings.
group membership: *PushMessageGroup*

	Element	Type	Comment
⊡	*BookingChange*	BookingChangeType	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="BookingAlertPushMessageType">
  <xs:sequence>
    <xs:element name="BookingChange" type="BookingChangeType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.51: BookingAlertPushMessageType

Vollständige Buchungsabonnementinformation

CompleteBookingAlertRequestType: Request for getting the complete current status of subscribed booking alerts.
group membership: *RequestMessageGroup*

	Element	Type	Comment
□	*MaxResults*	xs:integer	Defines the maximum number of results to be returned in one message.

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CompleteBookingAlertRequestType">
  <xs:sequence>
    <xs:element name="MaxResults" type="xs:integer" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.52: CompleteBookingAlertRequestType

CompleteBookingAlertResponseType: Message for getting the complete current status of subscribed bookings. Several such responses may belong to one single request.
Basetype: *AbstractBaseResponseType*
group membership: *ResponseMessageGroup*

	Element	Type	Comment
	MessageBlockID	`xs:token`	ID of the block, this message belongs to. All messages of one block have the same ID.
□	*Last*	`xs:boolean`	If set, this message is the last one in the corresponding block. Otherwise more messages of the same block will follow.
□ ⊡	*BookingChange*	`BookingChangeType`	

```
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="CompleteBookingAlertResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="MessageBlockID" type="xs:token"/>
        <xs:element name="Last" type="xs:boolean" default="false" minOccurs="0"/>
        <xs:element name="BookingChange" type="BookingChangeType" minOccurs="0" maxOccurs="unbounded"/
            ↪ >
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.53: CompleteBookingAlertResponseType

7.11 Dienst 6 - Preisauskunft

Preisinformation

PriceInformationRequestType: Request for getting a price.
group membership: *OriginDestGroup UserTriggeredRequestGroup*

	Element	Type	Comment
	BookingTargetID	`BookingTargetIDType`	
	TimePeriodProposal	`TimePeriodProposalType`	Proposed time period for the usage of the booking target.
	Distance	`xs:nonNegativeInteger`	Distance to drive in meters.

```xml
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="PriceInformationRequestType">
  <xs:sequence>
    <xs:element name="BookingTargetID" type="BookingTargetIDType"/>
    <xs:element name="TimePeriodProposal" type="TimePeriodProposalType"/>
    <xs:element name="Distance" type="xs:nonNegativeInteger"/>
    <xs:group ref="OriginDestGroup"/>
  </xs:sequence>
</xs:complexType>
```

XML source code 7.54: PriceInformationRequestType

PriceInformationResponseType: Response for getting a price.
Basetype: *AbstractBaseResponseType*
group membership: *UserTriggeredResponseGroup*

	Element	Type	Comment
□	*Tariff*	`TariffType`	

```xml
<xs:complexType xmlns="http://www.ixsi-schnittstelle.de/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ↪ name="PriceInformationResponseType">
  <xs:complexContent>
    <xs:extension base="AbstractBaseResponseType">
      <xs:sequence>
        <xs:element name="Tariff" type="TariffType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML source code 7.55: PriceInformationResponseType

8 Code Tabellen

Die syntaktische Definition für IXSI enthält keine konkreten Werte (Enumerations) für beispielsweise Fahrzeugtypen oder Fehlercodes. Stattdessen werden diese Werte in den nachfolgenden Codetabellen festgelegt. Nur die hier festgelegten Werte dürfen innerhalb von IXSI verwendet werden.

8.1 Fahrzeugklassen

Für die Auswahl *ClassType* können folgende Werte verwendet werden:

Wert	Bedeutung
`bike`	Fahrrad
`motorcycle`	Motorrad
`micro`	Kleinstwagen (z. B. Smart4two)
`mini`	Kleinwagen (z. B. Opel Corsa)
`small`	Kompaktwagen (z. B. VW Golf)
`medium`	Mittelklassewagen (z. B. Audi A4)
`large`	Oberklassewagen (z. B. BMW 7er)
`van`	Van (z. B. VW T5 Multivan)
`transporter`	Transporter (z. B. Ford Transit)

8.2 Antriebsklassen

Für die Auswahl *EngineType* können folgende Werte verwendet werden:

Wert	Bedeutung
none	Kein Kraftantrieb (Muskelkraft)
diesel	Dieselmotor
gasoline	Ottomotor
electric	Elektromotor
liquidgas	Flüssiggas (LPG)
naturalgas	Erdgas (CNG)
hydrogen	Wasserstoffantrieb
hybrid	Hybridantrieb mit Elektro- und Verbrennungsmotor

Hinweis: Ein Pedelec kann als Kombination Fahrzeugklasse bike und Antriebsklasse electric dargestellt werden.

8.3 Fahrzeugeigenschaften

Attribute und Eigenschaften eines Buchungsziels oder eines Standorts können klassifiziert werden, um sie automatisch interpretieren zu können. Dazu werden Attributsklassen verwendet. Attributsklassen werden in IXSI im Datentyp *AttributeClassType* gespeichert. Folgende Werte sind dabei erlaubt:

Wert	Bedeutung
trailer_hitch	Anhängerkupplung
automatic	Automatikgetriebe
convertible	Cabriolet
air_condition	Klimaanlage
navigation	Navigationssystem
cruise_control	Tempomat
winter_tyres	Winter- bzw. Ganzjahresreifen
child_seat_0	Babyschale
child_seat_1	Kindersitz (9-18kg)
child_seat_4	Kindersitz (15-36kg)
utility	Kombi
doors_4	4/5-Türer
seats_9	Mindestens 9 Sitze
seats_7	Mindestens 7 Sitze
seats_5	Mindestens 5 Sitze
seats_4	Mindestens 4 Sitze

8.4 Fehlercodes

Das FVS kann alternativ zu den normalen Response-, Update- und Handshake-Nachrichten auch im Fall eines Fehlers Fehlernachrichten an das RIS schicken. Fehlercodes werden im Datentyp *ErrorCodeType* gespeichert.

Wert	Bedeutung
`auth_provider_unknown`	Authentifizierung: Unbekannte Provider-ID
`auth_invalid_password`	Authentifizierung: User-Passwort-Kombination ungültig
`auth_invalid_token`	Authentifizierung: User-Token-Kombination ungültig
`auth_session_invalid`	Authentifizierung: Session ist ungültig/abgelaufen
`auth_anon_not_allowed`	Authentifizierung: Anonymer User nicht erlaubt
`auth_not_authorized`	Autorisation: Nutzer ist zu dieser Anfrage nicht berechtigt
`sys_backend_failed`	System: Hintergrundsystem antwortet nicht
`sys_unknown_failure`	System: unbekannter Fehler
`sys_not_implemented`	System: Request nicht implementiert
`sys_request_not_plausible`	System: Request ist nicht plausibel. Dieser Wert sollte stets verwendet werden, wenn inhaltlich Fehler im Request enthalten sind.
`booking_target_unknown`	Buchungsziel unbekannt
`price_info_not_available`	Preisinformationen nicht verfügbar
`booking_too_short`	Buchungsdauer zu kurz
`booking_too_long`	Buchungsdauer zu lang
`booking_target_not_available`	Buchungsziel im gegebenen Zeitraum nicht buchbar
`booking_change_not_possible`	Buchungsänderung kann nicht durchgeführt werden
`booking_id_unknown`	Unbekannte Buchungs-ID. Dieser Wert sollte auch verwendet werden, wenn die Buchungs-ID einem anderen User zugeordnet ist.
`language_not_supported`	Angefragte Sprache nicht vollständig unterstützt, andere Sprache geliefert.

Source Code Index

Index