LBSP Routing API Specification

Version 1.0
July 2010

1 Notice

This content is provided under an associated License Agreement or Evaluation Agreement and is subject to the terms thereof.

Table of Contents

2 Introduction

The purpose of this document is to describe the NAVTEQ Routing & Navigation interfaces.

NAVTEQ Routing & Navigation is comprised of several services based on the LBSP Service Framework. Each Service implements a specific routing or navigation capability.

NAVTEQ LBSP services have been designed with the following principles in mind:

- Compliance with NAVTEQ standard formats and data models
- Consistent style for all NAVTEQ LBSP services
- Compliance with industry standards where possible
  - data models (OGC / OpenLS / KML)
  - formats (XML, CSV, JSON, JSONP)
  - protocols (RESTful HTTP, SOAP)
- Extensibility with respect to future requirements
- Self-explaining naming and structures
- Extensive documentation

2.1 Scope

This document describes shared data types for all NAVTEQ Routing & Navigation interfaces and the request and response structures of each of the following services:

- CalculateRoute
- GetRoute
- GetLinkInfo

For the developer of the NAVTEQ Routing and Navigation service, this document provides the specification. Users (application programmers) of NAVTEQ Routing can see how to specify service requests and how to process the corresponding responses.

2.2 Audience

This document is targeted at a technically-oriented audience that either develops the NAVTEQ Routing and Navigation service or use NAVTEQ Routing and Navigation within their applications.

The following groups should read this interface definition document:

- IT & Software Developers
- Business & IT Architects
- Testing and Deployment Team
- Product Management

3 Basic Concepts

3.1 Request Parameter

3.1.1 URL Encoding

Query Parameters have to be encoded in UTF-8, i.e. special characters have to be defined using a %FF hex representation (e.g. “K%C3%B6ln” for “Köln”).

When using HTML forms to post queries to the service interface it is strongly recommended not to rely on the browser settings for parameter encoding but use the accept-charset attribute to specify UTF-8 encoding e.g.

```
<form action=".../6.2/geocode.xml" method="GET" accept-charset="UTF-8">
<!—-  etc. -->
</form>
```

3.1.2 Output Formats

The Service supports xml, json and jsonp as output format. The format is defined in the service endpoint URL:

```
http://{base}/{service}/{version}/{operation}.{format}
```

The format string can be either “xml” or “json”. JSONP (“json with padding”) is supported by using the “json” format identifier and including a query parameter that specifies the callback function (“the padding”):

```
http://.../{operation}.json?...&jsonpcallback=<method name>
```

3.2 Mercator Projection

Mercator spherical is the underlying projection for all resources that are supported LBSP services. The projection supports a seamless projection for the complete world map which ensures that the shape for objects is correct in high zoom levels (conformal projection) and directions are straight, e.g. north/south is up/down and west/east is left/right (cylindrical projection)

Because of the distortion around the poles the map latitudes are from range [-85.0 … 85.0] only.

The image below is an example of a Mercator projected map:

Figure : Mercator projected map

3.3 Quadkey Scheme

The quadkey scheme is an easy to use tile scheme for a complete world map. The scheme is organized in a (quad-) tree structure whereas the root node covers the entire world (-90 <= latitude <= 90, -180 <= longitude <= 180) and every inner node has exactly 4 child nodes for the upper-left, upper-right, lower-right and lower-left portion of its coverage.

Figure : Quadkey Levels

The scheme presents a world map on each level of the underlying tree.

- The width/height of every tile (incl. the root tile) is 256*256 pixels
- The number of tiles in level n is 2^n.
- The width/height of the map at level n is 256 * 2^n (= 2^(n+8))

Mapping applications which follow this scheme can easily be implemented.

- Map Coordinates are actually pixel coordinates that can be mapped to the client's canvas by adding offset values for the canvas upper/left border.
- Zooming is achieved by bit-shifting on the map-coordinates
- Panning is achieved by simple addition/subtraction to actual map-coordinates.

3.3.1 Quadkey Format

For an easy (1-dimensional) addressing of quad-tree tiles in their corresponding (level*level)-map-matrix the x/y-coordinate is transformed into a string. The building rule for the quadkey is derived from the bit presentation of the x- and the y- portion of the tile coordinate as follows:

Base be the char-offset character in the quadkey

- (x, y) be the coefficient of a tile in the map-matrix at level n
- x[n-1] … x[0] be the bit presentation of x (x[0] is least significant bit)
- y[n-1] … y[0] be the bit presentation of y (y[0] is least significant bit)
- quadkey(x,y,n) then becomes the sum (char)(base + 2*y[i] + x[i]) (0 <= i < n)

Facts:

- The empty string "" is the quad-key for the tile at level 0.
- The length of a quad-key equals the level of its corresponding tile the quad-tree.

- Let **q** be the quadkey of a tile, then the quadkey of every of its sub-tiles in the quadkey starts with **q**.

At level 17 the ground resolution is about 1.194 (meters / pixel at equator). This is the maximum zoom level to which the interfaces will be tested. More precise zoom levels (> 17) may be supported by the some resources, but the accuracy of any information that is returned at zoom levels higher than 17 is not guaranteed.

3.3.2 Bitmap Samples

Level 1 (quadkey 1)

MGI	satellite	MS-VirtualEarth

Level 2 (quadkey 12)

MGI	satellite	MS-VirtualEarth

Level 3 (quadkey 120)

MGI	satellite	MS-VirtualEarth

Level 4 (quadkey 1202)

MGI	satellite	MS-VirtualEarth

Level 5 (quadkey 12020)

MGI	satellite	MS-VirtualEarth

Level 7 (quadkey 1202033)

MGI	satellite	MS-VirtualEarth

Level 9 (quadkey 120203302)

MGI	satellite	MS-VirtualEarth

Level 11 (quadkey 12020330202)

MGI	satellite	MS-VirtualEarth

Level 13 (quadkey 1202033020212)

MGI	satellite	MS-VirtualEarth

Level 15 (quadkey 120203302021231)

MGI	satellite	MS-VirtualEarth

Level 17 (quadkey 12020330202123131)

MGI	satellite	MS-VirtualEarth

3.3.3 Converting tile x/y coordinate to quadkey

Converting tile coefficients into quadkeys is in particular necessary for Mapping Clients that want to access the quad-key interfaces. The following (Java) Code Snippet translates the coefficient of a tile in its (level*level) map-tiles matrix into a quadkey:

```
String tileXYToQuadKey(int x, int y, int levelOfDetail, char base)
{
   StringBuffer quadKey = new StringBuffer();
```

```
   for (int mask= (1 << (levelOfDetail – 1)); (mask > 0); mask= mask >> 1)
   {
      char digit= base;

      if ((x & mask) != 0)
     digit+= 1;

      if ((y & mask) != 0)
   digit+= 2;

      quadKey.append(digit);
   }

   return quadKey.toString();

} // tileXYToQuadKey
```

Samples:

- tileXYToQuadKey (x: 2, y: 4, level: 3) returns "210"
- tileXYToQuadKey (x: 4, y: 8, level: 4) returns "2100"
- tileXYToQuadKey (x: 2, y: 4, level: 4) returns "0210"

3.3.4 Converting quadkey to tile x/y coordinate

The length of the quadkey determines the level of detail (levelOfDetail is quadkey.length()). Along the quadkey the index of the corresponding tile (in 2^levelOfDetail*2^levelOfDetail matrix of tiles) is calculated.

```
long[] quadKeyToTileXYLevelOfDetail(String quadkey, char base)
{
   long tileX=          0;
   long tileY=          0;
   long levelOfDetail=        quadkey.length();

   for (int i= 0; (i < levelOfDetail); i++)
   {
        tileX= tileX << 1;
        tileY= tileX << 1;

        int mask= (int)(quadkey.charAt(i) - base);

        if ((mask & 0x01) != 0)
      tileX+=  1;

        if ((mask & 0x02) != 0)
      tileY+=  1;

   }
```

```
    return new long[] { tileX, tileY, levelOfDetail };

} // quadKeyToTileXYLevelOfDetail
```

4 Attribute Switches

Data structures returned by the services can become pretty complex if all available information would be returned. On the other hand, in many use cases, only a sub set of attributes are really required and used by client applications.

Attribute switches can be used in LBSP services to determine which attributes should be included in the response data structure. This helps reducing the data volume and speeds up the server-side processing if not all attributes are required.

4.1 Attribute switch definitions

Attribute switches are defined for each service in a separate chapter. For a complete list of attribute switches available in the Routing service, please refer to Chapter Error: Reference source not found.

To give an example, the PersonAttributeType could define the following switches:

- personId **Short value: id**
- firstName **Short value: fn**
- lastName **Short value: ln**
- maidenName **Short value: mn**
- birthday **Short value: bd**

Switches also have a corresponding short value, which may be used instead of the full attribute value and allows for a more compact representation in query strings. In fact, the short value is the preferred way of specifying attribute switches.

The default set of attributes is defined wherever the attribute switches are used in request structures. Let's say, the PersonAttributeType now defines that the default set of activated switches is {personId, firstName, lastName}.

Two additional values are defined implicitly for all attribute switch groups:

- all (meaning the set of all attributes)
- none (meaning the empty set)

4.2 Adding or skipping attributes

If no attribute switches are specified, the pre-defined default set of attributes is returned, i.e. in this case the default attributes {personId, firstName, lastName}.

The semantics of switches is to add or remove certain attributes from the default attribute set. The "-" sign as prefix is used to indicate that an attribute should be skipped.

So the following request would remove the firstName attribute from the default list and return only the attribute set {personId, lastName}..

http://...?...&personattributes=-fn

To add attributes to the default set, switches are specified without the "-" prefix. As the "+" sign can be problematic in URL encoded query parameters for HTTP based backend services, "+" signs are just omitted. The following example would return the attribute set {personId, firstName, lastName, birthday}.

http://...?...&personattributes=bd

4.3 Using the shortcut values "all" and "none"

If all attributes besides the birthday is required, you can make use of the implicitly defined "all" value and combine with a switch to remove some attributes. The following example would return the attribute set {personId, firstName, lastName, maidenName}.

http://...?...&personattributes=all,-bd

If you only want to specify a small set of attributes, the shortcut value "none" can be combined with switches to add desired attributes. The following example would only return {firstName, lastName}.

http://...?...&personattributes=none,fn,ln

The last approach can also be used if you want to specify the exact list of attributes to be returned and to not want to rely on the pre-defined default attribute set.

5 Types used by all services

This chapter describes data types which are shared across all services.

5.1 Built-in Types

This chapter defines the representation formats for XML schema built-in types.

If not explicitly specified, representation formats for built-in types are used as defined in the by the W3C in XML Schema Part 2: Datatypes Second Edition.

5.1.1 xs:dateTime

The lexical representation of xs:dateTime consists of finite-length sequences of characters of the following form:

YYYY + **"-"** + MM + **"-"** + DD + **"T"** + hh + **":"**+ mm + **":"** + ss + (**"."** + s)? + (zzzzzz)?

YYYY	'-'? yyyy is a four-or-more digit optionally negative-signed numeral that represents the year; if more than four digits, leading zeros are prohibited, and '0000' is prohibited
MM	is a two-digit numeral that represents the month
DD	is a two-digit numeral that represents the day
hh	is a two-digit numeral that represents the hour The value 24 is permitted if the minutes and seconds represented are zero, and the dateTime value so represented is the first instant of the following day (the hour property of a dateTime object in the value space cannot have a value greater than 23)
mm	is a two-digit numeral that represents the minute
ss	is a two-integer-digit numeral that represents the whole seconds
s	represents the fractional seconds (optional)
zzzzzz	represents the timezone

The timezone portion zzzzzz is represented as follows:

((**"+"** | **"-"**) + hh + **":"** + mm) | **"Z"**

"+" | **"-"**	• **"+"** indicates a non-negative duration, • **"-"** indicates a non-positive duration.
hh	is a two-digit numeral (with leading zeros as required) that represents the hours
mm	is a two-digit numeral that represents the minutes
"Z"	"Z" represent the zero-length duration timezone, UTC "Z" is the canonical representation for UTC dateTime values.

Example:

2002-10-10T12:00:00-05:00 (noon on 10 October 2002, Central Daylight Savings Time as well as Eastern Standard Time in the U.S.) is 2002-10-10T17:00:00Z, five hours later than 2002-10-10T12:00:00Z.

5.1.2 xs:duration

The lexical representation for duration is the ISO 8601 extended format:

"-"? + **"P"** + (nY + **"Y"**)? + (nM + **"M"**)? + (nD + **"D"**)? + (**"T"** (nH + **"H"**)? + (nM + **"M"**)? + (nS + **"S"**)?)?

"-"	Optional minus sign to represent negative durations. If the sign is omitted a positive duration is indicated.
nY	Represents the number of years. Must be an unsigned integer.
nM	Represents the number of months. Must be an unsigned integer.
nD	Represents the number of days. Must be an unsigned integer.
nH	Represents the number of hours. Must be an unsigned integer.
nM	Represents the number of minutes. Must be an unsigned integer.
nS	Represents the number of seconds, which can include decimal digits represent fractions of seconds.

Example:

P1Y2M3DT10H30M indicates a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes.

P120D indicates a duration of minus 120 days.

5.1.3 List Types

XML Schema provides a list type which allows including more than one item within a single XML element. Items are separated by space.

The following example defines a list element named "Foo" which contains string elements.

```
<xs:simpleType name="MyListType">
   <xs:list itemType="xs:string"/>
</xs:simpleType>
<xs:element name="Foo" type="MyListType">
```

A valid instantiation of the “Foo” element would be

```
<Foo>ABC DEF GHI</Foo>
```

In this example, the “Foo” element contains the three items “ABC”, “DEF”, and “GHI”.

In representation formats other than XML, the space as separator sometimes becomes unhandy. Therefore, a standard representation format for list types is defined for each representation format below.

Query Parameter Representation

In query strings, items of list types are separated by one of the following characters:

“,” (comma) is used at top level

<Foo[0]> + “,” + <Foo[1]> + “,” + <Foo[2]> + “,” + …

“;” (semicolon) is used at second level, or if comma is already used in the representation of the list items

<Foo[0]> + “;” + <Foo[1]> + “;” + <Foo[2]> + “;” + …

“!” (exclamation point) is used at third level, or if comma and semicolon are already used in the representation of the list items

<Foo[0]> + “!” + <Foo[1]> + “!” + <Foo[2]> + “!” + …

The following syntax is introduced as shortcut to represent list types:

<Foo[]>

Note: 1 Note: list types and element sequences are represented in the same manner (see also Chapter Error: Reference source not found).

JSON Representation

List types will be represented as arrays:

[<list[0]>, <list[1]>, <list[2]>, …]

5.1.4 Sequence of elements of same type

A schema can define elements that can appear more than once. The following example defines an element named “Bar” which can be included n times.

```
    <xs:element name="Bar" type="xs:string"
maxOccurs="unbounded"/>
```

A valid instantiation of the “Bar” element would be:

```
<Bar>ABC</Bar>
<Bar>DEF</Bar>
<Bar>GHI</Bar>
```

Element sequences are described in all API documentations with an "[]" as suffix like this:

- Bar [] sequence of strings

In representation formats other than XML, listing the same element more than once can become unhandy. Therefore, a standard representation format for sequences is defined for each representation format below.

Query Parameter Representation

In query strings, sequence items are separated by ";" (semicolon) character:

"," (comma) is used at top level

<Bar[0]> + "," + <Bar[1]> + "," + <Bar[2]> + "," + …

";" (semicolon) is used at second level, or if comma is already used in the representation of the list items

<Bar[0]> + ";" + <Bar[1]> + ";" + <Bar[2]> + ";" + …

"!" (exclamation point) is used at third level, or if comma and semicolon are already used in the representation of the list items

<Bar[0]> + "!" + <Bar[1]> + "!" + <Bar[2]> + "!" + …

When applying this rule to the above example, the result would be:
ABC, DEF, GHI

The following syntax is introduced as shortcut to represent element sequences:

<Bar[]>

Note: 2 Note: list types and element sequences are represented in the same manner (see also Chapter Error: Reference source not found).

JSON Representation

Element sequences will be represented as arrays:

[<list[0]>, <list[1]>, <list[2]>,…]

5.2 Simple Types

This chapter defines the representation formats for types extending or restricting XML schema built-in types.

Note: 3 **If not explicitly specified, representation formats for simple types are identical to the XML representation.**

5.2.1 LatitudeType

Designates the location of a place on Earth north or south of the equator.

LatitudeType is an xs:double type with the following restrictions:

Must satisfy: -90 ≤ value ≤ 90

Unit: decimal degrees

Reference System: WGS 84

Precision**: 7 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
45.1234567
-12.3456789
45.123
-12.3

5.2.2 LongitudeType

Designates Location of a place on Earth east or west of the prime meridian.

LongitudeType is an xs:double type with the following restrictions:

Must satisfy: -180 ≤ value ≤ 180

Unit: decimal degrees

Reference System: WGS 84

Precision**: 7 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
90.1234567
-120.3456789
90.123
-120.3

5.2.3 AltitudeType

The altitude specifies the elevation of a point above mean sea level.

AltitudeType is an xs:double type with the following restrictions:

Unit: meters

Precision**: 2 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
300
-12
512.50

5.2.4 QuadKeyType

A quadkey uniquely identifies a single tile at a particular level of detail. For more detailed information on the quadkey scheme, please refer to Chapter Quadkey Scheme.

QuadkeyType is an xs:string type with the following restrictions:

Must match regular expression**:** [0-3]+

Format: as defined in Chapter Quadkey Format

Query Parameter Representation

Valid examples:
12020330202123131
12010310212

5.2.5 SpeedType

This type can be used whenever speed information needs to be represented.

SpeedType is an xs:double type with the following restrictions:

Unit: m/s

Precision**: 2 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
90.12
65.1
30

5.2.6 DistanceType

This type can be used whenever distances need to be represented.

DistanceType is an xs:double type with the following restrictions:

Unit: meters

Precision**: 4 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
45.1234
899344

5.2.7 WeightType

This type can be used whenever weights need to be represented (e.g. when defining truck restrictions on links).

WeightType is an xs:double type with the following restrictions:

Must satisfy: 0 ≤ value

Unit: t (metric tons)

Precision**: 3 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
5.123
5

5.2.8 HeadingType

Heading in degrees starting at true North and continuing clockwise around the compass. North is 0 degrees, east is 90 degrees, south is 180 degrees, and west is 270 degrees. This type can be used whenever directions need to be represented.

HeadingType is an xs:double type with the following restrictions:

Must satisfy: 0 ≤ value < 360

Unit: decimal degrees

Measured: clockwise starting at 0 degrees at true North

Precision**: 4 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
90.12
65.1
30

5.2.9 AccuracyType

The accuracy type is used to specify the accuracy level of measured coordinates. The accuracy level is specified in meters and must be a non-negative real number.

AccuracyType is an xs:double type with the following restrictions:

Must satisfy: 0 < value

Unit: meters

Precision**: 4 positions after decimal point, additional positions are ignored**

Query Parameter Representation

Valid examples:
0.5
10

5.2.10 FunctionalClassType

The functional class is used to classify roads depending on the speed, importance and connectivity of the road.

The value represents one of the five levels:

- 1: allowing for high volume, maximum speed traffic movement
- 2: allowing for high volume, high speed traffic movement
- 3: providing a high volume of traffic movement
- 4: providing for a high volume of traffic movement at moderate speeds between neighborhoods
- 5: roads whose volume and traffic movement are below the level of any functional class

FunctionalClassType is an xs:byte type with the following restrictions:

Must satisfy: 1 ≤ value < 5

Query Parameter Representation

Valid examples:
1
2
3

5.2.11 ElementReferenceType

This type is used whenever an element needs to be cross-referenced. The Common Data Model does explicitly not use the xs:ID construct to make element referencing transparent to other representation formats than XML.

ElementReferenceType is an xs:string with no further restrictions.

5.2.12 LinkIdType

A network link can be identified using the LinkIdType as functional key. It is used by different complex types (e.g. RouteLinkType and MapLinkType).

LinkIdType is an xs:string type with the following restrictions:

Must match the following regular expression: [-]\d+

Query Parameter Representation

No special representation format for query parameters. Format is used as defined above.

5.2.13 TMCCodeType

TMCCodeType is an xs:string type with the following restrictions:

Fixed length: 9 characters

Must match regular expression**:** \w\d{2}[+-PN]\d{5}

Format: B + CC + D + EEEEE, where the following rules apply:

B	The one character EBU Country Code. Note: EBU are defined for European Countries. There are no official EBU codes for Canada and the U.S. NAVTEQ has defined "C" for Canada and "1" for the U.S.
CC	The two digit Location Table number.
D	is the one character RDS direction, where: • **"+"**is in the positive direction and external to the Problem Location. • **"-"** is in the negative direction and external to the Problem Location. • **"P"**is in the positive direction and internal to the Problem Location. • **"N"** is in the negative direction and internal to the Problem Location.
EEEEE	The five digit Location Code. This has leading zeros if necessary.

Query Parameter Representation

No special representation format for query parameters. Format is used as defined above.

5.2.14 LanguageCodeType

This type is used whenever a language of a value needs to be referenced.

LanguageCodeType is an xs:string type with the following restrictions:

Format: Language code following RFC5646

Query Parameter Representation

Valid examples:
de
en-US

5.2.15 LanguageCodeParameterType

This type is used whenever a language filter needs to be specified.

LanguageCodeparameterType is an xs:string type with the following restrictions:

Format: Language code following RFC4647

Query Parameter Representation

Valid examples:
de
en-US

5.2.16 CountryCodeType

This type is used whenever a country needs to be referenced.

CountryCodeParameterType is an xs:string type with the following restrictions:

Format: Country code according to ISO 3166-1-alpha-3

Query Parameter Representation

Valid examples:
DEU
GBR

5.2.17 ColorType

This type is a marker type to identify color information.

ColorType is an xs:string type with the following restrictions.

Format: RGB value in CSS format

Query Parameter Representation

Valid examples:
#FFEEFF
#0066A5

5.2.18 SideOfStreetType

Designates values to indicate on which side of a link an object is placed.

The following enumeration values are available for SideOfStreetType:

- left
- right
- neither

Query Parameter Representation

SideOfStreetType values are expected to be specified verbatim as defined above.

Allowed values: "left", "right", "neither"

5.2.19 TextSemanticsType

Available types of text semantics used in type AlternativeValue.

The following enumeration values are available for TextSemanticsType:

- synonym
- exonym
- unclassified

Query Parameter Representation

TextSemanticsType values are expected to be specified verbatim as defined above.

Allowed values: "synonym", "exonym", "unclassified"

5.2.20 NameTypeType

Designates available name types used in type AlternativeValue.

The following enumeration values are available for NameTypeType:

- baseName
- shortBaseName
- abbreviation

Query Parameter Representation

NameTypeType values are expected to be specified verbatim as defined above.

Allowed values: "baseName", "shortBaseName", "abbreviation"

5.3 Complex Types

5.3.1 GeoCoordinateType

The **Geo**CoordinateType represents a geographical position (coordinate).

Figure : GeoCoordinateType

As most of the services currently do not consider height information, the altitude is defined as optional.

The following attributes are defined:

- **Latitude** WGS-84 / degrees with (-90 <= Latitude <= 90), precision is limited to 7 positions after decimal point
- **Longitude** WGS-84 / degrees with (-180 <= Longitude <= 180) precision is limited to 7 positions after decimal point
- **Altitude** Altitude in meters. Optional.

Query Parameter Representation

Whenever a coordinate needs to be represented in query strings, use the following format:

<Latitude> + "," + <Longitude> (+ "," + <Altitude>)?

5.3.2 GeoShiftedCoordinateType

A coordinate is usually defined in the WGS84 system. Some countries however request for a country specific shift of their coordinates to obfuscate their maps. If a coordinate is defined in such a shifted system, the **GeoShifted**CoordinateType is used.

Figure : GeoShiftedCoordinateType

The following attributes are defined in addition to those derived from **GeoCoordinateType**:

- **Shift** The shift attribute will define which system has been used to calculate the position.

Query Parameter Representation

Whenever a coordinate needs to be represented in query strings, use the following format:

<Latitude> + "," + <Longitude> (+ "," + <Altitude>)?

5.3.3 CoordinatesArrayType

In many use cases, coordinates appear in large series. However, using a sequence of many Coordinate typed elements in data structures might bloat the document instances inadequately.

To provide a more compact format, the CoordinatesArray type allows concatenating serialized coordinates within a single data element.

The CoordinatesArray is list type (see also Chapter Error: Reference source not found) with items of type xs:string, where the following restrictions apply:

Item separator**: space character (according to the xs:list definition)**

Allowed item types: Coordinate, in query parameter representation

The list is ordered, but the order does not have any defined semantics. If you wish to be more distinct regarding the semantics of the listed coordinates, you might find the Polyline or Polygon type useful.

XML Representation

The following example illustrates how the CoordinatesArray looks like for a list of two 2 dimensional coordinates:

Code Sample
`<Foo>37.7914050,-122.3987030 37.7866569,-122.4026513</Foo>`

Query Parameter Representation

According to the standard representation format for list types, the CoordinatesArray would be represented in query strings as:

<Coordinate[]>

Notes on representation formats:
Coordinate see representation format for CoordinateType

The XML example would be represented as query string as:
37.7914050,-122.3987030;37.7866569,-122.4026513

5.3.4 GeoPolylineType

While the CoordinatesArrayType merely defines an ordered list of coordinates, the PolylineType can be used to express that the subsequent connection between the points form a polyline.

The PolylineType directly derives from the CoordinatesArrayType and thus can be represented in the same manner.

5.3.5 GeoPolygonType

While the CoordinatesArrayType merely defines an ordered list of coordinates, the PolygoneType can be used to express that the subsequent connection between the points form a polygon. In contrast to the PolylineType, it is assumed that the last coordinate within the list is connected with the first coordinate and thus constructs an enclosed geometry.

The PolylineType directly derives from the CoordinatesArrayType and thus can be represented in the same manner.

5.3.6 GeoBoundingBoxType

A **Geo**BoundingBoxType defines a rectangular area in a geographic coordinate system. This type is derived from the abstract base type **GeoAreaType**.

Figure : GeoBoundingBoxType

As the bounding box is specified by its top-left and bottom-right corner the box, it is not necessarily the smallest rectangle spanned by these two points; it is possible to define bounding boxes that are wider than 180° or higher than 90° (e.g. by setting the longitude of top-left corner to a bigger value than the longitude of the bottom-right corner).

A bounding box with longitude of ▲180° for the top-left corner and a longitude of 180° for the bottom-right corner constructs an area that encircles the globe whereas a bounding box with the same longitude values for both corners constructs a bounding box of width 0°.

The GeoBoundingBoxType has the following attributes:

- **TopLeft** Top-left corner of the rectangular area.
  - **Latitude** WGS-84 / degrees with (-90 <= Latitude <= 90)
  - **Longitude** WGS-84 / degrees with (-180 <= Longitude <= 180)
  - Altitude **Altitude in meters (optional)**
- **BottomRight** Bottom-right corner of the rectangular area.
  - **Latitude** WGS-84 / degrees with (-90 <= Latitude <= 90)
  - **Longitude** WGS-84 / degrees with (-180 <= Longitude <= 180)
  - Altitude **Altitude in meters (optional)**

Figure : The red rectangle in the image below was specified as a BoundingBox

Query Parameter Representation

A BoundingBoxType is represented in a compact format if used as query parameter:

<TopLeft.Latitude> + “,” + <TopLeft.Longitude> + “,” + <BottomRight.Latitude> + “,” + <BottomRight.Longitude>

If possible, i.e. if there is no risk of name clashes, the preferred name for the parameter of a BoundingBoxType element is “bbox”.

Example:
bbox=37.7902858,-122.4027371,37.7890649,-122.3993039

5.3.7 GeoProximityType

A **Geo**ProximityType represents a circular area. This type is derived from the abstract base type **GeoAreaType**.

The ProximityType has the following attributes:

- **Center**Center of the circular area.
  - **Latitude** WGS-84 / degrees with (-90 <= Latitude <= 90)
  - **Longitude**WGS-84 / degrees with (-180 <= Longitude <= 180)
  - Altitude **Currently unused in Proximity**
- **Radius** Radius in meters

Query Parameter Representation

The **GeoProximityType** can be represented in query strings as follows:

<Latitude> + “,” + <Longitude> (+ “,” + <Radius>)?

If possible, i.e. if there is no risk of name clashes, the preferred name for the parameter of a ProximityType element is “prox”.

Example:
prox=37.7890649,-122.4027371,1000

5.3.8 GeoCorridorType

A **Geo**CorridorType is a two-dimensional area consisting of all points within a given width of a line. This type is derived from the abstract base type **GeoAreaType**.

The GeoCorridorType has the following attributes:

- **Line** List of coordinates representing a geographical line as center of the area (see **Geo**PolylineType)
- **Width** Corridor widths in meters

Query Parameter Representation

The Corridor can be represented in query strings as follows:

<Line> + “;” + <Width>

Notes on representation formats:
<Line> see representation format for GeoPolylineType
<Width> see representation format for DistanceType

If possible, i.e. if there is no risk of name clashes, the preferred name for the parameter of a CorridorType element is “corridor”.

Example:
corridor=37.7914050,-122.3987030;37.7866569,-122.4026513;1000

5.3.9 PeriodType

The PeriodType represents a time period between two distinct points in time.

Figure : PeriodType

The PeriodType has the following attributes:

- From Start date and time of the period
- Until End date and time of the period

Query Parameter Representation

The Period can be represented in query strings as follows:

<From> + “,” + <Until>
Note: Null values are represented as “-“

Notes on representation formats:
<From>, <Until> The representation formats for xs:datetime apply.

5.3.10 GeoPositionType

This type represents a snapshot of positioning information. This information is usually provided by a GPS device.

Figure : GeoPositionType

GeoPositionType defines the following attributes:

- Coordinate Coordinate position that has been measured by the hosted device
- Timestamp Time at which the position has been measured
- Accuracy Accuracy level of the latitude and longitude coordinates
- AltitudeAccuracy Accuracy level of the altitude value.
- Heading Direction of travel of the hosting device
- Speed Current ground speed of the hosting device

5.3.11 KeyValuePairType

This type is a generic container for arbitrary information. Key/value pairs are supported to transport non typed generic information which has not (yet) been defined in API data structures.

Figure : KeyValuePairType

The following attributes are defined:

- Key Name of the property

The value **is included directly in the body of the element.**

Query Parameter Representation

A KeyValuePair can be represented in query strings as follows:

<Key> + "," + <Value>

Note on representation formats:
Special characters need to be URL-encoded.

5.3.12 AlternativeValueType

Generic container to keep textual values for any attribute in different languages and/or with different semantics and/or type.

Figure : AlternativeValueType

AlternativeValueType defines the following attributes:

- Key Attribute/field name
- Value Text value in selected language and with selected semantics and type
- Type Type of the alternative text (see NameType)
- Semantics Semantics of the alternative text (see TextSemanticsType)
- Language Language of the alternative text (RFC 5646 Language Tag)

5.3.13 AddressType

Address record of a Location. Attributes are normalized to US feature names and can be mapped to the local feature levels (e.g. State matches "Bundesland" in Germany) using the NAVTEQ mapping tables for GDF.

Figure : AddressType

AddressType defines the following attributes:

- Label Assembled address value for displaying purposes.
- Country ISO 3166-alpha-3 country code
- State Includes the first subdivision level(s) below the country. The Nokia PNSS "region" is directly mapped to Address.state. For GDF, a country-wise mapping is required.

- County Includes the second subdivision level(s) below the country. Use of this field is optional if a second subdivision level is not available.
- City Refers to the locality of the address. The Nokia PNSS "city" is directly mapped to Address.city. For GDF, a country-wise mapping is required.
- District Includes the subdivision level below the city. Use of this field is optional if a second subdivision level is not available.
- Street Street name in the requested primary language. For API implementations which do not support separate street and house number fields, this field may also contain the house number value.
- HouseNumber House number. Depending on regional characteristics, can also be house name. For API implementations which do not support separate street and house number fields, this field can be omitted.
- PostalCode Postal code
- Floor Floor number
- Suite Suite number or name
- AlternativeAttributes Attributes in different languages and/or with different semantics. (see AlternativeValueType)
- AdditionalData Generic key/value container to keep additional attributes. The defined key/values are:
  - "CrossingStreet0"
    First cross street near matched address
  - "CrossingStreet1"
    Second cross street near matched address. This value is omitted by the search module, if the location's distance to CrossingStreet0 is less than half of the distance to CrossingStreet1.

6 Types used in all Routing Services

This chapter describes data types which are shared across all routing services.

6.1 Objects

There are a number of object types which are common to the routing domain and therefore are shared across several different NAVTEQ Routing & Navigation services. Most typically, the domain objects described in this chapter are used as result objects in response structures.

6.1.1 RouteType

A Route describes a distinct path through the route network covering two or more waypoints. It consists of route legs each of them covering the section between two waypoints.

Figure : RouteType

RouteType defines the following attributes:

- RouteId Unique identifier of the route. The RouteId is calculated from the route links and can be used to reproduce any previously calculated route. Thus, the RouteId is not a temporary ID, but can be used as persistent reference to the route.
- Waypoint List of waypoints that have been defined when requesting for a route calculation.
  The first waypoint is defined as the start of the route; the last waypoint marks the destination. Any points in between the two define via points.
- Mode Settings for route calculation. One mode can be specified for each route. (see RoutingModeType)
- Shape Shape of the route as a polyline. The accuracy might depend on the resolution specified in mpp (meters per pixel) when requesting the route.
  In some use cases (like web portals), only the route's shape is required without the nested structure of a route and detailed knowledge of the links and LinkIds. In this case, the shape does not need to be acquired by traversing the route's links, but can be represented using this attribute at route level.
- BoundingBox Bounding Box of the route.
- Leg Partition of the route into legs between the different waypoints. (see RouteLegType)
  - Start Route waypoint that is located at the start of this route leg. This waypoint matches one of the waypoints in the Route.
  - End Route waypoint that is located at the end of this route leg. This waypoint matches one of the waypoints in the Route.
  - Length Length of the leg.
  - TravelTime Time needed to travel along this route leg. Depending on the routing type traffic information is considered in this value.

- Maneuver List of all maneuvers which are included in this portion of the route.
- Link List of all links which are included in this portion of the route.

- **PublicTransportLine** List of all public transport lines which are used by public transport links and maneuvers of the route. (see PublicTransportLineType)
- **Note** Notes that are either related to the calculation (violated routing options) or that refer the route as a whole. In addition to these notes additional notes can be attached to maneuvers. The maneuver notes are usually related to the route segment following the maneuver and would be of interest when passing this segment. (see RouteNoteType)
- **Summary** Overall route distance and time summary. (see RouteSummaryType)

Note: 4 **SummaryByCountry** Route distance and time summary per traversed country. (see RouteSummaryByCountryType)
Note: not yet supported

6.1.2 RouteLegType

A RouteLeg defines the portion of a route between two way points.

Figure : RouteLegType

RouteLegType defines the following attributes:

- Start Route waypoint that is located at the start of this route leg. This waypoint matches one of the waypoints in the Route.
- End Route waypoint that is located at the end of this route leg. This waypoint matches one of the waypoints in the Route.
- Length Length of the leg.
- TravelTime Time needed to travel along this route leg. Depending on the routing type traffic information is considered in this value.
- Maneuver List of all maneuvers which are included in this portion of the route. (see **ManeuverType**)
- Link List of all links which are included in this portion of the route.

6.1.3 ManeuverType

A maneuver describes the action needed to leave a street segment and enter the next link following the route. This type is an abstract base class for **PrivateTransportManeuver** and **PublicTransportManeuver** and only includes most common attributes which every maneuver type provides.

Figure : ManeuverType

ManeuverType defines the following attributes:

- @id Key that identifies this maneuver element uniquely within the response.
- Position Position where the maneuver starts.
- Instruction Textual instruction describing the required maneuver like "Turn left onto Minna St."
- PlaceEquipment List of obstacles at the maneuver.

- TravelTime Travel time needed for the segment following this maneuver until the next maneuver. Depending on the routing type traffic information is considered in this value.
- Length Length of the segment following this maneuver until the next maneuver.
- Shape Shape of the route segment following this maneuver until the next maneuver as a polyline.
- Time Point in time when this maneuver is supposed to take place. For public transport maneuvers the time denotes the arrival or departure time according to the line's schedule. For private transport routes that have been calculated based on an explicit arrival or departure time the time denotes the predicted time of the maneuver. The time information is given in the time zone of the maneuver's position.
- Note Additional information about the route segment following the maneuver, e.g. "sharp curve ahead", "accessing toll road", etc.
- NextManeuver Reference to the next maneuver on the recommended route.
- ToLinkThe key of the next (outgoing) link.

6.1.4 RouteLinkType

A link is an edge in the routing network. This type is an abstract base class for PrivateTransportLink and PublicTransportLink and only includes most common attributes which every link type provides.

Figure : RouteLinkType

RouteLinkType defines the following attributes:

- LinkId The unique Identifier of a NAVTEQ Link. The sign indicates weather the link is driven/walked from reference node to node (no sign) or from node to reference node (-).
- Shape Shape of the link. (see PolylineType)
- Length Length of the link in meters.
- RemainDistance Distance from the start of this link until the destination is reached.
- RemainTime Predicted time needed from the start of this link until the destination will be reached. Traffic information is taken into account if the user is authorized to use traffic information.
- AdditionalData Generic container to store additional information
- NextLink Reference to the next link on the recommended route.
- Maneuver Reference to the maneuver which needs to take place at the end of the link.

6.1.5 RouteSummaryType

Route summary information for the whole route.

Figure : RouteSummaryType

RouteSummaryType defines the following attributes:

- Distance Total travel distance
- TrafficTime Total travel time considering traffic
- BaseTime Total travel time not considering traffic but taking the transport mode into account.
- Flags Special link characteristics (like ferry or motorway usage) which are covered by the route
- Entries A route summary contains a rough description of the route limited to the essential streets and maneuvers. Each of these maneuvers/routes is described in a route summary entry.
  - Label Textual description of the summary entry.
  - Distance Travel distance for the part of the route represented by the summary entry.

6.1.6 RouteSummaryByCountryType

Route summary information per country.

Figure : RouteSummaryByCountryType

In addition to its base type RouteSummaryType, the RouteSummaryByCountryType defines the following attributes:

- Country Country code

6.1.7 PrivateTransportManeuver

This type represents a maneuver relevant for private transport such as car, truck, pedestrian. This type is derived from the abstract typeManueverType.

Figure : PrivateTransportManeuverType

PrivateTransportManeuverType defines the following attributes in addition to those derived from ManeuverType:

- Direction Maneuver direction hint. Please refer to the enumeration type Direction for supported values.
- Action Code that identifies the action for this maneuver. Please refer to the enumeration type PrivateTransportAction for supported values.
- RoadName Name of the road the maneuver where the maneuver starts
- SignPost Name of the street as given on the sign post
- NextRoadName Name of the next road the maneuver is heading for.
- RoadNumber Number of the road where the maneuver starts (e.g. A5, B49, etc.)
- NextRoadNumber Number of the road (e.g. A5, B49, etc.) the maneuver is heading for.
- Lane List of lanes at maneuver position; each lane supporting a single line of vehicles.

Note: 5 Note: not yet supported

- Direction List of directions that are allowed on the link. This corresponds to the arrows usually given on the road.
- SelectedLanes List of lanes which can be used for this maneuver. Lanes are references by their list index starting at index 0.
- RoadTemplate External SVG resource that provides a template for displaying the route (e.g. A5, B49, etc.). It is the client's responsibility to add the route name into the template to create the correct image for the maneuver.

Note: 6 Note: not yet supported

6.1.8 PrivateTransportLinkType

A private transport link is a link traversed in a route using private transport as car, truck, pedestrian, etc.

Figure : PrivateTransportLinkType

PrivateTransportLinkType defines the following attributes in addition to those derived from RouteLinkType:

- SpeedLimit Legal speed limit (as static speed information associated with the link independent of the vehicle type)
-
- DynamicSpeedInfo Dynamic speed information on the link which can change over time.
  - TrafficSpeed Traffic enabled speed. Estimated speed when considering all traffic relevant constraints.
  - TrafficTime Traffic enabled time. Estimated time spent on this link based on the TrafficSpeed. There might be additional penalties taken into account when calculating the duration which can therefore be greater than link.length / trafficSpeed.
  - BaseSpeedBase speed. Estimated speed when considering no traffic relevant constraints.
  - BaseTime Base time. Estimated time spent on this link based on the base speed. There might be additional penalties taken into account when calculating the duration which can therefore be greater than link.length / baseSpeed
  - JamFactor The number between 0.0 and 10.0 indicating the expected quality of travel. As the number approaches 10.0 the quality of travel is getting worse. -1.0 indicates that a Jam Factor could not be calculated
  - JamFactorTrend The number between -1.0 and +1.0 indicating the trend of the jam factor over a period of time. As the number approaches +1.0 the jam factor is getting worse.
  - Confidence This indicates the level of confidence that Traffic.com has in the flow data. This will be a value from 0.0 to 1.0 where 1.0 is the highest level of confidence (i.e. dense sensor coverage) and 0.0 is the lowest level of confidence (i.e. completely estimated). The confidence value of the predicted flow will take into account the current flow confidence.
- ExternalResource Series of references to external resources (e.g. bitmaps). The client is responsible for downloading the referenced resource.

- ResourceType The semantics of the resource. Please refer to the enumeration type ResourceType for a list of supported values.
- Filename Filename of the resource

- Flags Flags that describe special characteristics of the link like "motorway", "tollroad", "ferry", etc. (see LinkFlag enumeration for possible values)
- FreewayExit Name of the freeway exit the link is passing
- FreewayJunction Name of the freeway junction the link is passing
- TMCCodes List of TMCCodes that cover the link.
- FunctionalClass The functional class assigned to the link.
- Address Address of the street the link is passing
- RoadNumber Number of the road (e.g. A5, B49, etc.) the link is passing
- Timezone Timezone (including information about daylight saving) the link is located in.
- Incident An incident describes a temporary event on a route network link. It typically refers to a real world incident (accident, road construction, weather condition, etc.) on a street or street segment.
  - ValidityPeriod Time period when the incident is relevant
  - Text A textual description of the event
  - Type Classifier for the incident; there is currently no semantics or value restriction defined for this attribute
  - Criticality Criticality on an integer scale:

0 = critical

1 = major

2 = minor

3 = low impact

- CorridorLevel The corridor level defines a logical distance from the base route. Any time a "faulty" maneuver is detected the level is increased by 1. The base route has a constant corridor level of 0.
  Note: not yet supported
- Stubs Reference keys to other links within the corridor, which are not part of the recommended route, but can be entered from this link.
  Note: not yet supported
- TruckRestrictions Restrictions that apply to trucks on this link. (Only relevant if truck has been selected as TransportMode). Please refer to TruckRestrictionsType for detailed information on this type.

6.1.9 TruckRestrictionsType

This type is used to describe which restrictions apply on a link for trucks.

Figure : TruckRestrictionsType

TruckRestrictionsType defines the following attributes:

- TrailerForbidden Indicates that truck with trailers may not pass
- ForbiddenHazardousGoods

- Types of hazardous goods which are forbidden. Please refer to the enumeration type HazardousGoodType for supported values.
- PermittedGrossWeight
- Maximum permitted gross weight of the truck in tons
- LimitedWeight Maximum weight in tons
- WeightPerAxle Maximum weight per axle in tons
- TrailerWeight Maximum trailer weight in tons
- Height Maximum truck height in meters
- Width Maximum truck width in meters.
- Length Maximum truck length in meters.

6.1.10 PublicTransportManeuverType

This type represents a maneuver relevant for public transport such as bus, train. This type is derived from the abstract type ManueverType.

Two different types of public transport maneuvers are supported: "Enter" and "Leave" (see enumeration type PublicTransportAction). A direct change between two transport lines will be represented with two maneuvers: one for the action "Leave" and a subsequent one to "Enter" the next transport line.

Figure : PublicTransportManeuverType

In addition to its base type ManueverType, the PublicTransportManeuverType defines the following attributes:

- Action Identifier for the action to be performed at this maneuver
- StopName Name of the stop where the user has to leave (action == "Leave") or enter (action == "Enter") the transport line.
- PlatformNamePlatform name where the transport line stops (action == "Leave") or starts from (action == "Enter").
- PlatformLevel Platform level where the transport line stops (action == "Leave") or starts from (action == "Enter").
- Time Arrival (action == "Leave") or departure (action == "Enter") time of the public transport line.
- Line Reference key to the PublicTransportLine object. To reduce data volume, the PublicTransport element is not directly embedded in the ManeuverType object, but is swapped out into the Route element.

6.1.11 PublicTransportLinkType

A public transport link is a link traversed in a route using public transport.

Figure : PublicTransportLinkType

In addition to its base type RouteLinkType, the PublicTransportLinkType defines the following attributes:

- NextStopName Name of the stop at the end of the link.

- Line Reference key to the PublicTransportLine object. To reduce data volume, the PublicTransport element is not directly embedded in the ManeuverType object, but is swapped out into the Route element.

6.1.12 PublicTransportLineType

Defines the information available for a public transport line.

Figure : PublicTransportLineType

PublicTransportLineType defines the following attributes:

- LineName Name of the line
- LineForeground Color that is to be used as foreground color when drawing the line.
- LineBackground Color that is to be used as background color when drawing the line.
- LineStyle Style that is to be used when drawing the line.
- CompanyName Name of the transit line's company
- CompanyShortName Short name of the transit line's company
- CompanyLogo Logo of the transit line's company
- Destination Final destination of the transport line
- Flags Additional attributes classifying the transport line
- Type Type of the transport line
- TypeName Name of the transport line's type (e.g. "ICE", "TGV", etc.).

6.1.13 WaypointType

A waypoint can represent both a position exactly specified by LinkId as well as the result of a map matching.

In the first case the mappedPosition attribute will not be filled.

Figure : WaypointType

WaypointType defines the following attributes:

- LinkId Id of the link which covers the waypoint.
- MappedPosition The nearest point on the link to the original position.
- OriginalPosition Original position as it was specified in the corresponding request.

6.1.14 RouteNoteType

Route notes are used to store additional information about the route. These notes can either be related to the calculation itself (like violated routing options) or to the characteristics of the route (like entering a toll road, passing a border, etc.).

Figure : RouteNoteType

RouteNoteType defines the following attributes:

- Type Type of the note. Please refer to the enumeration type RouteNoteTypeType for available values.
- Code A code that uniquely identifies the note. This code can be used to decide how to display the note (e.g. with a warning icon). Please refer to the enumeration type RouteNoteCode for available values.
- Text A short text describing the note. Please note that this attribute is not subject to internationalization and should therefore not be used in user displays.
- Position Position at which the note is relevant.
- LinkIds Location at which the note is relevant given by a list of linkIds.
- ValidityPeriod Validity period of the note. This attribute is used to specify time dependent restrictions (e.g. "road is closed during winter").
- AdditionalData Container for additional data to be stored along with the note. (currently not used)

6.2 Parameters and Switches

6.2.1 RouteRepresentationOptionsType

In many use cases, not all objects and attributes in the route's object model are required at once. The enumeration type RouteRepresentationMode has been introduced to conveniently define which part of the route shall be returned by services for standard use cases. Any custom route representation definition besides these predefined modes is possible by using the detailed AttributeType switches (see RouteAttribute, RouteLegAttribute, RouteAttribute, ManeuverAttribute and RouteLinkAttribute).

Figure : RouteRepresentationOptionsType

RouteRepresentationOptionsType defines the following attributes:

- Language The language to be used for all textual information
- RepresentationMode The representation mode is a convenience parameter to define the data representation of the route for common use cases (see RouteRepresentationMode for supported values)
- RouteAttributes Sequence of attribute keys of the fields that are included in the routes (see RouteAttribute for supported attributes)
  If not specified, defaults to "waypoints", "summary", "legs".
- LegAttributes Sequence of attribute keys of the fields that are included in the route legs (see RouteLegAttribute for a list of supported attributes)
  If not specified, defaults to "maneuvers", "waypoint", "length", "travelTime".
- ManeuverAttributes Sequence of attribute keys of the fields that are included in the maneuvers (see ManeuverAttribute for a list of supported attributes)
  If not specified, defaults to "position", "length", "travelTime".
- LinkAttributesSequence of attribute keys of the fields that are included in the links (see RouteLinkAttribute for a list of supported attributes)
  If not specified, defaults to "shape", "speedLimit", "dynamicSpeedInfo", "street"

- CorridorDepthThe corridor depth defines the depth of the link network that will be spanned around the base route. A depth of 0 will only return the base route; a depth of 1 will additionally return the routes to the destination allowing one wrong maneuver drifting away from the optimal base route
  Note: not yet supported
- ViewBounds If the view bounds are given in the request only shapes and links which fit into these bounds will be returned. A common use case for this is the drag and drop scenario where the client is only interested in a rough visual update of the route in the currently visible bounds.
- ViewResolution Resolution of the view in meters per pixel (mpp). This information allows the route shape in the response to reflect the client's resolution.

InstructionFormat Defines the representation format of the maneuver's instruction text. (see InstructionFormatType)

Query Parameter Representation

RepresentationOptions can be specified as follows:

Element	Type	Representation
Language	Language Code Type (see LBSP - Common)	**"&language="** + <Language>
RepresentationMode	RouteRepresentationModeType	**"&representation="** + <RepresentationMode>
RouteAttribute []	RouteAttributeType	**"&routeattributes="** + <RouteAttribute[]>
LegAttribute []	RouteLegAttributeType	**"&legattributes="** + <RouteLegAttribute[]>

ManeuverAttribute []	ManeuverAttributeType	**"&maneuverattributes="** + <ManeuverAttribute[]>
LinkAttribute []	RouteLinkAttributeType	**"&linkattributes="** + <LinkAttribute[]>
CorridorDepth	xs:int	**"&corridordepth="** + <CorridorDepth>
ViewBounds	BoundingBoxType	**"&viewbounds="** + <ViewBounds>
ViewResolution	xs:int	**"&resolution="** + <ViewResolution>
InstructionFormat	InstructionFormatType	**"&instructionformat="** + <InstructionFormat>

6.2.2 WaypointParameterType

Waypoints of a route can be specified in four different manners, either by specifying a rough position (**GeoWaypointParameterType**), an exact reference to a link (**NavigationWaypointParameter** with **LinkPositions**), and a reference to a street (**NavigationWaypointParameter** with **StreetPositions**) or a positioning history (**PositioningWaypointParameter**).

The routing service tries to find the best matching link to be used based on the routing network. Each waypoint may be specified in either mode.

GeoWaypointParameterType

Figure : GeoWaypointParameterType

GeoWaypointParameterType defines the following attributes:

- Position Coordinate as center for the approximation

- TransitRadius Matching Links are selected within the specified TransitRadius.

NavigationWaypointParameterType

The NavigationWaypointParameter is used to define a waypoint based on navigational information like link positions. A common use case for this scenario is when the user specifies a waypoint by selecting a place or a location after having executed a search.

Figure : NavigationWaypointParameterType

NavigationWaypointParameterType defines the following attributes:

- DisplayPosition be displayed on a map. It usually denotes the center of the location and is not navigable, i.e. it is not located on a link in the routing network in contrast to the navigation positions of a location. The display position allows the routing engine to decide whether the waypoint is located on the left or on the right-hand side of the route.

Either StreetPositions or LinkPositions can be specified:

- StreetPosition A StreetPosition is used to define a navigation position based on a coordinate and a streetname. The streetname helps choosing the right road in complex intersection scenarios (e.g. a bridge crossing another road).
  - Position Coordinate identifying the street position
  - StreetName Name of the street the street position is located. This name allows you to distinguish both streets crossing in one coordinate, potentially separated by a bridge/tunnel. The streetname might contain the road name or the road number (if no road name is available).
- **LinkPosition** The LinkPosition is used to define an accurate position on a link.
  - LinkId LinkId of the link position
  - Spot Relative position of the location along the link. Spot is defined as the fractional distance from the link's reference-node to the non-reference node, i.e. the value range is between 0 and 1. This attribute is only relevant if a link is referenced.
  - SideOfStreet Indicates whether the waypoint is on the left or right side of the link (if heading from the reference node to the non-reference node).

PositioningWaypointParameterType

Note: 7 Note: not yet supported

The PositioningWaypointParameter is used to define a waypoint based on the data provided by a GPS device. A list of measurements is passed as input to a map matching algorithm.

Figure : PositioningWaypointParameterType

PositioningWaypointParameterType defines the following attributes:

- PositionHistory List of position measurements provided by a GPS device. (see **GeoPositionLogType**)

Query Parameter Representation

Since the WaypointParameterType is used as atomic input value for a waypoint, a compact representation is required.

The query parameter representation for WaypointParameterType depends on which sub type has been selected.

Element	Type	Representation
Any element of type	GeoWaypointParameterType	**"geo!"**? + <Position> (+ **";"** +<Radius>)?
Any element of type	NavigationWaypointParameterType	**using link positions:** "link!" + <DisplayPosition>? + "!" + <LinkId> (+ "," + <Spot> (+ "," + <SideOfStreet>)?)? **using street positions:** **"street!"** + <DisplayPosition>? + **"!"** + <Position> (+ **";"** + <StreetName>)? (+ **"!"** + <Position> (+ **";"** + <StreetName>)?)*
Any element of type	PositioningWaypointParameterType	**(currently not supported)** **"log!"** + {geopositionlog} (+ **"!"** + {geopositionlog})* {geopositionlog}= <Position> + **";"** + <Timestamp> (+ **";"** + <Heading> (+ **";"** + <Speed> (+ **";"** +

		<Accuracy> (+ “;” + <AltitudeAccuracy> (+ “;” + <DistanceToPrevious >)?)?)?)?)?)?

Below are some examples how to specify waypoints in query parameters:

// using a coordinate:
…&waypoint0=geo!37.7914050,-122.3987030
// using a coordinate omitting the “geo!” prefix (“geo” is the default mode):
…&waypoint0=37.7914050,-122.3987030
// using a coordinate plus transit radius:
…&waypoint0=geo!37.7914050,-122.3987030; 500

// using exact link information:
…&waypoint0=link!37.7914050,-122.3987030!743460791,0.8996,left
…&waypoint1=link!!743460795,0.1342,right

// using the street position
…&waypoint0=street!37.7914050,-122.3987030!37.7914055,-122.3987033;ABC Street
!37.7914053,-122.3987028;DEF Avenue
// the display position can be omitted:
…&waypoint1=street!!37.7914055,-122.3987033;ABC Street!37.7914053,-122.3987028;DEF Avenue

// using the GPS log (currently not supported):
…&waypoint0=log!37.7914050,-122.3987030;2010-07-02T17:00:00Z;220.767;15.3;5.4443;27.34
!37.8914250,-122.3977122;2010-07-02T17:00:15Z,220.767,15.9,5.433,27.34,10
!37.9223257,-122.3988078;2010-07-02T17:00:27Z,220.767,15.3,5.4443,27.34,10

6.2.3 RoutingModeType

The routing mode encapsulates the parameters for one route calculation.

Figure : RoutingModeType

RoutingModeType defines the following attributes:

- Type Routing type relevant for this calculation, e.g. "DirectDrive", "FastestNow", "Shortest" (see RoutingType for supported values)
- TransportModes Transport mode relevant for this calculation, e.g. "Car", "Truck", "Pedestrian" (see TransportMode for supported values)
- TrafficMode Traffic mode relevant for this calculation, e.g. "enabled", "disabled", "default" (see TrafficMode for supported values)
- Feature Route feature weightings to be applied when calculating the route. (WeightedRouteFeatureType derives from RouteFeatureType)
  - **@weight** Weight to be applied for the selected RouteFeatureType

[value] RouteFeature value

Query Parameter Representation

The RoutingModeType can be represented in query strings using a compact representation. It consists of sub elements, which are concatenated like this:

<Type> + “;” + <TransportMode> (+ ”,” + transportMode)*
(+ “;traffic:” + trafficMode)?
(
+ “;” + <routeFeature> + “:” + <routeWeight>
(+ “,” + <routeFeature> + “:” + <routeWeight>)*
)?

6.2.4 IncidentTypeType

Defines identifiers for different incidents

The following enumeration values are available for IncidentTypeType:

- accident
- congestion
- disabledVehicle
- roadHazard
- construction
- plannedEvent
- massTransit
- otherNews
- weather
- miscellaneous

Query Parameter Representation

IncidentTypeType values are expected to be specified verbatim as defined above.

Allowed values: "accident", "congestion", "disabledVehicle", "roadHazard", "construction", "plannedEvent", "massTransit", "otherNews", "weather", "miscellaneous"

6.2.5 PlaceEquipmentType

Defines identifiers for equipment available at a place.

The following enumeration values are available for PlaceEquipmentType:

- stairs
- elevator
- escalator
- barrierFree
- shelter
- restroom

- shopping
- restaurant
- baggageRoom
- bikeDepot
- bikeAccepted
- parking

Query Parameter Representation

PlaceEquipmentType values are expected to be specified verbatim as defined above.

Allowed values: "stairs", "elevator", "escalator", "barrierFree", "shelter", "restroom", "shopping", "restaurant", "baggageRoom", "bikeDepot", "bikeAccepted", "parking"

6.2.6 RouteLinkFlagType

Defines a list of special characteristics that may apply to a link

The following enumeration values are available for RouteLinkFlagType:

- tollroad Link is part of a toll road
- motorway Link is part of a motorway
- boatFerry Link can only be traversed by using a boat ferry
- railFerry Link can only be traversed by using a rail ferry
- publicTransport Link is part of a public transport connection
- tunnel Link passes through a tunnel
- dirtRoad Link is part of a dirt road
- park Link is part of a park
- HOVLane Link can only be traversed by using high-occupancy vehicle (HOV) lanes
- stairs Link can only be traversed by using stairs
- railFerry Link can only be traversed by using a rail ferry
- boatFerry Link can only be traversed by using a boat ferry
- gatedArea Link is part of a gated area
- privateRoad Link is part of a private road
- station Link is located at a station
- unpaved Link is part of an unpaved road
- fourWheelDrive Link is part of a road that is suitable only for vehicles with four-wheel drive
- noThroughRoad Link is part of a road that you can enter but you have to exit the same way
- scenic Link is part of a road that has been marked as scenic

Query Parameter Representation

RouteLinkFlagType values are expected to be specified verbatim as defined above.

Allowed values: "tollroad", "motorway", "boatFerry", "railFerry", "publicTransport", "tunnel", "dirtRoad", "park", "HOVLane", "stairs", "railFerry", "boatFerry", "gatedArea", "privateRoad", "station", "unpaved", "fourWheelDrive", "noThroughRoad", "scenic"

6.2.7 HazardousGoodTypeType

For truck routing, the HazardousGoodsType defines names for different types of hazardous goods.

The following enumeration values are available for HazardousGoodTypeType:

- explosive Explosive material
- gas Gas
- flammable Flammable material
- combustible Combustible material
- organic Organic material
- poison Poison
- radioActive Radio-active material
- corrosive Corrosive material
- poisonousInhalation Goods which are poisonous upon inhalation
- harmfulToWater Goods which are harmful to water
- other Other types of hazardous goods

Query Parameter Representation

HazardousGoodTypeType values are expected to be specified verbatim as defined above.

Allowed values: "all", "explosive", "gas", "flammable", "combustible", "organic", "poison", "radioActive", "corrosive", "poisonousInhalation", "harmfulToWater", "other"

6.2.8 ResourceTypeType

Defines a list of supported resource type identifiers.

The following enumeration values are available for ResourceTypeType:

- junctionView Identifier for the bitmap representing the Junction View.
- signAsReal Identifier for the bitmap representing the sign-as-real information.
- directionArrow Identifier for the bitmap representing the direction arrow at a junction which will be displayed on top of the junction view.
- advertising Identifier for the bitmap representing any kind of advertisement.
- vendorIcon Identifier for the bitmap representing the vendor icon e.g. of a public transport system.
- vendorLogo Identifier for the bitmap representing the vendor logo e.g. of a public transport system.

- routeTemplate Identifier for the svg resource representing a template for displaying routes (e.g. highway signs). Clients will have to include the route number in the graphic to display a concrete route sign.

Query Parameter Representation

ResourceTypeType values are expected to be specified verbatim as defined above.

Allowed values: "junctionView", "signAsReal", "directionArrow", "advertising", "vendorIcon", "vendorLogo", "routeTemplate"

6.2.9 DirectionType

Enumeration type to identify directions at a maneuver.

The following enumeration values are available for DirectionType:

- forward
- bearRight
- lightRight
- right
- hardRight
- uTurnRight
- uTurnLeft
- hardLeft
- left
- lightLeft
- bearLeft

Query Parameter Representation

DirectionType values are expected to be specified verbatim as defined above.

Allowed values: "forward", "bearRight", "lightRight", "right", "hardRight", "uTurnRight", "uTurnLeft", "hardLeft", "left", "lightLeft", "bearLeft"

6.2.10 PrivateTransportActionType

The PrivateTransportAction type lists available actions which can be taken in a PrivateTransportManeuver.

The following enumeration values are available for PrivateTransportActionType:

- depart Identifier for a departure maneuver. Example text: "Start at"
- departAirport Identifier for a departure at an airport maneuver. Example text: "Start toward the airport exit"
- arrive Identifier for an arrival maneuver. Example text: "Arrive at"
- arriveAirport Identifier for an arrival at the airport maneuver. Example text: "Follow the signs to your terminal"

- arriveLeft Identifier for an arrival maneuver with the destination on the left hand side. Example text: "Arrive at"
- arriveRight Identifier for an arrival maneuver with the destination on the right hand side. Example text: "Arrive at"
- leftLoop Identifier for a left-hand loop maneuver. Example text: "Make a left-hand loop onto"
- leftUTurn Identifier for a left-hand U-turn maneuver. Example text: "Make a U-turn at"
- sharpLeftTurn Identifier for a sharp left turn maneuver. Example text: "Make a hard left turn onto"
- leftTurn Identifier for a left turn maneuver. Example text: "Turn left on"
- slightLeftTurn Identifier for a slight left turn maneuver. Example text: "Bear left onto"
- continue Identifier for a continue maneuver. Example text: "Continue on"
- slightRightTurn Identifier for a slight right turn maneuver. Example text: "Bear right onto"
- rightTurn Identifier for a right turn maneuver. Example text: "Turn right on"
- sharpRightTurn Identifier for a sharp right turn maneuver. Example text: "Make a hard right turn onto"
- rightUTurn Identifier for a right u-turn maneuver. Example text: "Make a right U-turn at"
- rightLoop Identifier for a right loop maneuver. Example text: "Make a right-hand loop onto"
- leftExit Identifier for a left exit maneuver. Example text: "Take the left exit to"
- rightExit Identifier for a right exit maneuver. Example text: "Take the right exit to"
- leftRamp Identifier for a left ramp maneuver. Example text: "Take the left ramp onto"
- rightRamp Identifier for a right ramp maneuver. Example text: "Take the right ramp onto"
- leftFork Identifier for a left fork maneuver. Example text: "Take the left fork onto"
- middleFork Identifier for a middle fork maneuver. Example text: "Take the middle fork onto"
- rightFork Identifier for a right fork maneuver. Example text: "Take the right fork onto"
- leftMerge Identifier for a left merge maneuver. Example text: "Merge left onto"
- rightMerge Identifier for a right merge maneuver. Example text: "Merge right onto"
- nameChange Identifier for a name change maneuver (no maneuver action needed). Example text: "Road becomes"
- trafficCircle Identifier for a traffic circle maneuver. Example text: "At the traffic circle take the exit to"
- ferry Identifier for a ferry maneuver. Example text: "Take the ferry to"
- leftRoundaboutExit1 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the first exit of the roundabout onto"

- leftRoundaboutExit2 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the second exit of the roundabout onto"
- leftRoundaboutExit3 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the third exit of the roundabout onto"
- leftRoundaboutExit4 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the fourth exit of the roundabout onto"
- leftRoundaboutExit5 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the fifth exit of the roundabout onto"
- leftRoundaboutExit6 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the sixth exit of the roundabout onto"
- leftRoundaboutExit7 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the 7th exit of the roundabout onto"
- leftRoundaboutExit8 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the 8th exit of the roundabout onto"
- leftRoundaboutExit9 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the 9th exit of the roundabout onto"
- leftRoundaboutExit10 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the 10th exit of the roundabout onto"
- leftRoundaboutExit11Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the 11th exit of the roundabout onto"
- leftRoundaboutExit12 Identifier for a roundabout maneuver (left-hand traffic) Example text: "Take the 12th exit of the roundabout onto"
- rightRoundaboutExit1 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the first exit of the roundabout onto"
- rightRoundaboutExit2 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the second exit of the roundabout onto"
- rightRoundaboutExit3 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the third exit of the roundabout onto"
- rightRoundaboutExit4 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the fourth exit of the roundabout onto"
- rightRoundaboutExit5 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the fifth exit of the roundabout onto"
- rightRoundaboutExit6 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the sixth exit of the roundabout onto"
- rightRoundaboutExit7 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the 7th exit of the roundabout onto"
- rightRoundaboutExit8 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the 8th exit of the roundabout onto"
- rightRoundaboutExit9 Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the 9th exit of the roundabout onto"
- rightRoundaboutExit10
  Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the 10th exit of the roundabout onto"
- rightRoundaboutExit11
  Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the 11th exit of the roundabout onto"

- rightRoundaboutExit12
  Identifier for a roundabout maneuver (right-hand traffic) Example text: "Take the 12th exit of the roundabout onto"

Query Parameter Representation

PrivateTransportActionType values are expected to be specified verbatim as defined above.

Allowed values: "depart", "departAirport", "arrive", "arriveAirport", "arriveLeft", "arriveRight", "leftLoop", "leftUTurn", "sharpLeftTurn", "leftTurn", "slightLeftTurn", "continue", "slightRightTurn", "rightTurn", "sharpRightTurn", "rightUTurn", "rightLoop", "leftExit", "rightExit", "leftRamp", "rightRamp", "leftFork", "middleFork", "rightFork", "leftMerge", "rightMerge", "nameChange", "trafficCircle", "ferry", "leftRoundaboutExit1", "leftRoundaboutExit2", "leftRoundaboutExit3", "leftRoundaboutExit4", "leftRoundaboutExit5", "leftRoundaboutExit6", "leftRoundaboutExit7", "leftRoundaboutExit8", "leftRoundaboutExit9", "leftRoundaboutExit10", "leftRoundaboutExit11", "leftRoundaboutExit12", "rightRoundaboutExit1", "rightRoundaboutExit2", "rightRoundaboutExit3", "rightRoundaboutExit4", "rightRoundaboutExit5", "rightRoundaboutExit6", "rightRoundaboutExit7", "rightRoundaboutExit8", "rightRoundaboutExit9", "rightRoundaboutExit10", "rightRoundaboutExit11", "rightRoundaboutExit12"

6.2.11 RoutingTypeType

The RoutingType provides identifiers for different optimizations which can be applied during the route calculation. Selecting the routing type will internally affect which constraints, speed attributes and weights are taken into account during route calculation.

The following enumeration values are available for RoutingTypeType:

- fastest Route calculation from start to destination optimizing based on the travel time. If the travel time forecast is considering current traffic information depends on the provided traffic mode.
- shortest Route calculation from start to destination disregarding any traffic conditions. In this mode, the distance of the route is minimized.
- economic Route calculation that optimizes the route in consideration of economic aspects ("green routing").
- scenic Route calculation that favors scenic routes and landscapes.
- fastestNow Route calculation from start to destination considering the actual traffic-conditions. Depending on the credentials of the requestor (Basic/Advanced API), the Fastest-Now option will automatically select appropriate constraints to take the actual traffic conditions into account (real-time, patterns, predictive).
  This value is a convenience value that is equivalent to routing type "fastest" combined with traffic mode "enabled".
- directDrive Route calculation from start to destination disregarding any traffic conditions. For requestors with Basic-API credentials, only the NAVTEQ Speed Category and road closures are applied, for Advanced-API requestors, additional (wherever available) free-flow information is applied.

This value is a convenience value that is equivalent to routing type "fastest" combined with traffic mode "disabled".

Query Parameter Representation

RoutingTypeType values are expected to be specified verbatim as defined above.

Allowed values: "fastest", "shortest", "economic", "scenic", "fastestNow", "directDrive"

6.2.12 TrafficModeType

The TrafficMode defines if traffic information shall be considered when calculating the route.

The following enumeration values are available for TrafficModeType:

- enabled Depending on the credentials of the requestor (Basic/Advanced API), the enabled option will automatically select appropriate constraints to take the actual traffic conditions into account (real-time, patterns, predictive).
- disabled For requestors with Basic-API credentials, only the NAVTEQ Speed Category and road closures are applied, for Advanced-API requestors, additional (wherever available) free-flow information is applied.
- default The default mode will consider all traffic information that is applicable for the selected TransportMode and the user is authorized to use.

Query Parameter Representation

TrafficModeType values are expected to be specified verbatim as defined above.

Allowed values: "enabled", "disabled", "default"

6.2.13 RouteFeatureType

The routing features can be used to define special conditions on the calculated route. The user can weight each feature with positive or negative weights, see type RouteFeatureWeight.

The following enumeration values are available for RouteFeatureType:

- tollroad Identifier for toll roads
- motorway Identifier for motorways
- boatFerry Identifier for boat ferries
- railFerry Identifier for rail ferries
- publicTransport Identifier for public transport
- tunnel Identifier for tunnels
- dirtRoad Identifier for dirt roads
- park Identifier for links through parks
- HOVLane Identifier for high-occupancy vehicle (HOV) lanes

- stairs Identifier for stairs. This route feature is only applicable for pedestrian routing.

Query Parameter Representation

RouteFeatureType values are expected to be specified verbatim as defined above.

Allowed values: "tollroad", "motorway", "boatFerry", "railFerry", "publicTransport", "tunnel", "dirtRoad", "park", "HOVLane", "stairs"

6.2.14 RouteFeatureWeightType

Route feature weights are used to define weighted conditions on special route features like "tollroad", "motorways", etc. (see type RouteFeature for a complete list) that the calculated route is supposed to fulfill.

The following enumeration values are available for RouteFeatureWeightType:

- -3 strictExclude: The routing algorithm guarantees that strictly excluded features won't be part of the resulting route. If this condition cannot be fulfilled no route will be returned.
  "HOVLanes" and "stairs" are examples for features where a strict exclusion might be required.
- -2 softExclude: The routing engine will not consider links with the corresponding feature. If no route can be found because of these limitations the condition will be weakened.
- -1 avoid: The routing engine will assign penalties for links with the corresponding feature.
- 0 normal: The routing engine will neither prefer nor avoid or exclude links with the corresponding feature.
- 1 prefer: The routing engine will prefer links with the corresponding route feature.

Query Parameter Representation

RouteFeatureWeightType values are expected to be specified verbatim as defined above.

Allowed values: "-3", "-2", "-1", "0", "1"

6.2.15 TransportModeType

The TransportMode provides identifiers for differentiating the transport mode. Depending on the transport mode special constraints, speed attributes and weights are taken into account during route calculation.

The following enumeration values are available for TransportModeType:

- car Route calculation for cars.
- pedestrian Route calculation for a pedestrian. As one effect, maneuvers will be optimized for walking, i.e. segments will consider actions relevant for pedestrians and

maneuver instructions will contain texts suitable for a walking person. This mode disregards any traffic information.

- publicTransport Route calculation using public transport only.
- truck Route calculation for trucks. This mode will consider truck limitations on links and will use different speed assumptions when calculating the route.
- bicycle Route calculation for bicycles.
  Note: Not yet supported.

Query Parameter Representation

TransportModeType values are expected to be specified verbatim as defined above.

Allowed values: "car", "pedestrian", "publicTransport", "truck", "bicycle"

6.2.16 PublicTransportActionType

Enumeration to identify different action types needed for public transport maneuvers.

The following enumeration values are available for PublicTransportActionType:

- enter
- leave

Query Parameter Representation

PublicTransportActionType values are expected to be specified verbatim as defined above.

Allowed values: "enter", "leave"

6.2.17 PublicTransportLinkFlagType

Identifiers for additional attributes classifying a public transport link

The following enumeration values are available for PublicTransportLinkFlagType:

- barrierFree
- bicycleAllowed
- lowFloorVehicle

Query Parameter Representation

PublicTransportLinkFlagType values are expected to be specified verbatim as defined above.

Allowed values: "barrierFree", "bicycleAllowed", "lowFloorVehicle"

6.2.18 PublicTransportTypeType

Enumeration type to identify different public transport types.

The following enumeration values are available for PublicTransportTypeType:

- busPublic
- busTouristic
- busIntercity
- busExpress
- railMetro
- railLight
- railRegional
- trainRegional
- trainIntercity
- trainHighSpeed
- monoRail
- aerial
- inclined
- water

Query Parameter Representation

PublicTransportTypeType values are expected to be specified verbatim as defined above.

Allowed values: "busPublic", "busTouristic", "busIntercity", "busExpress", "railMetro", "railLight", "railRegional", "trainRegional", "trainIntercity", "trainHighSpeed", "monoRail", "aerial", "inclined", "water"

6.2.19 LineStyleType

Linestyle used to draw a line

The following enumeration values are available for LineStyleType:

- solid
- dotted
- dash

Query Parameter Representation

LineStyleType values are expected to be specified verbatim as defined above.

Allowed values: "solid", "dotted", "dash"

6.2.20 RouteRepresentationModeType

In many use cases, not all objects and attributes in the route's object model are required at once. The enumeration type RouteRepresentationMode has been introduced to conveniently define which part of the route shall be returned by services for standard use cases. Any custom route representation definition besides these predefined modes is possible by using the detailed AttributeType switches (see RouteResponseAttribute, RouteAttribute, ManeuverAttribute and LinkAttribute).

The following enumeration values are available for RouteRepresentationModeType:

- overview Overview mode only returning the Route and the RouteSummary object
- displayDisplay mode that allows to show the route with all maneuvers. Links won't be included in the response
- dragNDrop Drag and Drop mode to be used during drag and drop (re-routing) actions. The response will only contain the shape of the route restricted to the view bounds provided in the representation options.
- navigation Navigation mode to provide all information necessary to support a navigation device. This mode activates the most extensive data response as all link information will be included in the response to allow a detailed display while navigating.
- linkPaging Paging mode that will be used when incrementally loading links while navigating along the route. The response will be limited to link information.

Query Parameter Representation

RouteRepresentationModeType values are expected to be specified verbatim as defined above.

Allowed values: "overview", "display", "dragNDrop", "navigation", "linkPaging"

6.2.21 RouteResponseAttributeType

In many use cases, not the entire route object model is required at once. To be able to specify which part of the route shall be returned by services, the enumeration type RouteResponseAttribute defines switches for including or excluding child objects of the route.

The following enumeration values are available for RouteResponseAttributeType:

- maneuvers Indicates whether maneuvers should be provided in the response.
- links Indicates whether route links should be provided in the response.

Query Parameter Representation

RouteResponseAttributeType values are expected to be specified verbatim as defined above.

Allowed values: "maneuvers", "links"

6.2.22 RouteAttributeType

In many use cases, not all route attributes are required at once. To be able to specify which part of the route shall be returned by services, the enumeration type RouteAttributes defines switches for route attribute inclusion and exclusion.

The following enumeration values are available for RouteAttributeType:

- waypoints Short value: "wp". Indicates whether via points shall be included in the route.
- summary Short value: "sm". Indicates whether a route summary shall be provided for the route.
- summaryByCountry Short value: "sc". Indicates whether a country-based route summary shall be provided for the route.
- shape Short value: "sh". Indicates whether the shape of the route shall be provided for the route.
- boundingBox Short value: "bb". Indicates whether the bounding box of the route shall be provided for the route.
- legs Short value: "lg". Indicates whether the legs of the route shall be provided for the route.
- notes Short value: "no". Indicates whether additional notes shall be provided for the route.

Query Parameter Representation

RouteAttributeType values are expected to be specified verbatim as defined above.

Allowed values: "waypoints", "summary", "summaryByCountry", "shape", "boundingBox", "legs", "notes"

6.2.23 RouteLegAttributeType

In many use cases, not all route leg attributes are required at once. To be able to specify which part of the route shall be returned by services, the enumeration type RouteLegAttribute defines switches for route leg attribute inclusion and exclusion.

The following enumeration values are available for RouteLegAttributeType:

- waypoint Short value: "wp". Indicates whether the waypoint shall be included in the route leg.
- maneuvers Short value: "mn". Indicates whether the maneuvers of the route leg shall be provided.

- links Short value: "li". Indicates whether the links along the route leg shall be provided.
- length Short value: "le". Indicates whether the route leg should include its length
- travelTime Short value: "tt". Indicates whether the route leg should include its duration

Query Parameter Representation

RouteLegAttributeType values are expected to be specified verbatim as defined above.

Allowed values: "waypoint", "maneuvers", "links", "length", "travelTime"

6.2.24 ManeuverAttributeType

In many use cases, not all maneuver attributes are required at once. To be able to specify which part of the maneuvers shall be returned by services, the enumeration type ManeuverAttributes defines switches for maneuver attribute inclusion and exclusion.

The following enumeration values are available for ManeuverAttributeType:

- position Short value: "po". Indicates whether the position should be included in the maneuvers.
- shape Short value: "sh". Indicates whether the shape of the segment to the next maneuver should be included in the maneuvers.
- travelTime Short value: "tt". Indicates whether the time needed to the next maneuver should be included in the maneuvers.
- length Short value: "le". Indicates whether the distance to the next maneuver should be included in the maneuvers.
- time Short value: "ti". Indicates whether the point in time when the maneuver will take place should be included in the maneuvers.
- link Short value: "li". Indicates whether the link where the maneuver takes place shall be included in the maneuver.
- publicTransportLine Short value: "pt". Indicates whether the information about the public transport line should be included in the maneuvers.
- platform Short value: "pl". Indicates whether the platform information for a public transport line should be included in the maneuvers.
- equipment Short value: "eq". Indicates whether equipment at the maneuver should be included in the maneuvers.
- lane Short value: "la". Indicates whether lane information should be included in the maneuvers.
- roadName Short value: "rn". Indicates whether the road name should be included in the maneuvers.
- nextRoadName Short value: "nr". Indicates whether the name of the next road shall be included in the maneuvers.
- roadNumber Short value: "ru". Indicates whether the road number should be included in the maneuvers.

- nextRoadNumber Short value: "nu". Indicates whether the number of the next road should be included in the maneuvers.
- roadTemplate Short value: "rt". Indicates whether the template for route display should be included in the maneuvers.
- signPost Short value: "sp". Indicates whether the sign post information should be included in the maneuvers.
- notes Short value: "no". Indicates whether additional notes should be included in the maneuvers.
- action Short value: "action". Indicates whether actions should be included in the maneuvers.
- direction Short value: "di". Indicates whether directions should be included in the maneuvers.

Query Parameter Representation

ManeuverAttributeType values are expected to be specified verbatim as defined above.

Allowed values: "position", "shape", "travelTime", "length", "time", "link", "publicTransportLine", "platform", "equipment", "lane", "roadName", "nextRoadName", "roadNumber", "nextRoadNumber", "roadTemplate", "signPost", "notes", "action", "direction"

6.2.25 RouteLinkAttributeType

In many use cases, not all link attributes are required at once. To be able to specify which part of the link shall be returned by services, the enumeration type RouteLinkAttributes defines switches for maneuver attribute inclusion and exclusion.

The following enumeration values are available for RouteLinkAttributeType:

- shape Short value: "sh". Indicates whether the link should include its geometry
- length Short value: "le". Indicates whether the link should include its length
- speedLimit Short value: "sl". Indicates whether the link should include SpeedLimit
- dynamicSpeedInfo Short value: "ds". Indicates whether the link should include dynamic speed information
- incidents Short value: "ic". Indicates whether the link should include incidents
- truckRestrictions Short value: "tr". Indicates whether the link should include truck restrictions
- externalResources Short value: "er". Indicates whether the link should include external resources
- flags Short value: "fl". Indicates whether the link should include link flags
- address Short value: "ad". Indicates whether the link should include the link's address
- roadNumber Short value: "rn". Indicates whether the link should include the link's road number
- freewayExit Short value: "fe". Indicates whether the link should include the name of the freeway exit

- freewayJunction Short value: "fj". Indicates whether the link should include the name of the freeway junction
- timezone Short value: "tz". Indicates whether the link should include the timezone
- corridorLevel Short value: "cl". Indicates whether the link should include the corridor level
- nextLink Short value: "nl". Indicates whether the link should include the link which will be next when following the route
- stubs Short value: "st". Indicates whether the link should include the corridor stubs
- publicTransportLine Short value: "pt". Indicates whether the link should include information about the public transport line.
- TMCCodes Short value: "tm". Indicates whether the link should include information about the covered TMC Codes.
- jamFactor Short value: "jf". Indicates whether the link's dynamic speed info should include information about the jam factor.
- jamFactorTrend Short value: "jt". Indicates whether the link's dynamic speed info should include information about the jam factor trend.
- confidence Short value: "co". Indicates whether the link's dynamic speed info should include information about the level of confidence wrt. traffic information.
- remainTime Short value: "rt". Indicates whether the link should include information about the remaining time until the destination is reached.
- remainDistance Short value: "rd". Indicates whether the link should include information about the remaining distance until the destination is reached.

Query Parameter Representation

RouteLinkAttributeType values are expected to be specified verbatim as defined above.

Allowed values: "shape", "length", "speedLimit", "dynamicSpeedInfo", "incidents", "truckRestrictions", "externalResources", "flags", "address", "roadNumber", "freewayExit", "freewayJunction", "timezone", "corridorLevel", "nextLink", "stubs", "publicTransportLine", "TMCCodes", "jamFactor", "jamFactorTrend", "confidence", "remainTime", "remainDistance"

6.2.26 RouteNoteCodeType

Defines identifiers for **RoutingNote** objects.

The following enumeration values are available for RouteNoteCodeType:

- routingOptionViolated Indicates that routing options have been violated, i.e. the route contains route features that should have been avoided. The violated routing feature will be provided in the additional data container of the corresponding note.
- passingPlace Indicates that a special place (city, country border, POI, etc.) will be passed in the segment following the maneuver. The name and type of the place will be provided in the additional data container of the corresponding note.
- roadNameChanged Indicates that road name and/or route number will change at the given position without an additional maneuver taking place. The new street name

and/or route number will be provided in the additional data container of the corresponding note.

- sharpCurveAhead Indicates that a sharp curve is ahead of the maneuver's position.
- linkFeatureAhead Indicates that a special link characteristic will be met after the maneuver. The identifier for the link characteristic (see enumeration type LinkFlag) will be provided in the additional data container of the corresponding note.
- timeDependentRestriction
  Indicates time dependent restrictions for the segment following the maneuver, e.g. "road closed in winter". The validity period of this restriction will be provided in the corresponding note.

6.2.27 InstructionFormatType

Representation formats for instruction texts.

The following enumeration values are available for InstructionFormatType:

- text

html

Query Parameter Representation

InstructionFormatType values are expected to be specified verbatim as defined above.

Allowed values: "text", "html"

7 CalculateRoute Service

This chapter describes data types which are exclusively used by the CalculateRoute service.

7.1 CalculateRouteRequestType

CalculateRouteRequest is the data structure for calling the CalculateRoute service.

Figure : CalculateRouteRequestType

CalculateRouteRequestType defines the following attributes:

- MetaInfo Request parameters which are not specific to route calculation
  - RequestId Clients may pass in arbitrary values to trace request processing through the system. The RequestId will be mirrored in the MetaInfo element of the response structure.
- RepresentationOptions
  In many use cases, not all objects and attributes in the route's object model are required at once. The enumeration type RouteRepresentationMode has been introduced to conveniently define which part of the route shall be returned by services for standard use cases. Any custom route representation definition besides these predefined modes is possible by using the detailed AttributeType switches (see RouteAttribute, RouteLegAttribute, RouteAttribute, ManeuverAttribute and RouteLinkAttribute).
- Waypoint Array of waypoints that should be passed (in the given order) along the route.
  The first element marks the start, the last the end point. Waypoints in between are interpreted as via points.
- AvoidArea Areas which the route must not cross.
- AvoidLinks Links that the route may not cross.

You can either specify Arrival or Departure time in the route request.

- Departure Time when the travel is expected to start. The routing engine will consider time dependent traffic patterns and incidents when calculating the route.
- Arrival Time when the travel is expected to end. The routing engine will consider time dependent traffic patterns and incidents when calculating the route.
  Note: not yet supported
- Alternatives Number of alternative routes that will be returned.
- Mode The routing mode specifies how the route shall be calculated. One route (resp. number of alternatives if set) will be calculated for each requested routing mode.
  - Type Routing type relevant for this calculation, e.g. "DirectDrive", "FastestNow", "Shortest" (see RoutingType for supported values)
  - TransportModes Transport mode relevant for this calculation, e.g. "Car", "Truck", "Pedestrian" (see TransportMode for supported values)

- TrafficMode Traffic mode relevant for this calculation, e.g. "enabled", "disabled", "default" (see TrafficMode for supported values)
- Feature Feature weights to be applied when calculating the route.

- PublicTransportProfile
  Defines settings relevant for public transport routing.
  Note: not yet supported
  - MaxNumberOfChanges Maximal number of transit changes
  - MinDurationForChange Minimal duration that is needed to change from one transport line to the next. This value overrides the default settings applied in the routing engine.
  - AvoidTransportTypes Public transport types that shall not be included in the response route. Please refer to PublicTransportType for a list of supported values.
  -

- TruckProfile Defines settings relevant for truck routing.
  - HasTrailer Defines if the truck has a trailer
  - ShippedHazardousGoods
    Defines the list of hazardous goods that is shipped by the truck.
  - PermittedGrossWeight
    Defines the permitted vehicle gross weight
  - LimitedWeight Defines the vehicle's limited weight
  - WeightPerAxle Defines the vehicle's weight per axle
  - TrailerWeight Defines the weight of the vehicle's trailer
  - Height Defines the height of the truck in meters
  - Width Defines the width of the truck in meters
  - Length Defines the length of the truck in meter

Query Parameter Representation

Routing-related parameters can be specified as follows:

Element	Type	Representation
Waypoint []	Waypoint ParameterType	“&waypoint0=” + <Waypoint[0]> “&waypoint1=” + <Waypoint[1]> … “&waypointn=” + <Waypoint[n]>
AvoidArea []	Bounding BoxType	“avoidareas=” + <AvoidArea[]>
AvoidLinks	List of LinkIdType (see	“avoidlinks=” + <AvoidLinks>

	LBSP-Common)	
Mode []	RoutingModeType	A single mode object can be specified like this: **"&mode="** + {mode} Multiple modes can indexed like this: **"&mode0="** + {mode} + **"&mode1="** + {mode} + … For detailed syntax of {mode} see **RoutingModeType**.
Departure	xs:dateTime	"&departure=" + <Departure>
Arrival	xs:dateTime	"&arrival=" + <Arrival>
Alternatives	xs:int	"&alternatives=" + <Alternatives>
PublicTransportProfile	PublicTransportProfileType	see table below
TruckProfile	TruckProfileType	see table below
Representation Options	RouteRepresentationOptionsType	see table below

PublicTransportProfile parameters can be specified as follows *(Note: not yet supported)*:

Element	Type	Representation

MaxNumberOfChanges	xs:int	“maxnumberofchanges=” + <MaxNumberOfChanges >
MinDurationForChange	DurationType (see LBSP-Common)	“mindurationforchange =” + <MinDurationForChange[]>
AvoidTransportTypes []	PublicTransportTypeType	“&avoidtransporttypes =” + <AvoidTransportTypes[]>

TruckProfile parameters can be specified as follows:

Element	Type	Representation
HasTrailer	xs:boolean	“hastrailer=” + <WeightPerAxle>
ShippedHazardousGoods []	HazardousGoodsTypeType	“shippedhazardousgoods=” + <ShippedHazardousGoods[]>
PermittedGrossWeight	WeightType (see LBSP-Common)	“&permittedgrossweight=” + <PermittedGrossWeight>
LimitedWeight	WeightType (see LBSP-Common)	“&limitedweight=” + <LimitedWeight>
WeightPerAxle	WeightType (see LBSP-Common)	“weightperaxle=” + <WeightPerAxle>

TrailerWeight	WeightType (see LBSP-Common)	“&trailerweight=” + <TrailerWeight>
Height	DistanceType (see LBSP-Common)	“height=” + <Height>
Width	DistanceType (see LBSP-Common)	“width=” + <Width>
Length	DistanceType (see LBSP-Common)	“length=” + <Length>

A complex sample request could look like this:

```
http://.../calculateroute.xml?
        waypoint0=geo!123.45,-34.45;12.7;300
        &waypoint1=link!-3456
        &waypoint2=geo!123.45,-34.45;12.7;300
        &waypoint3=link!-12345
        &avoidareas=37.7890649,-122.4027371,37.7902858,-122.3993039
        &mode=fastest;car
        &departure=2010-04-01T17:00:00Z
        &locale=en-us
        &maneuverattributes=position;link
        &linkattributes=shape;dynamicSpeedInfo
        &requestid=my_own_tracing_id_12345
```

7.2 CalculateRouteResponseType

CalculateRouteResponseType is the data structure for the responses from the CalculateRoute operation. A CalculateRouteResponseType element always corresponds to a request of type CalculateRouteRequestType.

Figure : CalculateRouteResponseType

CalculateRouteResponseType defines the following attributes:

- MetaInfo Meta information which is not directly related to the route calculation is wrapped within a separate element.

- RequestId Mirrored RequestId value from the request structure. Can be used to trace back requests.
- Timestamp Time at which the route calculation was performed.
- NextPageHandle
  The next page handle allows you to identify the next portion that will be loaded in a paging scenario.
  Note: not yet supported
- AdditionalData Generic key/value container to keep additional attributes. The defined key/values are:

“Service”..... [service name] + “, “ + [service version]

“Map0”, “Map1” Information regarding underlying map(s):
[map id] + “, “ + [map version]

“Module0”, **“Module1”**Information regarding the underlying module(s):
[module id] + “, “ + [module version]

- Route List of calculated routes.
  (see **RouteType**)

8 GetRoute Service

This chapter describes data types which are exclusively used by the GetRoute service.

8.1 GetRouteRequestType

GetRouteRequest is the data structure for calling the GetRoute service.

The main input parameter for the GetRoute service is the RouteId that has been returned by a previous route calculation and encodes the entire route.

In a corridor routing scenario additional information besides the RouteId are needed to reconstruct the corridor around the base type.

Figure : GetRouteRequestType

GetRouteRequestType defines the following attributes:

- MetaInfo Request parameters which are not specific to route retrieval.
  - RequestId Arbitrary value which will be mirrored in the response structure. Can be used to trace back requests.
- RepresentationOptions
  The RouteRepresentationOptions defines options to fine-tune the scope and granularity of a route representation in the response.
- RouteId RouteId that codes all information about a previously calculated route needed to reconstruct exactly the same route.
- CurrentPosition If this current position is provided the response will include the information about the travel progress with updated remaining travel times. This value can be one of the sub types of **WaypointParameterType**.
- AvoidArea Areas which the route must not cross
- AvoidLinks Links that the route may not cross.
- Departure Time when the travel is expected to start. The routing engine will consider time dependent traffic patterns and incidents when calculating the route
- ArrivalTime when the travel is expected to end. The routing engine will consider time dependent traffic patterns and incidents when calculating the route
  Note: Not yet supported
- Mode The routing mode specifies how the route shall be calculated. One route (resp. number of alternatives if set) will be calculated for each requested routing mode.
  - Type Routing type relevant for this calculation, e.g. "DirectDrive", "FastestNow", "Shortest" (see RoutingType for supported values)
  - TransportModes Transport mode relevant for this calculation, e.g. "Car", "Truck", "Pedestrian" (see TransportMode for supported values)
  - TrafficMode Traffic mode relevant for this calculation, e.g. "enabled", "disabled", "default" (see TrafficMode for supported values)
  - Feature Feature weights to be applied when calculating the route.

- PublicTransportProfile
  Defines settings relevant for public transport routing.
- TruckProfile Defines settings relevant for truck routing.

Query Parameter Representation

Routing-related parameters can be specified as follows:

Element	Type	Representation
RequestId	xs:string	“&requestid=” + <RequestId[]>
RouteId	xs:string	“&routeid=” + <RouteId>
CurrentPosition	WaypointParameterType	“&pos=” + <CurrentPosition>
AvoidArea []	BoundingBoxType	“&avoidareas=” + <AvoidArea[]>
AvoidLinks	List of LinkIdType (see LBSP-Common)	“&avoidlinks=” + <AvoidLinks>
Mode	RoutingModeType	See **RoutingModeType**
Departure	xs:dateTime	“&departure=” + <Departure>
Arrival	xs:dateTime	“&arrival=” + <Arrival>
Alternatives	xs:int	“&alternatives=” + <Alternatives>
PublicTransportProfile	PublicTransportProfileType	see table below
TruckProfile	TruckProfileType	see table below

RepresentationOptions	RouteRepresentationOptionsType	see table below

PublicTransportProfile parameters can be specified as follows:

Element	Type	Representation
MaxNumberOfChanges	xs:int	“&maxnumberofchanges=” + <MaxNumberOfChanges >
MinDurationForChange	DurationType (see LBSP-Common)	“&mindurationofchange =” + <MinDurationForChange[]>
AvoidTransportTypes []	PublicTransportTypeType	“&avoidtransporttypes =” + <AvoidTransportTypes[]>

TruckProfile parameters can be specified as follows:

Element	Type	Representation
HasTrailer	xs:boolean	“&hastrailer=” + <WeightPerAxle>
ShippedHazardousGoods []	HazardousGoodsTypeType	“shippedhazardousgoods=” + <ShippedHazardousGoods[]>
PermittedGrossWeight	WeightType (see LBSP-Common)	“&permittedgrossweight=” + <PermittedGrossWeight>
LimitedWeight	WeightType (see LBSP-Common)	“&limitedweight=” + <LimitedWeigh

		t>
WeightPerAxle	WeightType (see LBSP-Common)	“weightperaxle =” + <WeightPerAxle>
TrailerWeight	WeightType (see LBSP-Common)	“&trailerweight =” + <TrailerWeight>
Height	DistanceType (see LBSP-Common)	“&height=” + <Height>
Width	DistanceType (see LBSP-Common)	“&width=” + <Width>
Length	DistanceType (see LBSP-Common)	“&length=” + <Length>

A sample request, which specifies all parameters (also optional ones), could look like this:

```
http://.../getroute.xml?
        routeid=ZAK128JSS90JN1729
        &avoidareas=37.7890649,-122.4027371,37.7902858,-122.3993039
                        &mode=fastest;car
        &language=en-us
        &maneuverattributes=position,link
        &linkattributes=shape,DynamicSpeedInfo
        &requestid=my_own_tracing_id_12345
```

8.2 GetRouteResponseType

Response type for the GetRoute service.

Figure : GetRouteResponseType

GetRouteResponseType defines the following attributes:

- MetaInfo Meta information which is not directly related to the route calculation is wrapped within a separate element.
  - RequestId Mirrored RequestId value from the request structure. Can be used to trace back requests.

- Timestamp Time at which the route calculation was performed.
- NextPageHandle
  The next page handle allows you to identify the next portion that will be loaded in a paging scenario.
  Note: not yet supported
- AdditionalData Generic key/value container to keep additional attributes.

- Progress Reflects the current travel progress based on the current position provided in the request.
  - mappedPosition The current position, potentially evaluated after map matching a sequence of positions provided in the request.
  - remainDistance Remaining distance to the destination from the current position
  - remainTime Remaining time needed to reach the destination from the current position considering traffic information
- Route Updated Route as requested by the corresponding GetRouteRequest. In case of an error the association might be empty.

9 GetLinkInfo Service

This chapter describes data types which are exclusively used by the GetLinkInfo service.

9.1 GetLinkInfoRequestType

With a set of Link IDs or a routeId encoding a set of Link IDs, detailed information can be requested using the GetLinkInfoRequest.

Figure : GetLinkInfoRequestType

GetLinkInfoRequestType defines the following attributes:

- MetaInfo Request parameters which are not related to the GetLinkInfo service are specified separately in this element.
  - RequestId Arbitrary value which will be mirrored in the response structure. Can be used to trace back requests.
- LinkIds List of Link IDs for which the detailed link information is requested.
- RouteId The RouteId is an alternative way of defining the links for which detailed information is requested.
- Mode The Transport Mode which the speed calculation should be based on. (see **TransportModeType**)
- ReferenceTime The time for which the dynamic traffic information shall be returned. All link information will refer to this timestamp (which differs from the GetRoute specification where the time will be increased following the route). If the timestamp is not provided the result will be based on the current time.
- RepresentationOptions
  Options to fine-tune the scope and granularity of the output (Note: You will recognize that "RepresentationOptions" is the common name for this type of information in all NAVTEQ LBSP services.)
  - Language The language to be used for all textual information
  - LinkAttribute Sequence of attribute names to be included in the links (see LinkAttribute for a list of available values) If not specified, links will include the attributes Shape, SpeedLimit, DynamicSpeedInfo, Street.
  - ViewBounds If the view bounds are given in the request only links which fit into these bounds will be returned.

Query Parameter Representation

Parameters for the GetLinkInfoRequest can be specified as follows:

Element	Representation
LinkIds	“&linkids=” + <LinkIds>
RouteId	“&routeid=” + <RouteId>

Mode	“&mode=” + <Mode>
ReferenceTime	“&time=” + <ReferenceTime>

Representation options can be specified as follows:

Element	Representation
RepresentationOptions.Language	“&language=” + <Language>
RepresentationOptions.LinkAttribute []	“&linkattributes=” + <LinkAttribute[]>
RepresentationOptions.ViewBounds	“&viewbounds=” + <ViewBounds>

MetaInfo can be specified as follows:

Element	Representation
MetaInfo.RequestId	“&requestid=” + <RequestId>

A sample request, which specifies all parameters (also optional ones), could look like this:

```
http://.../getlinkinfo.xml?
        linkids=+12345,-567890,+134562,+135678
        &mode=truck
        &linkattributes=shape;dynamicSpeedInfo
        &requestid=my_own_tracing_id_12345
```

9.2 GetLinkInfoResponseType

The NAVTEQ Routing Service returns a GetLinkInfoResponse element as answer to a GetLinkInfoRequest.

Figure : GetLinkInfoResponseType

GetLinkInfoResponseType defines the following attributes:

- MetaInfo Meta information which is not directly related to the GetLinkInfo service is wrapped within a separate element.
  - RequestId Mirrored RequestId value from the request structure. Can be used to trace back requests
  - MapVersion Version of the underlying map
  - ModuleVersion Version of the module which performed the route calculation

- InterfaceVersion
  Version of the used schema definition. This information is required since representation formats other than XML do not have a concept like namespaces.
- Timestamp Time at which the request has been processed

- Link Sequence of link details (see LinkInfo) for each link will be included in the response.

10 Errors

If something goes wrong during request processing, the service returns an error. Error details are transferred to the client using a generic error structure, which is returned instead of the actual response structure.

Figure : ServiceErrorType

The error type defines the following attributes:

- @type Errors are categorized into one of the following groups:
  - ApplicationError: Errors that are thrown since the business logic has detected some error
  - SystemError: Errors that are thrown due to technical reasons
  - **PermissionError:** Errors that are thrown to reject incoming requests due to invalid credentials or missing entitlements
- @subtype Defined name of the concrete error, e.g. "InvalidInputData", "ExceededUsageLimit".
- Details Clear text error message
- RequestId Arbitrary value to trace back the request. This value is copied from the request which caused the error.
- AdditionalData Generic container to include structured details regarding the error. Each concrete error sub type defines the semantics of the values in the container.

Errors are returned in the same format (XML, JSON, JSONP) as the expected response.

10.1 Error Types

All defined errors are grouped into one of the three basic error types defined below:

- ApplicationError Errors that are thrown since the business logic has detected some error
- SystemError Errors that are thrown due to technical reasons
- PermissionError Errors that are thrown to reject incoming requests due to invalid credentials or missing entitlements

The error type name is mapped to the ServiceError.type attribute. Services may define their own, more specific errors based on these three error types. The name of the more specific error type is then mapped to the ServiceError.subtype attribute.

However, the basic error types may also be used for concrete errors if there is no sub type that better fits the error situation. In that case, the ServiceError.subtype attribute is omitted.

The usage of the ServiceError.AdditionalData element is not defined for the basic error types. Error sub types may define their own values.

10.2 Error Sub Types

10.2.1 InvalidCredentials

This error is returned if the specified token was invalid or no contract could be found for this token.

The following field mapping applies:

Field	Value
ServiceError.type	“PermissionError”
ServiceError.subtype	“InvalidCredentials”
ServiceError.AdditionalData	Entry 1: Key: “token” Value: token specified in the request

10.2.2 InsufficientRights

This error is returned if the user's contract exists for the given token but does not have the required entitlements.

The following field mapping applies:

Field	Value
ServiceError.type	“PermissionError”
ServiceError.subtype	“InsufficientRights”
ServiceError.AdditionalData	Entry 1: Key: “token” Value: token specified in the request

	Entry 2: Key: “missing_entitlement” Value: entitlement that is missing

10.2.3 ContractViolated

This error is returned if the user's contract exists for the given token but is not yet active or has already expired.

The following field mapping applies:

Field	Value
ServiceError.type	“PermissionError”
ServiceError.subtype	“ContractViolate”

10.2.4 ExceededUsageLimit

This error is returned if the contract associated with the token has reached the usage limit for a certain entitlement.

The following field mapping applies:

Field	Value
ServiceError.type	“PermissionError”
ServiceError.subtype	“ExceededUsageLimit”
ServiceError.AdditionalData	Entry 1: Key: “token” Value: token specified in the request Entry 2: Key: “entitlement” Value: entitlement that has reached usage limit Entry 3: Key: “usage_limit” Value: Usage limit

10.2.5 InvalidInputData

This error is returned if the specified request parameters contain invalid data, e.g. due to wrong parameter syntax or invalid parameter combinations.

The following field mapping applies:

Field	Value
ServiceError.type	“ApplicationError”
ServiceError.subtype	“InvalidInputData”
ServiceError.AdditionalData	One entry for each parameter that caused the error: Key: name of the query parameter Value: value specified for the query parameter

10.3 HTTP Status Codes

For all RESTful requests with XML or JSON format, the above defined errors are mapped to corresponding HTTP status codes:

Error Type	HTTP status code
ApplicationError and sub types	400 Bad Request
PermissionError and sub types	403 Forbidden
SystemError and sub types	500 Internal Server Error

In JSONP, errors are always returned with HTTP status code 200 (OK), since other status codes cannot be handled correctly by JSONP clients and the callback will fail.

11 Security

11.1 Authentication

Authentication for LBSP services is based on an authentication token mechanism. For each contract, Navteq assigns a unique authentication token, which is required to be passed along with other query parameters in every service request:

…/foo.xml?param1=abc&param2=def&token=abfd342fbdfcda1234