Transport for London

TfL Journey Planner API

XML Interface Documentation TRIP_REQUEST2

Transport for London
50 Victoria Street
Westminster
London
SW1H 0TL

www.tfl.gov.uk/developers
digital@tfl.gov.uk

Please pass any queries to digital@tfl.gov.uk with *JP API Feedback* in the subject line

Transport for London

Table of Contents

Document history

Document Version	Software Version	Date	Name	Revision Type
1.0	EFA 9.16	25.07.2011	AK	Initial Version
1.1	EFA 9.16	02.08.2011	AK/Phil Young	Update Coordinates Long/ Lat and general updates
1.2	EFA 9.16.23.4	02.08.2011	Phil Young	Final changes before release to pre-beta development group
1.3	EFA 9.16.23.4	02.12.2011	AK	Update “cycleType”
1.4	EFA 9.16.28.30	16.11.2012	AK	Update London Overground, Section 4.1.1
1.5	EFA 9.16.28.30	10.12.2012	CW	Update Cycle JP, Section 3.1.2 and 5.1, 5.7

Release

	Date	Name	Signature
Reviewed:	02/08/11	Phil Young	
Released:	02/08/11	Phil Young	

1 Introduction

Building on the success over the past 10 years of the Journey Planner which provides customers with journey planning solutions within desktop and mobile browsers, Transport for London is now releasing a API.

This will enable registered application developers to freely access the same journey solutions that are available to customers on the TfL website and mobile site.

The API is intended to enable developers to create solutions on a variety of platforms and increase the reach of reliable travel information in London.

It is not intended as a tool for research or reverse engineering of source data.

The TfL Journey Planner API is controlled via a number of different HTTP requests and parameters.

The answer to each request is given in as XML.

A request is structured as follows:

```
http://server:port/virtualDirectory/Request?HTTP parameters
```

All requests are derived from the request object. As a result some basic functionality, such as session handling or language settings, and the associated HTTP parameters, are automatically available in each request. There are also different groups of HTTP parameters which can be commonly used by different request types. These are described in the section entitled Common Components. The section HTTP-Requests is a description of the request-specific HTTP parameters.

2 Common Components

2.1 *Request Object*

The basis for all requests is the request object. The request object offers HTTP parameters that are especially defined for all requests. However not every HTTP parameter is appropriate or can be used for every request.

2.1.1 HTTP Parameters of the Request Object

language = <language>

This parameter gives the language of the session. As value of <*language*> you can use all of the language codes which are given by the standard ISO 639:1988 (E/F). The language of the interface is also controlled by this parameter. You can find this parameter in the XML output as the attribute *language* of the request elements *itdRequest*.

Example: The value for English is *en*, German *de,* Italian *it* and French *fr*.

Note: If you do not define this parameter the default value will be used, which is given in the configuration file of the Journey Planner Controller as the parameter *DefaultLanguage*. If this parameter is not given by the parameter and is also not defined in the Journey PlannerController, the request will fail. The TfL Journey Planner defaults to 'English' so if that's the value you want then you can leave this parameter out.

2.2 *Date/Time*

Entering the date and time is used by almost all types of requests. They can be used in many different ways with different date and time formats. The specification of this parameter is optional. If the parameter is missing the current date and time settings of the server will be used.

2.2.1 HTTP Parameters to specify the Date and Time of the journey

itdDate = <YYYYMMDD> | <YYMMDD>

Input of a full date. The value <*YYYYMMDD*> or <*YYMMDD*> consists of the four-or two-digit year, followed by the two-digit month (range: [1 .12]) and the two-digit indication of the day (range: [1 .31]). A delimiter is not used.

itdDateDay = <DD> | <D>

Input of the numerical day of the month. The day is given as one-or two-digit <*DD*> or <*D*>. At levels below ten, the leading zero will be given or taken away. Range of values: [1 .31].

itdDateMonth = <MM> | <M>

Input of the numerical calendar month of the year. The month is specified as one-or two-digit <*M*> or <*MM*>. At levels below ten, the leading zero will be given or taken away. Range of values: [1 .12].

itdDateYear = <YYYY> | <YY>

Input of the year. The year is specified as two-or four-digit value <*YYYY*> or <*YY*>.

itdDateYearMonth = <YYYYMM>

Input of the year and month. The value <*YYYYMM*> consists from the four-digit year, followed by the two digit month (range: [1 .12]). A delimiter is not used.

Note: This parameter will be overwritten by the parameter *itdDate*.

itdDateDayMonthYear = <DDMMYY> | <DDMMYYYY> | <DDxMMxYYYY>

Entering the day, month and year. The value <*DDMMYY*> or <*DDMMYYYY*> consists of two-digit day numbers (range: [1 .31]), followed by the two-digit month (range: [1 .12]), followed by the two-or four-digit year.

If you wish, you can use any delimiter x. In this case, the year must be in its four-digit form. The value of the parameter <DDxMMxYYYY> consists of the two-digit day, followed by a separator, followed by the two digit month, followed by a separator, followed by the four-digit year.

Note: Month and day cannot be handed over as single digits. This would lead to the separation of the value at the wrong place. The result of the output would be error messages.

Note: This parameter will be overwritten by the parameter *itdDate* or *itdDateYearMonth*.

itdTime = <time>

Entering the time. The value of <*Time*> can be given in one of the following formats:

- **HHMM**

  The two-digit number of hours are followed by the two-digit number of minutes of minutes.

- **HH:MM**

  The two-digit number of hours are followed by the two-digit number of minutes of minutes, separated by a colon.

- **HH.MM**

  The two-digit number of hours are followed by the two-digit number of minutes of minutes, separated by a dot.

- **MM**

  If only the number of minutes is passed, it is assumed for the hour to be 0. The specification of minutes can have one or two digits, i.e. with or without a leading zero.

- **HHMMa**

  The two-digit number of hours 12-hour format are followed by the two-digit number of minutes. The following a stands for the Anglo-American AM.

- **HHMMh**

  The two-digit number of hours are followed by the two-digit number of minutes. The following h indicates explicitly that the time will be displayed in the 24-hour format.

- **HHMMp**

  The two-digit number of hours 12-hour format are followed by the two-digit number of minutes. The following p stands for the Anglo-American PM.

The number of hours HH has the value range: [0 .24], the number of minutes the range [0 .59].

itdTimeAMPM = am | pm

In the Anglo-American time in 12-hour format, the value differs with the *am* and *pm* between morning and afternoon.

itdTimeHour = HH

Input of the hour. The value <*hh*> is in one-or two-digit form. At levels below ten, the leading zero can be omitted. Value range: [0 .24].

Note: To give the hour specification for the Anglo-American 12-hour format, the additional parameters *itdTimeAMPM* is necessary.

itdTimeMinute = MM

Input of the minutes. The value <*MM*> is in one-or two-digit. At levels below ten, the leading zero can be omitted. Value range: [0 .59].

itdTripDateTimeDepArr = arr | dep (default value)

Determines whether the date and time is related to the departure (*dep*), or the arrival (*arr*) time.

2.3 Point Verification – Origin, Destination, Via (ODV)

The point verification is required by virtually all types of requests. It can be carried out in a number of ways. Points can, for example, be determined by using coordinates, via stop IDs or by free text search. The HTTP parameters which can be used for the different types of input are listed below. Additional parameters can be used to enhance the search behavior. Others act as filters and restrict the search area or obtain a reduction of the list of possible matches.

ODV refers to the role that the point has within a request. The role of a point is particularly important for requests that require multiple verified points. A trip request, needs at least a start and an end point specified to be able to calculate the trip. In addition, intermediate points (also called via points) can be used. The function of a point is determined by adding _<*usage*> at the end of a parameter. Parameters that need this additional information are defined separately for each item according to its functionality. The additional information required depends on the request and will be explained in the subsequent sections of this documentation.

2.3.1 Minimal Request for the Point Verification

For the point verification by the Journey Planner System both of the parameters *name_<usage>* and *type_<usage>* are needed. Stops as well as points of interest and all other available types of objects input will be searched.

For the locality (which sets the wider area – such as a city boundary) the parameter *place_<usage>* needs to be set, i.e. for London *place_<usage>*=London. This parameter is recommended in every request.

If the stateful Journey Planner is used and it is not guaranteed that the verification point is a direct hit, a session must be generated. This enables the desired point to be selected from the hit list in a subsequent request by using the session ID. In order to generate the session, the parameter *sessionID = 0* in addition to the above parameters has to be set.

In the subsequent request the IDs of session and request have to be passed by the parameters *requestID* and *sessionID*. The ID of the selected hit is the value attribute of the element *odvNameElem* and is passed through the parameter *name_<usage>.* Additionally, the verification status indicated by the parameter *nameState_<usage>* needs to be given.

deleteAssignedStops_<usage> = 1 | on | selected

This parameter suppresses the selection of stops in the point verification, if the stop is marked by the suffix _<*usage*>, when it is activated by setting *selectAssignedStops = 1* or *selectAssignedStops_<usage> = 1*. The bus stop is directly verified.

Note: The selection of stops in the point verification process can be generally suppressed when you activate the parameter *deleteAssignedStops = 1*

name_<usage> = <name>

The value <*Name*> specifies the name of the item to be verified. For the point verification this parameter is the only user input which is required. For the two-field entry a specification of the place *place_<usage>* is necessary in addition.

For further specification of the value <*Name*> the parameter *type_<usage>* is required. For the two-field entry it is specifying the type

- stop,
- address,
- poi *[point of interest]*
- coord *[coordinates]*
- locator *[postcode]*.

<usage> = <name>

The usage can be set to

- origin,
- destination,
- via.

nameState_<usage> = empty (default value) | identified | list

The parameter indicates the server status of the verification of the point at *name_<usage>*. It is required for stateful requests. In the initial attempt to verify a point the value is empty. Since this is the default parameter it can be omitted.

In the XML output the server status is stored in the attribute *state* of the element *itdOdvName*. If an error occurs, the value of the attribute *state = unidentified*. A detailed error text is given in *itdMessage* as child element of *itdOdvName*.

See also *placeState_<usage>*.

Example: If a point request does not produce a single result, it must be followed by a choice of the desired item from a selection list of possible results. The choice is made by a request over the index of the list entry *name_<usage> = <index of listEntry>*. To prevent the initial re-verification of the value at *name_<usage>*, and in order to instruct the server to select a list item, *nameState_<usage> = list* must be set.

Note: When specifying stateful requests the ID of the session *sessionID* and the ID of the request *requestID* is mandatory.

place_<usage> = <location>

For two-field entry it is mandatory for the point verification to use <*location*>, besides the parameter *name_<usage>* and *type_<usage>,* as input. It specifies the location in which the point will be searched.

There is the possibility to verify a location on the municipal code (OMC) or the identification number. In order to do so <location> has to be specified in the following form. *placeID: <OMC>: <IdNumber>*. Hereby, the text *placeID* gives the type of the content which is to be verified. The following are the municipal code <*OMC*> and the identification number of the place <*IdNumber*>.

Note: The value of this parameter can be overwritten by the parameter *placeInfo_<usage>*.

placeDefaultText_<name> = <text>

A default text <*Text*> can be displayed in the input field for the location *place_<usage>*, without being considered in the point verification. This gives the possibility of displaying a help text in the entry field itself.

placeState_<usage> = empty (default value) | identified | list

The parameter indicates the server status of the verification of the point at *place_<usage>*. It is required for stateful requests. In the initial attempt to verify a point the value is empty. Since this is the default parameter it can be omitted.

In the XML output the server status is stored in the attribute *state* of the element *itdOdvName*. If an error occurs, the value of the attribute *state = unidentified*. A detailed error text is given in *itdMessage* as child element of *itdOdvName*.

See also *nameState_<usage>*.

Example: If a point request does produce a single result, it must be followed by a choice of the desired item from a selection list of possible results. The choice is made by a request over the index of the list entry *place_<usage> = <index of listEntry>*. To prevent the initial re-verification of the value at *place_<usage>*, and in order to instruct the server to select a list item, *placeState_<usage> = list* must be set.

Note: When specifying stateful requests the ID of the session *sessionID* and the ID of the request *requestID* is mandatory.

stateless = 1 | on | selected

This parameter is required for stateless requests so that the verification of a point is not lost in follow up requests. If the *sessionID* parameter is not specified, there is no session in which a clearly verified point is stored. If the value of the parameter in the initial request is set to 1, the output XML contains an attribute *stateless* in the *odvNameElem* elements. Using this attribute, the point can be clearly verified in a follow up request if the value of the attribute *stateless* is passed in the following request by the parameter *name_<usage>*.

type_<usage> = <type>

Specifies the type of an object which is given by *name_<usage>* For the older two-field input, the parameter can take different values.

- **address**
  Adress
- **locator**
  Postcode (mainly used in regions with very detailed zip code subdivisions, e.g. England)
- **place**
  Location (Gives the possibility to leave the parameter *name_<usage>* and only use *place_<usage>*)
- **poi**
  Point of interest or ID of a point of interest with prefix *poiID:*
- **poiID**
  ID of a point of interest
- **stop**
  Stops alias-Name or stops ID with prefix *stopID:*
- **stopID**
  ID of a stop
- **coord**
  Coordinate on GIS

Note: The value of this parameter can be overwritten by the parameter *typeInfo_<usage>*. The selected item type must be present in the GIS data. Otherwise the verification is unsuccessful.

2.3.2 Coordinates

Coordinates (coord) input:
The Journey Planner system can deal with different coordinate systems, for example Longitude/Latitude should be explained. Below parameters are required for a Long/ Lat coordinate query in London with the displayed name of *71 Fallowcourt Avenue, N12*: &name_destination=-0.179349:51.608163:WGS84[DD.ddddd]:71%20Fallowcourt%20Avenue,%20N12 &type_destination=coord

The structure is as follows:
&name_destination=WGS84[DD.DDDDD] &type_destination=coord

Coordinates (coord) output:
The coordinate output format can be specified:
&coordOutputFormat

The possible values are as follows:

WGS84	WGS84-Coordinates multiplied with factor 1,000,000
WGS84[DD.DDDDD] | WGS84[DD,DDDDD]	WGS84-Coordinates
WGS84[GGZHTXX]	WGS84-Coordinates multiplied with factor 100,000
WGS84[GGZHT]	WGS84-Coordinates multiplied with factor 1,000

2.4 Connection Options

The basic parameters of an HTTP request are perfect for the calculation of a trip, but for more complex requests, especially for people with special transport needs, they are not always sufficient. Therefore the additional parameters, referred to as Connection Options, offer further control over the results of a trip request. This makes it possible, for example, to calculate trips for people with reduced mobility, to reduce the walking speed or to exclude certain means of transport.

2.4.1 Parameter groups

The advanced features can be summarized into groups of HTTP parameters. To use the parameters of a group, it is necessary to activate them. Individual areas of functionality can thus be selectively enabled or disabled by using HTTP parameters. The following groups are available:

- *imparedOptionsActive* activates the required HTTP parameters for mobility options
- *itOptionsActive* activates the HTTP parameters for individual transport options
- *ptOptionsActive* activates the HTTP parameters that relate to the public transport options

2.4.2 HTTP parameters for changing connection options

calcNumberOfTrips = <number of trips>

Specifies the number of trips by public transport which are calculated. By default, at least four trips are calculated. If there are more alternative routes, the value of <number of trips> [Integer] in the XML output can be exceeded. Alternative routes in the XML output can be detected with the attribute *alternative*=1 in the *itdRoute* elements and can be suppressed with the parameter *noAlt*.

changeSpeed = <speed>

Sets the speed for interchange paths by foot, when the options for public transport are enabled by the parameter *ptOptionsActive = 1*. Sets the speed for the path from the starting point to the departure stop and the speed for the path from the destination stop to the destination, if the options for the individual transport are activated with the parameter *itOptionsActive =1*. If both options are active, the same speed will be used in both cases.

<speed> can have the following values:

- [25..400]

- fast
- normal
- slow

The default values are;

Alias-name	value
fast	50
normal	100
slow	200

As an alternative to the alias-names, values between 25 and 400 can be entered. The parameter value represents a kind of factor that modifies the default speed in the configuration file of the Journey Planner engine. The following formula is used:

```
speed [km/h] = (100 * speed) / value of parameter
```

Note: To use this parameter, you must have the options for public transport *ptOptionsActive* = *1*, the options for the individual transport *itOptionsActive* = *1* or both, enabled.

exclMOT_<ID> = -

This parameter causes the means of transport with the identification number <*ID*> to be excluded. To exclude multiple means of transport the parameter can be used multiple times. This parameter does not require a value. The means of transport with the identification number <ID> is excluded if the corresponding parameter is passed. In total there are 11 means of transport, which are documented in the appendix types of transportation.

Which means of transport were excluded from the calculation is in the XML output in the element *excludedMeans*. Each type is represented by an element *meansElem*. All elements with the attribute *selected* = *0* will be excluded from the calculation. The explanatory text corresponds to the standard means of transport regardless of the actual assignment.

Note: It is necessary activate this feature with the parameter *excludedMeans=checkbox*. Using this parameter means of transport can be excluded with the parameter *excludedMeans=<ID>*.

excludedMeans = checkbox | <ID of means of transport>

This *checkbox* the enables you to exclude certain means of transport for the trip request. The means of transport to be excluded are specified by the parameter *exclMOT_<ID>*. By default, all means of transport are active.

Alternatively, the means of transport to be excluded can be passed directly as a parameter *<ID of means of transport>*. To exclude multiple means of transport, the parameter can be used multiple times. The means of transport are documented in the section types of transportation.

imparedOptionsActive = 0 (default value) | 1

If this parameter is enabled, the options for restricted mobility are activated. If the parameter is not used or set to '0', the parameters that affect the mobility are not included.

inclMOT_<ID> = -

This parameter causes the means of transport with the identification number *<ID>* to be included by the system. If several means of transport are taken into account, the parameter can be used multiple times. This parameter does not require a value. The means of transport with the identification number is *<ID>* included when the corresponding parameter is passed. By default, only the chosen means of transport will be enabled when using the transportation inclusion. Altogether, there are 11 types which are described in the section types of transportation.

The means of transport excluded from the calculation are listed in the XML output in the element *excludedMeans* apparent. Each type is represented by an element *meansElem*. All elements with the attribute *selected = 0* are excluded from the calculation. The explanatory text corresponds to the above standard means of transport regardless of the actual assignment.

Note: It is necessary to activate this functionality with the parameter *includedMeans=1.*

includedMeans = checkbox | <ID of means of transport>

By the value of *checkbox* you can include certain means of transport in the trip request. The means of transport to be included are specified by the parameter *inclMOT_<ID>*. By default, all means of transport are disabled.

Alternatively, the id of the different kind of transport can be passed directly as a parameter *<ID of means of transport>*. To exclude multiple means of transport, the parameter can be used multiple times. The means of transport are documented in the section types of transportation.

itOptionsActive = 0 (default value) | 1

If this parameter is enabled, the options for individual transport are activated. If the parameter is disabled, the parameters for individual transport are not included.

cycleType = 107

If this parameter is enabled (together with &itOptionsActive=1), a cycle only option is calculated.

lowPlattformVhcl = -

If activated, only low-floor vehicles are considered. It is not necessary to assign a value to the parameter.

Caution: Once the parameter is passed, this option is active.

Note: To use this parameter the options for restricted mobility must be enabled with *imparedOptionsActive = 1*.

maxChanges = [0..9 (default value)]

Maximum number of changes in one trip. Results with more than the specified changes will be discarded for the trip request. By default 9 changes are defined as the maximum number. This applies as long as nothing else is specified at the server administration level. In the section [Parameters], the maximum can be set by the parameter *MaxNrChanges*.

Note: To use this parameter the options for restricted mobility must be enabled with *ptOptionsActive = 1*.

noAlt = 1 | true | on

Suppresses the calculation of alternative trips. By default alternative trips will be displayed also if they are considered as good connections, but would be considered as inferior judged by the value of the *routeType* parameter. Alternative trips can be selected in the XML output by the attribute *alternative=1* in the elements *itdRoute*.

Example: If route type, "shortest traveltime" is selected through the parameter *routeType=leastTime*, a trip with a time of 44 minutes and two changes under this option is better than a ride on the same departure time, which lasts 50 minutes and has only one change . If you want to offer this alternative, the alternative routes should not be suppressed by the parameter *noAlt = 1*.

Note: Disabling the alternative travel is not usually recommended as many users would rather see the available options and evaluate these themselves.

noElevators = -

If you pass this parameter, only trips with changes which don’t use elevators will be displayed. It is not necessary to assign a value to the parameter.

Caution: Once the parameter is passed, this option is active.

Note: To use this parameter, the option for limited mobility *imparedOptionsActive = 1* must be enabled.

noEscalators = -

If you pass this parameter, only trips with changes which don’t use escalators will be displayed. It is not necessary to assign a value to the parameter.

Caution: Once the parameter is passed, this option is active.

Note: To use this parameter, the option for limited mobility *imparedOptionsActive = 1* must be enabled.

noSolidStairs = -

If you pass this parameter, only trips with changes which don’t use stairs will be displayed. It is not necessary to assign a value to the parameter.

Caution: Once the parameter is passed, this option is active.

Note: To use this parameter, the option for limited mobility *imparedOptionsActive = 1* must be enabled.

ptOptionsActive = 0 (default value) | 1

With this parameter you turn on the additional options for public transport. If the parameter is disabled, the additional parameters that affect the public transport are not included.

routeType = <criterion>

Specifies the criterion <*criterion*> according to which the trip request should be optimized:

Criterion	Description
leastinterchange	Connections with least interchanges
leasttime	Fastest connections

leastwalking	Connections with least foothpats

Note: To use this parameter the options for public transport ptOptionsActive = 1 must be enabled.

trITArrMOT = <means of transport>

With this parameter, the means of transport <*means of transport*> from the starting point to the first stop for the individual transport is chosen. The possible means of transport are corresponding to those of the parameter *trITMOT*.

If the parameter *trITMOT* is also set, the chosen means of transport there for the starting point is overwritten by the parameter *trITArrMOT*. The maximum time for the way to the starting point can be influenced by the parameter *trITArrMOTvalue*.

Note: To use this parameter the options for individual transport with *itOptionsActive = 1* must be enabled.

trITArrMOTvalue = <time>

The value of the parameter <*time*> indicates the maximum time from the starting point to the first stop. The time is specified in minutes. The default is 10 minutes, if nothing else is defined at the server level.The default time can be set for all means of transport, which are described within the parameter *trITMOT,* in the section *[Parameter]* with the parameter *MaxITTime.* Optionally each type can be set individually by using the parameters *ITWalkSearchTime, ITBikeRideSearchTime*, and so on.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled. Also the means of transport that is used to calculate the time to the starting point has to be identified with the parameter *trITArrMOT*.

trITArrMOTvalue<means of transport> = <time>

The value of the parameter <*time*> indicates the maximum time from the starting point to the first stop for the selected means of transport <*means of transport*>. The time is specified in minutes. The default is 10 minutes, if nothing else is defined in the configuration file of the Journey Planner Controller or the Journey Planner Server. The default time can be set for all means of transport, which are described within the parameter *trITMOT,* in the section *[Parameter]* with the parameter *MaxITTime.* Optional each type can be set individually by using the parameters *ITWalkSearchTime, ITBikeRideSearchTime*, and so on.

The possible means of transport are analog to the *trITMOT* parameters. This parameter or the parameter *TrITArrMOT* is also used to select the means of transport for the starting point.

The parameter *trITArrMOTvalue<means of transport>* overwrites the selected travel time to the starting point from *trITMOTvalue<means of transport>*.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled.

trITDepMOT = <means of transport>

With this parameter, the means of transport <*means of transport*> from the destination stop to the destination target is selected. The possible means of transport are analog to the parameter *trITMOT*.

If the parameter *trITMOT* is also set, the selected means of transport for the destination target will be overwritten by the parameter *trITDepMOT*. The maximum time for the way from the destination stop to the destination target can be set with *trITDepMOTvalue*.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled.

trITDepMOTvalue = <time>

The value of the parameter <*time*> indicates the maximum time from the destination stop to the destination target. The time is specified in minutes. The default is 10 minutes, if nothing else is specified at the server level. The default time can be set for all means of transport, which are described within the parameter *trITMOT,* in the section *[Parameter]* with the parameters *MaxITTime.* Optionally each type can be set individually by using the parameters *ITWalkSearchTime, ITBikeRideSearchTime*, and so on.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled. Also the means of transport that is used until the starting point has to be selected with the parameter *trITArrMOT*.

trITDepMOTvalue<means of transport> = <time>

The value of the parameter <*time*> indicates the maximum time from the destination stop to the destination point for the selected means of transport <*means of transport*>. The time is specified in minutes. The default is 10 minutes, if nothing else is specified at the server level. The default time can be set for all means of transport, which are described within the parameter *trITMOT,* in the section *[Parameter]* with the parameters *MaxITTime.* Optionally each type can be set individually by using the parameters *ITWalkSearchTime, ITBikeRideSearchTime*, and so on.

The possible means of transport are analog to the *trITMOT* parameters. This parameter or the parameter *trITDepMOT* is also used to select the means of transport for the starting point.

The parameter *trITDepMOTvalue<means of transport>* overwrites the selected travel time from *trITMOTvalue<means of transport>*.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled.

trITMOT = < means of transport >

This parameter indicates the means of transport <*means of transport*> from the starting point to the first stop and from the destination stop to the destination point. The following values are possible:

ID	Means of transport
100	Footpaths
101	Bike & Ride
102	Take your bike along

The values are a subset of the means of transport for described in that section.

This parameter is overwritten by the parameters *trITArrMOT* and *trITDepMOT*. With them you can set the means of transport for start and end points separately.

The maximum time for the paths can be controlled via the parameter *trITMOTvalue*.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled.

trITMOTvalue = <time>

The value of the parameter <*time*> indicates the maximum time from the starting point to the first stop and also from the destination stop to the destination point. The time is specified in minutes. The default is 10 minutes, if nothing else is specified on the server side. The default time can be set for all means of transport, which are described within the parameter *trITMOT,* in the section *[Parameter]* with the parameters *MaxITTime.* Optionally each type can be set individually by using the parameters *ITWalkSearchTime, ITBikeRideSearchTime*, and so on.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled. Also the means of transport, that is used to calculate the time to the starting point, has to be identified with the parameter *trITArrMOT*. If the means of transport from start and destination are selected separately by the parameters *trITArrMOT* and *trITDepMOT* the parameters *trITArrMOTvalue* and *trITDepMOTvalue* must be set accordingly to influence the travel time.

trITMOTvalue<means of transport> = <time>

The value of the parameter <*time*> indicates the maximum time from the starting point stop to the first stop and from destination stop to destination target for the selected means of transport <*means of transport*>. The time is specified in minutes. The default is 10 minutes, if nothing else is defined in the configuration file of the Journey Planner Controller or the Journey Planner Server. The default time can be set for all means of transport, which are described within the parameter *trITMOT,* in the section *[Parameter]* with the parameters *MaxITTime.* Optional each type can be set individually by using the parameters *ITWalkSearchTime, ITBikeRideSearchTime*, and so on.

The travel times for start and finish can be set separately with the parameters *trITArrMOTvalue<means of transport>* and *trITDepMOTvalue<means of transport>.* These parameters overwrite the times which are chosen by the parameter *trITMOTvalue*.

The parameter *trITDepMOTvalue<means of transport>* overwrites the selected travel time from *trITMOTvalue<means of transport>*.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled.

wheelchair = -

If passed, only wheelchair accessible vehicles will be included in the request. Wheelchair accessible vehicles have a platform lift or a ramp or allow the same entry level. It is not necessary to assign a value to the parameter.

Caution: Once the parameter is passed, this option is active.

Note: To use this parameter, the options for individual transport *itOptionsActive = 1* must be enabled.

3 HTTP-Requests

3.1 Trip Request

Below the control of the Trip Request is explained. The trip request calculates trips to given date, time, arrival / departure, destination and travel options, which are controlled via HTTP parameters.

To get the XML-response you have to enter the following URL:
http://server:port/directory/ XML_TRIP_REQUEST2?HTTP parameters

Additionally, a PDF file which is suitable for printing can be generated from the response. There are different parameters *command=printOverview*, to create a PDF file with a trip overview, *command=printOverview* to create a PDF file with a trip overview and details, *command=printSingleview:<trip index>* to create a PDF file with the details of one single trip.

3.1.1 HTTP parameters of the Trip Request

command = <command>

With this parameter the execution of the request can be influenced. Depending on the request, <*command*> can accept many different values. There are some commands that apply to all types of requests and are documented on the request object.

In addition to the general commands there are the following values which can be used for the stop search:

- **changeDest**

  Reset the verification of the destination point. The previously verified points are preserved in the XML output to use as a predefined value in the input mask if needed. The status of *state* for the elements *itdOdvName* and *itdOdvPlace* (cf. *nameState_destination* and *placeState_destination*) within the container *itdOdv* with the attribute *usage=destination* is set to *notidentified*.

  Note: The complete trip can be reset by using the parameter *command=changeRequest*.

- **changeOrig**

  Resetting the verification of the starting point. The previously verified points are preserved in the XML output to use as a predefined value in the input mask if needed. The status of *state* for the elements *itdOdvName* and *itdOdvPlace* (cf. *nameState_origin* and *placeState_origin*) within the container *itdOdv* with the attribute *usage=origin* is set to *notidentified*.

  Note: The complete trip can be reset by using the parameter *command=changeRequest*.

- **changeRequest**

  Resetting the trip request. The previously verified points are preserved in the XML output to use as a predefined value in the input mask if needed. The status of *state* for the elements *itdOdvName* and *itdOdvPlace* (cf. *nameState_origin* and *placeState_origin*) is set to *notidentified*.

  Example: The interface offers a "change" button on the trip result monitor.

  Note: Start, destination and intermediate points can separately be reset with the parameters *command=changeOrig*, *command=changeDest*, *command=changeVia*. If the verification information of the points from an earlier request should be reset the parameter *command=odvReset* can be used.

- **changeVia**

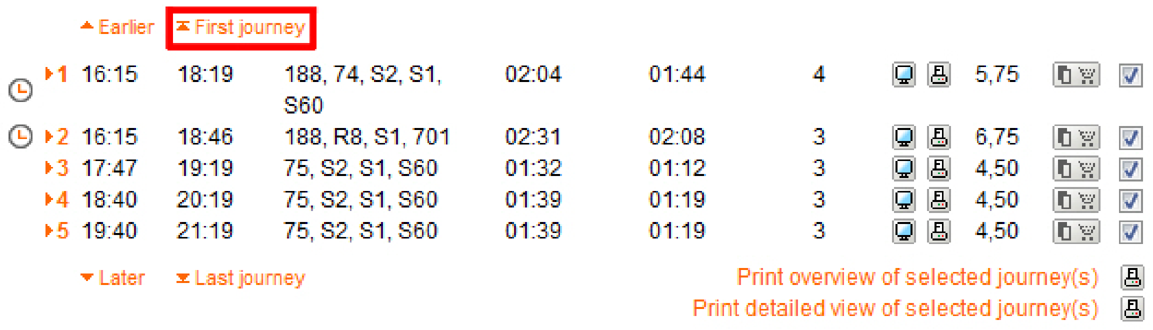
  Resetting the verification of via points. The previously verified points are preserved in the XML output to use as a predefined value in the input mask if needed. The status of *state* for the elements *itdOdvName* and *itdOdvPlace* (cf. *nameState_via* and *placeState_via*) within the container *itdOdv* with the attribute *usage=via* is set to *notidentified*.

  Note: The complete trip can be reset by using the parameter *command=changeRequest*.

- **tripFirst**

  Output of the first trip to the requested date. This trip will be displayed in addition to the other calculated trips.

  By default only the first trip is displayed. If the first x trips should be displayed, the number x can be set on the server side. The number of first and last trip (*see command=tripLast*) is identical.



▲ Earlier ▲ First journey

▸1	16:15	18:19	188, 74, S2, S1, S60	02:04	01:44	4	🖥	🖨	5,75	🛒	✓
▸2	16:15	18:46	188, R8, S1, 701	02:31	02:08	3	🖥	🖨	6,75	🛒	✓
▸3	17:47	19:19	75, S2, S1, S60	01:32	01:12	3	🖥	🖨	4,50	🛒	✓
▸4	18:40	20:19	75, S2, S1, S60	01:39	01:19	3	🖥	🖨	4,50	🛒	✓
▸5	19:40	21:19	75, S2, S1, S60	01:39	01:19	3	🖥	🖨	4,50	🛒	✓

▼ Later ▼ Last journey

Print overview of selected journey(s) 🖨

Print detailed view of selected journey(s) 🖨

  Example: A user gets the calculation for a trip from A to B. On the result screen, they choose the described functionality, because they want to know the earliest time they can arrive at the destination.

  Note: The last trip through can be calculated with the parameter *command=tripLast*. Earlier and later trips are calculated using *command=tripPrev* and *command=tripNext.*

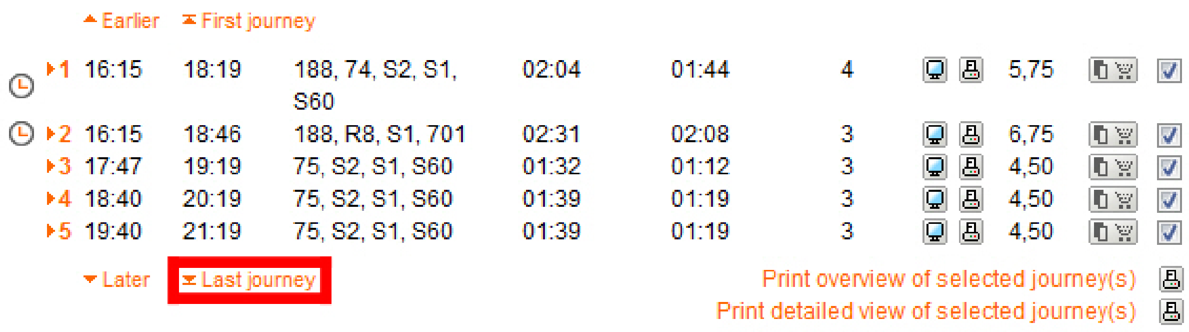
- **tripGoOn**

  Requests an onward journey. The starting point is initialized with the destination point of the previous request. The elements *itdOdvName* and *itdOdvPlace* within the container *itdOdv* with the attributes *usage=origin* will be filled with the information of the previous target point. The value of the attribute *state* of the elements *itdOdvName* and *itdOdvPlace* have the value *notidentified*. Thus, the later change of the starting point is possible.

- **tripLast**

  Output of the last trip to the requested date. This trip will be displayed in addition to the other calculated trips.

  By default only one last trip is displayed.The number of first and last trips shown (see *command=tripLast*) is identical.



1	16:15	18:19	188, 74, S2, S1, S60	02:04	01:44	4		5,75	✓
2	16:15	18:46	188, R8, S1, 701	02:31	02:08	3		6,75	✓
3	17:47	19:19	75, S2, S1, S60	01:32	01:12	3		4,50	✓
4	18:40	20:19	75, S2, S1, S60	01:39	01:19	3		4,50	✓
5	19:40	21:19	75, S2, S1, S60	01:39	01:19	3		4,50	✓

Earlier
First journey
Later
Last journey
Print overview of selected journey(s)
Print detailed view of selected journey(s)

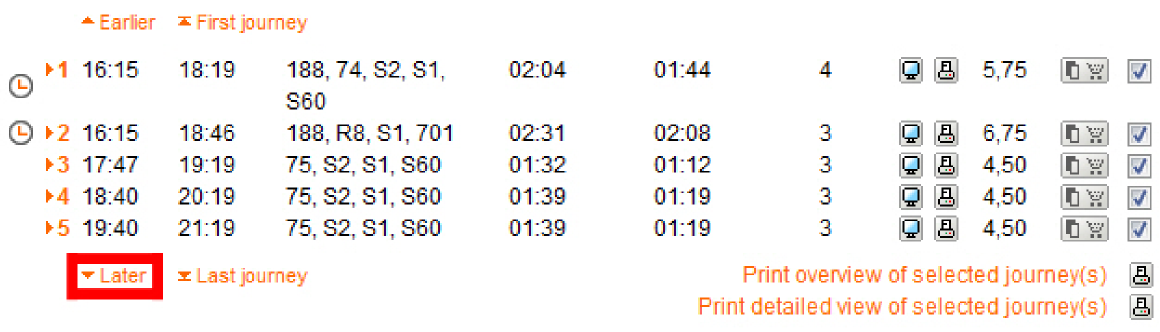
Example: A user wants to go out at night. They make a trip request from A to B. On the result screen, they choose the described functionality, because they want to know the last possible journey to get home.

Note: The last trip through can be calculated with the parameter *command=tripLast*. Earlier and later trips are calculated using *command=tripPrev* and *command=tripNext.*

- **tripNext**

  Output of later trips to the requested date. These trips will be displayed in addition to the other calculated trips.

  By default four later trips are displayed. The number of previous and later trips (see *command=tripPrev*) is identical.



▲ Earlier   ⏫ First journey

1	16:15	18:19	188, 74, S2, S1, S60	02:04	01:44	4	5,75	✓
2	16:15	18:46	188, R8, S1, 701	02:31	02:08	3	6,75	✓
3	17:47	19:19	75, S2, S1, S60	01:32	01:12	3	4,50	✓
4	18:40	20:19	75, S2, S1, S60	01:39	01:19	3	4,50	✓
5	19:40	21:19	75, S2, S1, S60	01:39	01:19	3	4,50	✓

▼ Later   ⏬ Last journey   Print overview of selected journey(s)   Print detailed view of selected journey(s)

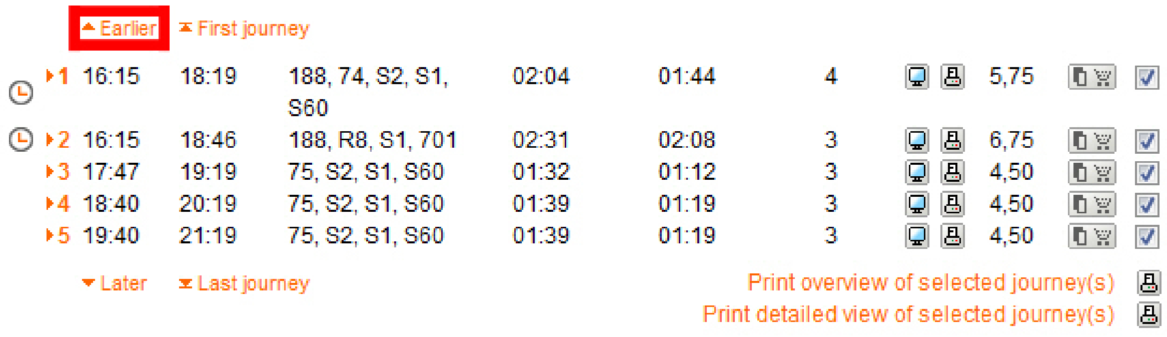
Example: In this way it enables the user to check on the results page, if they have a better connection departing a little later, or if they would also arrive on time with a later departure, without returning to the input screen and changing the departure time.

Note: The previous trips can be calculated with the parameter *command=tripPrev*. The first and last trips are calculated using *command=tripFirst* and *command=tripLast.*

- ### tripPrev

Output of previous trips to the requested date. These trips will be displayed in addition to the other calculated trips.

By default four previous trips are displayed. The number of previous and later trips (see *command=tripPrev*) is identical.



▲ Earlier   ⏫ First journey

1	16:15	18:19	188, 74, S2, S1, S60	02:04	01:44	4	5,75	✓
2	16:15	18:46	188, R8, S1, 701	02:31	02:08	3	6,75	✓
3	17:47	19:19	75, S2, S1, S60	01:32	01:12	3	4,50	✓
4	18:40	20:19	75, S2, S1, S60	01:39	01:19	3	4,50	✓
5	19:40	21:19	75, S2, S1, S60	01:39	01:19	3	4,50	✓

▼ Later   ⏬ Last journey   Print overview of selected journey(s)   Print detailed view of selected journey(s)

Example: In this way it enables the user to check on the results page, if they had a better connection departing a little earlier, or would arrive late with a later departure time, without returning to the input screen and change the departure time.

Note: The previous trips can be calculated with the parameter *command=tripPrev*. The first and last trips are calculated using *command=tripFirst* and *command=tripLast.*

- **tripRetoure**

  Request a return trip. The starting point and destination point are preset with the reversed start and destination point of the previous request. That means that the elements *itdOdvName* and *itdOdvPlace* within the container *itdOdv* with the attribute *usage=origin* in the XML output, are filled with the information of the previous destination point. The attribute *state* of the elements *itdOdvName* and *itdOdvPlace* have the value *notidentified*. The default setting of the target point is the same.

Note: This parameter is often used for a stateful follow up request, such as the change of points or the sending of a trip as an E-Mail. It should be noted that the identification number of the session and request *sessionID* and *requestID* must be specified.

3.1.2 HTTP parameters of the Cycle Journey Planner

In this section the parameters for the cycle route planner are described. For the point verification following parameters are required:

anyObjFilter_destination=0

anyObjFilter_origin=0

convertCoordsITKernel2LocationServer=1

convertCrossingsITKernel2LocationServer=1

convertPOIsITKernel2LocationServer=1

convertStopsPTKernel2LocationServer=1

locationServerActive=1

name_destination=<name>

name_origin=<name>

place_destination=London

place_origin=London

type_destination=any

type_origin=any

useCrossingList_destination=1

useCrossingList_origin=1

useLocalityMainStop=1

For a detail description of these please refers to the API_Documentation ODV document ("API Documentation ODV V03.docx").

advOptActive_2=1

This parameter activates the application of the advanced options for public transport. The option itself can be activated with the parameter advOpt_2.

Note: To use this parameter, the advanced options for public transport ptAdvancedOptions = *1* must be enabled.

advOpt_2=1

Enable the take bike along options for the TfL underground.

Note: To use this parameter, the options for public transport ptAdvancedOptions = *1* must be enabled. Additionally the option advOptActive_2=1 must be enabled.

ptAdvancedOptions=1

This parameter activates the advanced customer-specific public transport options.

bikeProf= <default | commuter | leisure>

With this parameter you can select a custom configuration for the weighting of the different street type's easy (default), moderate (leisure) and fast (commuter).

Note: To use this parameter, the options for individual transport itOptionsActive = *1* must be enabled.

bikeProfSpeed=default:12

This parameter is a combination of the bikeProf and cycleSpeed parameter. The value consists of the profile and the speed, separated by a colon. More information about creating can be found in the parameter bikeProf. The indication of speed is documented for the parameter cycle speed.

Example: To get easy, moderate and fast cycle journeys parameters must be passed three times:

http://server:port/directory/ XML_TRIP_REQUEST2?otherHTTP parameters&

bikeProfSpeed=default:12&bikeProfSpeed=leisure:16&bikeProfSpeed=commuter:20

Note: To use this parameter, the options for individual transport itOptionsActive = *1* must be enabled.

calcAltVarBicycle=1

This parameter is required to calculate alternative cycle journeys.

calcNumberOfTrips=1

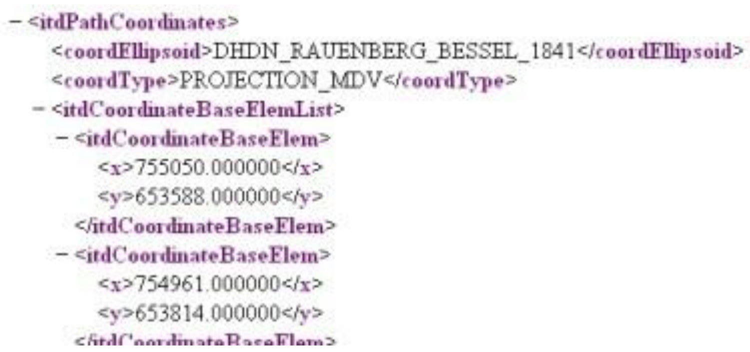
Please refer to Section calcNumberOfTrips = <number of trips> on page 14.

coordListOutputFormat=STRING

By default, the coordinates are displayed in a list itdCoordinateBaseElemList of elements itdCoordinateBaseElem in the container itdPathCoordinates in the XML output. To reduce the file size, the coordinate sequence can be displayed as a string. This parameter passed the coordListOutputFormat value string instead of list. The x and y values are separated by a comma, the coordinate pairs by a space.

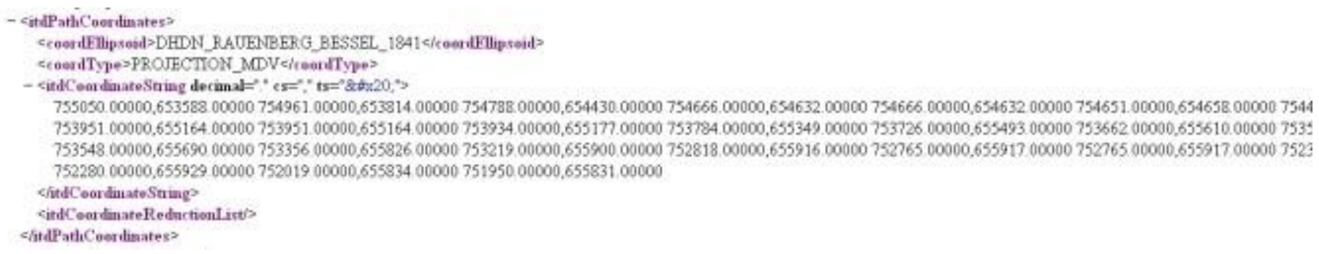
Example:

1. coordinate list: *list*



```
- <itdPathCoordinates>
    <coordEllipsoid>DHDN_RAUENBERG_BESSEL_1841</coordEllipsoid>
    <coordType>PROJECTION_MDV</coordType>
  - <itdCoordinateBaseElemList>
    - <itdCoordinateBaseElem>
        <x>755050.000000</x>
        <y>653588.000000</y>
      </itdCoordinateBaseElem>
    - <itdCoordinateBaseElem>
        <x>754961.000000</x>
        <y>653814.000000</y>
      </itdCoordinateBaseElem>
```

2. coordinate list: *string*



```
- <itdPathCoordinates>
    <coordEllipsoid>DHDN_RAUENBERG_BESSEL_1841</coordEllipsoid>
    <coordType>PROJECTION_MDV</coordType>
  - <itdCoordinateString decimal="." cs="," ts="&#x20;">
      755050.00000,653588.00000 754961.00000,653814.00000 754788.00000,654430.00000 754666.00000,654632.00000 754666.00000,654632.00000 754651.00000,654658.00000 7544
      753951.00000,655164.00000 753951.00000,655164.00000 753934.00000,655177.00000 753784.00000,655349.00000 753726.00000,655493.00000 753662.00000,655610.00000 7535
      753548.00000,655690.00000 753356.00000,655826.00000 753219.00000,655900.00000 752818.00000,655916.00000 752765.00000,655917.00000 752765.00000,655917.00000 7523
      752280.00000,655929.00000 752019.00000,655834.00000 751950.00000,655831.00000
    </itdCoordinateString>
    <itdCoordinateReductionList/>
  </itdPathCoordinates>
```

coordOutputFormatTail=0

By default, the coordinates are displayed with decimals. The number of decimal places may be limited by the value of the parameter. If the value is 0, no decimal point will be displayed.

Note: This parameter is only used to cut decimal figures. It does not increase the accuracy of the coordinates.

cycleSpeed=<speed>

With the value of the parameter <speed> the cycle speed is indicated in kilometers per hour.

Note: To use this parameter, the options for individual transport itOptionsActive = *1* must be enabled.

cycleType=107

This Parameter describes the role of the bicycle on the trip.

The following values are possible:

Value	Type
101	Bike & Ride
102	Take your bike along
107	Cycle only
99999994	Cycle hire stations

Note: To use this parameter, the options for individual transport itOptionsActive = *1* must be enabled.

cyclingActive=1

Calculate an additional cycle journey.

Note: To use this parameter, the options for individual transport itOptionsActive = *1* must be enabled.

delMinDistTrips=0

This parameter also calculates trips by public transport, even if a monomodal way is better.

everyMessage=1

There are also output error message either when no cycle journey or no public transport journeys are calculated.

imageNoTiles=1

Detailed maps can be generated to make the journey overview clearer. This functionality can be disabled with parameter imageNoTiles and the value is 1 or true.

purposeSuffix=HorizontalLayout

This parameter is used to request the PDF map format. This is set to horizontal by default.

searchLimitMinutes=360

The default look-ahead window for requests is 24 hours. With this parameter the preview window can be changed. The value is specified in minutes.

ITMOT=107

Please refer to Section trITMOT = < means of transport > on page 21.

trITMOTvalue=10

Please refer to Section trITMOTvalue = <time> on page 21.

useProxFootSearch=1

This parameter considers nearby stops in the trip calculation.

Note: To use this parameter, the options for public transport ptOptionsActive = *1* must be enabled.

vehTimeInclChTime=1

By default the total journey time of public transport is not including the connection time. With his parameter the time for connections and the total time of public transport will be added.

4 Appendix

4.1 *Means of Transport*

4.1.1 Public Transport

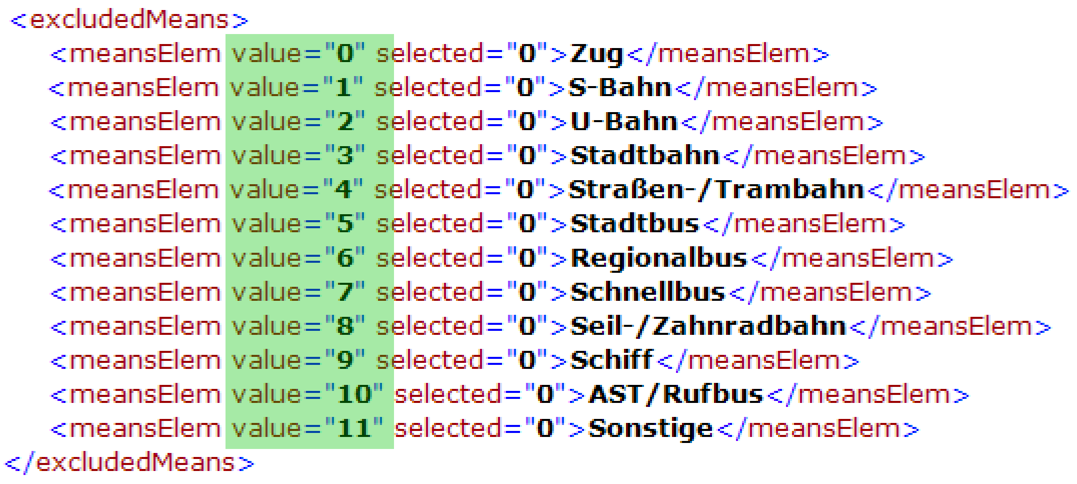
The means of transport (*MOTs* or *modes)* are defined as follows:

ID	Means of transport	TfL Usage (from w/c 09 December 2012)
0	Train	National Rail
1	Commuter railway	Docklands Light Railway
2	Underground	London Underground
3	City rail	London Overground
4	Tram	Tram
5	City bus	London Buses
6	Regional bus	*N.A.*
7	Coach	Coach
8	Cable car	Emirates Airline
9	Boat	London River Services
10	Transit on demand	*N.A.*
11	Other	Replacement Buses

Depending on the transport network there can be variations. Individual identification numbers can be assigned to other means of transport.

4.1.2 Using the Means of Transport IDs

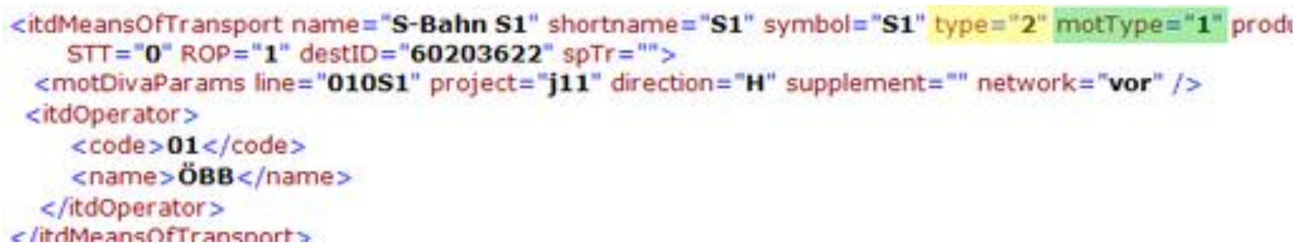
Traffic types are referenced by their ID number. The transport IDs are needed for example in the inclusion and exclusion of transport by the parameters *inclMOT* or *exclMOT.*



```
<excludedMeans>
   <meansElem value="0" selected="0">Zug</meansElem>
   <meansElem value="1" selected="0">S-Bahn</meansElem>
   <meansElem value="2" selected="0">U-Bahn</meansElem>
   <meansElem value="3" selected="0">Stadtbahn</meansElem>
   <meansElem value="4" selected="0">Straßen-/Trambahn</meansElem>
   <meansElem value="5" selected="0">Stadtbus</meansElem>
   <meansElem value="6" selected="0">Regionalbus</meansElem>
   <meansElem value="7" selected="0">Schnellbus</meansElem>
   <meansElem value="8" selected="0">Seil-/Zahnradbahn</meansElem>
   <meansElem value="9" selected="0">Schiff</meansElem>
   <meansElem value="10" selected="0">AST/Rufbus</meansElem>
   <meansElem value="11" selected="0">Sonstige</meansElem>
</excludedMeans>
```

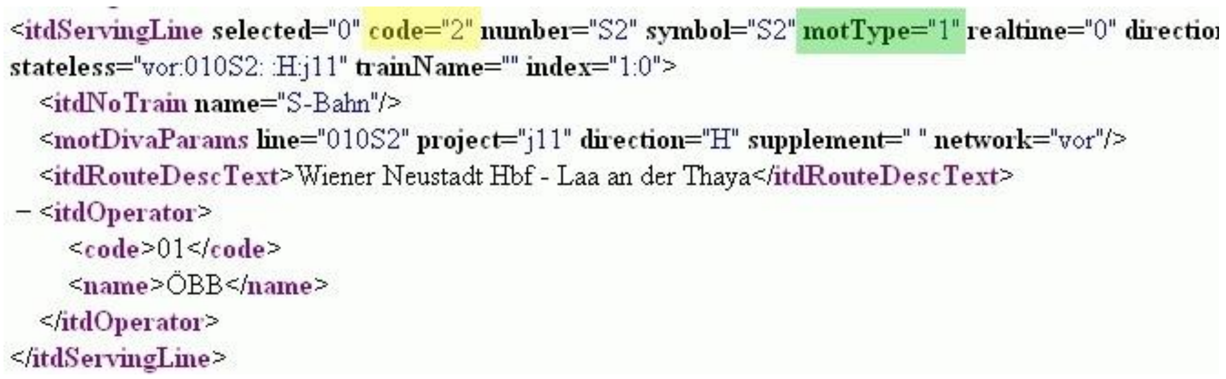
4.1.3 Means of Transport for Interchange vs. Transport Icons

Don’t confuse the means of transport for interchange (*type*) with the transport icon (*motType*). The means of transport for different partial routes *itdParitalRoute* is given in the attribute *motType* of the element *itdMeansOfTransport*, the transport icon is given in the attribute *type* of the element *itdMeansOfTransport.*



```
<itdMeansOfTransport name="S-Bahn S1" shortname="S1" symbol="S1" type="2" motType="1" prod
    STT="0" ROP="1" destID="60203622" spTr="">
  <motDivaParams line="010S1" project="j11" direction="H" supplement="" network="vor" />
  <itdOperator>
     <code>01</code>
     <name>ÖBB</name>
  </itdOperator>
</itdMeansOfTransport>
```

Information to enable the assignment of a transport icon is given in the attribute *code* of the element *itdServingLine* .



```
<itdServingLine selected="0" code="2" number="S2" symbol="S2" motType="1" realtime="0" directio
stateless="vor:010S2: :H:j11" trainName="" index="1:0">
   <itdNoTrain name="S-Bahn"/>
   <motDivaParams line="010S2" project="j11" direction="H" supplement=" " network="vor"/>
   <itdRouteDescText>Wiener Neustadt Hbf - Laa an der Thaya</itdRouteDescText>
 - <itdOperator>
      <code>01</code>
      <name>ÖBB</name>
   </itdOperator>
</itdServingLine>
```

The means of transport for interchange can be used to represent the transport icons in the web interface.

4.1.4 Individual Transport

There are the following means of transport defined when individual transport is enabled:

ID	Means of transport
100	Foothpath
101	Bike&Ride
102	Take your bike along
107	Bicycle

There is no separation in means of transport for interchange and transport icon. The IDs are used for the selection of the means of transport and the selection of the according transport icon. Input parameter that use the ID are for example the selection of means of transport by the parameter *trITMOT,* or the selection of bicycle mode by using the parameter *cycleType.* The ID, to distinguish between means of transport and display of the icon, can be found in the attribute *type* of the element *itdMeandsOfTransport* in the XML output.

4.1.5 Special Cases

The element *itdMeansOfTransport* in the XML output can have three other means of transport:

ID	Means of transport
97	Keep sitting (when the line changes number after a break)
98	Secure connection
99	Foothpath

For further transport types there are the same rules as for individual transport: There is no distinction between means of transport of interchange and transportation icon. In addition, they play no role in the entry, but only provide additional information in the XML output. The further means of transport are shown in the XML output as partial routes *itdPartialRoute.* They can be distinguished with the ID of the attribute *type* of the element *itdMeansOfTransport.*

5 Appendix A

TfL JP example queries:

5.1 Example Journey Request Stop to Stop

(Alexander Road --> Prince of Wales Gate) at 01/08/2011, 0800h

```
http://<api_location>/api/XML_TRIP_REQUEST2?language=en&sessio
nID=0&place_origin=London&type_origin=stop&name_origin=Alexand
er%20Road&place_destination=London&type_destination=stop&name_
destination=Prince%20Of%20Wales%20Gate&itdDate=20110801&itdTim
e=0800
```

5.2 Example Journey Request Address to Address

(12 Chatham Street --> 42 Gillingham Street)

```
http://<api_location>/api/XML_TRIP_REQUEST2?language=en&sessio
nID=0&place_origin=London&type_origin=address&name_origin=12%2
0Chatham%20Street&place_destination=London&type_destination=ad
dress&name_destination=42%20Gillingham%20Street
```

5.3 Example Journey Request Postcode to Postcode

(SW1H 0BD → AL2 1AE)

```
http://<api_location>/api/XML_TRIP_REQUEST2?language=en&sessio
nID=0&place_origin=London&type_origin=locator&name_origin=SW1H
%200BD&place_destination=London&type_destination=locator&name_
destination=AL2%201AE
```

5.4 Example Journey Request POI to POI

(O2 --> Madame Tussauds)

```
http://<api_location>/api/XML_TRIP_REQUEST2?language=en&sessio
nID=0&place_origin=London&type_origin=poi&name_origin=O2&place
_destination=London&type_destination=poi&name_destination=Mada
me%20Tussauds
```

5.5 Example Journey Request Stop to Coordinate

```
http://<api_location>/api/XML_TRIP_REQUEST2?language=en&sessio
nID=0&place_origin=London&type_origin=stop&name_origin=Alexand
er%20Road&name_destination=-
0.179349:51.608163:WGS84[DD.ddddd]:71%20Fallowcourt%20Avenue,%
20N12&type_destination=coord&itdDate=20110801&itdTime=0800
```

5.6 Example Cycle Journey Request Address to Address

```
http://<api_location>/api/XML_
TRIP_REQUEST2?language=en&sessionID=0&place_origin=London&type
_origin=address&name_origin=12 Chatham
Street&place_destination=London&type_destination=address&name_
destination=42 Gillingham
Street&advOptActive_2=1&advOpt_2=1&ptAdvancedOptions=1&bikePro
f=default&bikeProfSpeed=default:12&calcNumberOfTrips=1&cycleSp
eed=12&cycleType=107&cyclingActive=1&itOptionsActive=1&ptOptio
nsActive=1&searchLimitMinutes=360
```

5.7 Example Cycle Journey with easy, moderate and fast

```
http://cyclejourneyplanner.tfl.gov.uk/cycle/XSLT_TRIP_REQUEST2
?language=en&sessionID=0&locationServerActive=1&place_origin=L
ondon&type_origin=any&name_origin=12%20Chatham%20Street%20lond
on&place_destination=London&type_destination=any&name_destinat
ion=42%20Gillingham%20Street&advOptActive_2=1&advOpt_2=1&ptAdv
ancedOptions=1&bikeProf=default&bikeProfSpeed=default:12&bikeP
rofSpeed=leisure:16&bikeProfSpeed=commuter:20&calcNumberOfTrip
s=1&cycleSpeed=12&cycleType=107&cyclingActive=1&itOptionsActiv
e=1&ptOptionsActive=1&searchLimitMinutes=360&calcAltVarBicycle
=1
```