Real Time Passenger Information System and Related Services

UNIFIED REALTIME API 2

MAKING PREDICTION DATA AVAILABLE TO 3RD PARTIES

DOCUMENT CONTROL

AMENDMENT HISTORY

Issue	Date	Change Descriptions	Author
Draft 0.1	28.9.2011	Initial Document from PCP draft 5	Alex Gran
Draft 0.2	29.9.2011	Changes for full stream mode, tweaks and clarifications	Alex Gran
Draft 0.3	18.10.2011	Added StopPointType	Alex Gran
Draft 0.4	18.10.2011	EBNF added	Alex Gran
Draft 0.5	26.10.2011	Added TripNo, RegistrationNo	Alex Gran
Draft 0.6	26.10.2011	Changes from Anders LBSL Naming	Alex Gran
Issue 1.0	28.10.2011	Offer Version	Alex Gran
Issue 1.1	19.12.2011	After Review Meeting, includes static field order, some field compaction, API level versioning	Alex Gran
Issue 1.2	11.01.2012	Minor Corrections	Alex Gran
Issue 1.3	19.01.2012	Updates from Developer input, Adding priority to FLM (Array 2 renumbered)	Alex Gran
Issue 1.4	20.01.2012	Developer naming updates, typos	Peter Kehren
Issue 1.5	23.01.2012	Typo in EBNF, Circle in EBNF aligned to table 2, Made list behaviour clear in table 2	Alex Gran
Issue 1.6	25.01.2012	Remove obsolete reference to API Key, EBNF Typo, Version number must be positive	Alex Gran
Issue 1.7	31.01.2012	Drop search for Latitutde/Longitutde, remove required search parameter, remove outer array	Alex Gran
Issue 1.8	2.2.2012	TripID is Positive Integer, Response is newline seperated Fixed errors in Examples	Alex Gran
Issue 1.9	8.2.2012	Corrected EBFN	Alex Gran
Issue 1.9a	23.2.2012	Added ExpireTime to ReturnList	Alex Gran
Issue 1.9b	27.2.2012	Added further missing fields to ReturnList	Alex Gran
Issue 1.9c	28.2.2012	FLM StartTime, Stream parameter now different URL	Alex Gran

1.9d	1.3.2012	StopCode1 now StopID StopCode2 now StopCode1 StopCode3 now StopCode2	Alex Gran
1.9e	5.3.2012	Add telent reference number, Added extra Definitions	Andy Jepson
1.9f	16.3.2012	Ad IVU reference number Add messageUUID return list to	Alex Gran
1.9g	27.3.2012	Add TimeStamp in VersionArray	Alex Gran
1.9h	23.4.2012	Remove & escaping, not needed. Clarified http 401/403 returns	Alex Gran
1.9i	2.5.2012	Added Heartbeat	Alex Gran
1.9j	21.5.2012	As build preparation: mandatory URL, destination stop clarification	Alex Gran
1.9k	23.5.2012	After IVU review, minor clarifications	Alex Gran
2.0	11.6.2012	Formal Release as Built	Alex Gran
2.0a	12.6.2012	Initial suggestion for route & route geometry	Alex Gran
2.0b	18.7.2012	Further advanced features	Alex Gran
2.0c	20.9.2012	Adding main pattern	Alex Gran
2.0d	14.12.2012	Journey with timing Blocks (URASPEC-2) TripId now String (URASPEC-1)	Alex Gran
2.0f	22.01.2013	Update from Meeting with TfL	Alex Gran
2.0g	29.01.2013	Removed VehicleStatus from Vehicle Location	Alex Gran
2.0h	28.08.2013	Improve EstimatedTime Filtering, Add trip status (planned/unplanned, reasons for delay) Add vehicle modal type	Alex Gran
2.0i	11.09.2013	After coordination meeting. Replace various ArraySwitches by ResponseTypeFilter, Replace Journey by Trip Typos, fix inconsistencies between EBNF and tables	Verena Kriesel
2.0j	28.04.2014	Corrected some minor issues, extended the Vehicle Location Information Table with necessary, useful context information	Peter Kehren
2.0k	17.12.2014	Extended the Stop Information Table with information about car sharing and bike sharing	Zhaohui Wang

		stations	
2.0l	04.03.2015	Further extensions of the Stop Information Table, the Vehicle Modal Type Information Table and the Vehicle Location Information Table.	Dr. Alexander Kort
2.0m	02.06.2015	Added CourseId and PlannedDepartureTime to prediction table	Christoph Lischkowitz
2.0l	04.03.2015	Further extensions of the Stop Information Table, the Vehicle Modal Type Information Table and the Vehicle Location Information Table.	Dr. Alexander Kort
2.0m	02.06.2015	Added courseId and plannedDepartureTime to trip prediction table	Christoph Lischkowitz
2.01	4.11.2015	Changed modal types (chapter 1.6.2.8)	Iris Rottländer

1 UNIFIED REAL-TIME API 2.0

1.1 PREAMBLE:

This document uses special words in capital letters as per internet BCP RFC 2119.

Note that in EBNF statements:

- ^ is used as exclusive or: Only one alternative is accepted
- | is used as inclusive or: All alternatives may be used at most once in any sequence

1.2 OVERVIEW:

This API is intended to provide clients over the internet with real-time data.

Clients can be, but are not limited to:

- Third Party Servers
- Mobile (i.e. iPhone, Android) Apps
- Corporate Signs
- Websites integrating real-time Bus data

The aims of this interface are:

- Easy usage
- Short and simple standard
- Support for a broad range of applications

1.3 DATA TYPES SERVED

This API serves three types of data that have different characteristics and are therefore treated in differently:

1. Static Network Data, e.g. Schedule, bus stop names, line characteristics etc. This changes on a regularly, but relatively infrequently. In order to eliminate the client's need to download that separately – it can be of considerable size – this data is transmitted within every response as necessary.
2. Operational Types, e.g. Stop Point state types. This artifact can change as well, but very slowly. It varies however between PT organizations. It is therefore left to the API provider to define, without the need for a new Protocol Version
3. Known Fields, e.g. GPS Coordinates. These changes are very seldom. Any extension therefore needs a new API Version, however usually a new minor version is enough.

1.4 GENERAL OPERATION

The client MUST transmit the query parameters via the http query string. The HTTP transfer mode MUST be HTTP 1.1 with Content-Transfer-Encoding: chunked, enabling streaming of the data. The content-type of the data MUST be application/json as per RFC4627. The server MAY use a content-encoding header of gzip for long answers if the client supports that and signaled it in the HTTP request.

The client SHOULD set the ReturnList parameters to specify the needed values.

1.4.1 HTTP Status Codes

The Server MAY respond with HTTP status codes as per RFC 2616 to report errors. The table below lists protocol reason for specific error codes, they might be tied to the underlying data supply. Clients MUST be able to handle other RFC conform status as well.

HTTP Code	Reason	Comment
200 OK	Usual operation, response containing data	
204 No Content	System operation is ok, but no data is available.	This SHOULD not happen in normal operation, at least a base version or another error SHOULD be available.
400 Bad Syntax	URL is not understood, e.g. non-existing parameter	Client should not retry and needs to change the request
401 Unauthorized	Client did not pass a authentication, but the server is configured to require it, or credentials are wrong.	As per RFC
403 Forbidden	Server does not serve those request types, or credentials used are not sufficient.	Server COULD be configured not to support stream type requests, or requests for all data.
416 Requested range not satisfiable	Filter criteria out of range	The Client tried to use a filter criteria that is not matchable to the static data or formally out of range.
500 Internal Server Error	A generic error message, given when no more specific message is suitable	Exactly as per RFC
502 Bad Gateway	The Servers data source is not available	The response MAY contain an explanation which data source component is unavailable.

Table 1, Possible HTTP return codes

1.5 QUERY

The client requests a specific URL that instructs the server which data to return. This URL in EBNF is, in line with RFC2616:

```
http_URL = 'http:' '//' host [ ':' port ] [ abs_path/TYPE[_VN.N] [ '?'
query ]]
```

The fields host, port and abs_path are up to the server operator to define.TYPE is an indicator for the operation mode, either instant or stream. _VN.N with N being a positive Integer that MUST be appended to the abs_path to specify a specific version. The first Integer specifies the major version, the second Integer specifies the minor version. The minor Version can be omitted to use the most recent version.

In order to request a V2.* of this API, a client would therefore need to request abs_path_V2.

1.5.1 Query EBNF

Query and TYPE is specific to URA as:

```
TYPE := 'instant' ^ 'stream'

Query := QueryParameter *( '&' QueryParameter)

QueryParameter := (ReturnList | ResponseTypeFilter | EstimatedTimeFilter |
  CircleFilter | RectangularFilter | StopPointNameFilter |StopIDFilter |
  StopCode1Filter | StopCode2Filter | StopProviderFilter
  |StopPointTypeFilter | TowardsFilter | BearingFilter |
  StationSubtypeFilter | StopPointIndicatorFilter | StopPointStateFilter |
  LatitudeFilter | LongitudeFilter | StopAreaFilter |
  LocationCoordinateXFilter | LocationCoordinateYFilter |
  BoroughCodeFilter | StreetNameFilter | PostCodeFilter |
  ProductiveStopFilter | StopAccessibleFilter | VisitNumberFilter |
  LineIDFilter | LineNameFilter | DirectionIDFilter |
  DestinationTextFilter | DestinationNameFilter | VehicleIDFilter |
  TripIDFilter | TripID2Filter | RegistrationNumberFilter | RealtimeFilter
  | PredictionStatusFilter | VehicleLoadFilter | WheelChairLoadFilter |
  ExpireTimeFilter | MessageUUIDFilter | MessageTypeFilter |
  MessagePriorityFilter | MessageTextFilter | StartTimeFilter |
  PatternIDFilter | PatternVersionFilter | DirectionFilter |
  MainPatternSwitchFilter | OperatorCodeStartFilter |
  OperatorCodeEndFilter | LineTypeFilter | LineClassFilter |
  LineDescriptionFilter | LineFrequencyFilter | LineIdxStartFilter |
  LineIdxEndFilter | CalendarDayFilter | TripStartTimeFilter |
  TripEndTimeFilter | TripTypeFilter | OperatorIDFilter |
  OperatorNameFilter | OperatorAgencyFilter | GarageNameFilter |
  GarageCodeFilter | RunningNoFilter | DutyChangeOverFilter |
  BonnetNoFilter | VehicleStatusFilter | NameFilter | ModalTypeFilter |
  HeightFilter | LengthFilter | NoSeatsFilter | NoStandingFilter |
  NoWheelchairBaysFilter | LowFloorVehicleFilter | CO2Filter | NOXFilter |
  VehicleClassFilter |  EngineTypeFilter | TransmissionFilter |
  ObservationTimeFilter | VehicleLongitudeFilter | VehicleLatitudeFilter |
  HeadingFilter | OdometerFilter | PlannedDepartureTimeFilter |
  CourseIdFilter | Limiter )

~~StopAlsoSwitch := 'StopAlso=' ( true ^ false )~~

ResponseTypeFilter := 'ResponseType=' PositiveIntegerList

EstimatedTimeFilter := 'EstimatedTime=' PositiveIntegerList

CircleFilter := 'Circle=' Number ', ' Number ',' Number

RectangularFilter := 'Rectangle=' Number ',' Number ',' Number ',' Number

StopPointNameFilter := 'StopPointName=' StringList

StopIDFilter := 'StopID=' StringList

StopCode1Filter := 'StopCode1=' StringList

StopCode2Filter := 'StopCode2=' StringList

StopProviderFilter := 'Provider=' StringList

StopPointTypeFilter := 'StopPointType=' StringList

TowardsFilter := 'Towards=' StringList

BearingFilter := 'Bearing=' NumberList

StationSubtypeFilter := ‘StationSubtype=’ StringList
```

```
StopPointIndicatorFilter := 'StopPointIndicator=' StringList
StopPointStateFilter := 'StopPointState=' PositiveIntegerList
LatitudeFilter := 'Latitude=' NumberList
LongitudeFilter := 'Longitude=' NumberList
StopAreaFilter := 'StopArea=' StringList
LocationCoordinateXFilter := 'LocationCoordinateX=' NumberList
LocationCoordinateYFilter := 'LocationCoordinateY=' NumberList
BoroughCodeFilter := 'BoroughCode=' StringList
StreetNameFilter := 'StreetName=' StringList
PostCodeFilter := 'PostCode=' StringList
ProductiveStopFilter := 'ProductiveStop=' ( true ^ false )
StopAccessibleFilter := 'StopAccessible=' ( true ^ false ^ null)
VisitNumberFilter := 'VisitNumber=' NumberList
LineIDFilter := 'LineID=' StringList
LineNameFilter := 'LineName=' StringList
DirectionIDFilter := 'DirectionID=' ( 1 ^ 2 )
DestinationTextFilter := 'DestinationText=' StringList
DestinationNameFilter := 'DestinationName=' StringList
VehicleIDFilter := 'VehicleID=' StringList
TripIDFilter := 'TripID=' StringList
TripID2Filter := 'TripID2=' StringList
RegistrationNumberFilter := 'RegistrationNumber=' StringList
RealtimeFilter := 'Realtime=' ( true ^ false )
PredictionStatusFilter := 'PredictionStatus=' StringList
VehicleLoadFilter := 'VehicleLoad='
WheelChairLoadFilter := 'WheelChairLoad='
ExpireTimeFilter := 'ExpireTime=' PositiveIntegerList
MessageUUIDFilter := 'MessageUUID=' StringList
StopPointIndicatorFilter := 'StopPointIndicator=' StringList
MessageTypeFilter := 'MessageType=' PositiveIntegerList
MessagePriorityFilter := 'MessagePriority=' PositiveIntegerList
MessageTextFilter := 'MessageText=' StringList
StartTimeFilter := 'StartTime=' PositiveIntegerList
EndTimeFilter := 'EndTime=' PositiveIntegerList
PatternIDFilter := 'PatternID=' StringList
PatternVersionFilter := 'PatternVersion=' IntegerList
DirectionFilter := 'Direction=' StringList
MainPatternSwitchFilter := 'MainPatternSwitch=' PositiveIntegerList
OperatorCodeStartFilter := 'OperatorCodeStart=' StringList
OperatorCodeEndFilter := 'OperatorCodeEnd=' StringList
```

```
LineTypeFilter := 'LineType=' StringList
LineClassFilter := 'LineClass=' StringList
LineDescriptionFilter := 'LineDescription=' StringList
LineFrequencyFilter := 'LineFrequencyFilter=' NumberList
LineIdxStartFilter := 'LineIdxStart=' NumberList
LineIdxEndFilter := 'LineIdxEnd=' NumberList
CalendarDayFilter := 'CalendarDay=' PositiveIntegerList
TripStartTimeFilter := 'TripStartTime=' PositiveIntegerList
TripEndTimeFilter := 'TripEndTime=' PositiveIntegerList
TripTypeFilter := 'TripType=' StringList
OperatorIDFilter := 'OperatorID=' StringList
OperatorNameFilter := 'OperatorName=' StringList
OperatorAgencyFilter := 'OperatorAgency=' StringList
GarageNameFilter := 'GarageName=' StringList
GarageCodeFilter := 'GarageCode=' StringList
RunningNoFilter := 'RunningNo=' NumberList
DutyChangeOverFilter := 'DutyChangeOver=' ( true ^ false )
BonnetNoFilter := 'BonnetNo=' StringList
VehicleStatusFilter := 'VehicleStatus=' NumberList
NameFilter := 'Name=' StringList
ModalTypeFilter := 'ModalType=' StringList
HeightFilter := 'Height=' NumberList
LengthFilter := 'Length=' NumberList
NoSeatsFilter := 'NoSeats=' PositiveIntegerList
NoStandingFilter := 'NoStanding=' PositiveIntegerList
NoWheelchairBaysFilter := 'NoWheelchairBays=' PositiveIntegerList
LowFloorVehicleFilter:= 'LowFloorVehicle=' ( true ^ false )
CO2Filter := 'CO2=' NumberList
NOXFilter := 'NOX=' NumberList
VehicleClassFilter := 'VehicleClass=' StringList
EngineTypeFilter := 'EngineType=' StringList
TransmissionFilter := 'Transmission='
StringListObservationTimeFilter := 'ObservationTime=' PositiveIntegerList
VehicleLongitudeFilter := 'VehicleLongitude=' NumberList
VehicleLatitudeFilter := 'VehicleLatitude=' NumberList
HeadingFilter := 'Heading=' NumberList
OdometerFilter := 'Odometer=' PositiveIntegerList
PlannedDepartureTimeFilter := 'PlannedDepartureTime=' PositiveIntegerList
CourseIdFilter := 'CourseId=' StringList
ReturnList := 'ReturnList=' ReturnString *( ',' ReturnString )
```

```
ReturnString := 'StopPointName' | 'StopID' | 'StopCode1' | 'StopCode2' |
  'Provider' | 'PlaceAvailability' | 'StopPointState' | 'StopPointType' |
  'VisitNumber' | 'LineID' | 'LineName' | 'DirectionID' |
  'DestinationText' | 'DestinationName' | 'EstimatedTime' | 'Latitude' |
  'Longitude' | 'MessageText' | 'MessageType' | 'MessagePriority' |
  'MessageUUID| 'BaseVersion' | 'ExpireTime' | 'VehicleID' | 'TripID' |
  'TripID2' | 'RegistrationNumber' | 'Towards' | 'Bearing' |
  'StopPointIndicator | 'StartTime' | 'TripStartTime' | 'TripEndTime' |
  'PatternID' |  'CalendarDay' | 'OperatorName' | 'OperatorAgency' |
  'GarageName' | 'GarageCode' | 'DutyChangeOvers' | 'WaitTimeData' |
  'DriveTimeData' | 'StopArea' | 'LocationCoordinateX' |
  'LocationCoordinateY' | 'BoroughCode' | 'StreetName' | 'PostCode' |
  'ProductiveStop' | 'StopAccessible' | 'Realtime' | 'PredictionStatus' |
  'VehicleLoad' | 'WheelchairLoad' | 'PatternVersion' | 'Direction' |
  'MainPatternSwitch' | 'PatternStopIDs' | 'PatternDists' |
  'PatternCumulativeDists' | 'PatternShortDestinationNames' |
  'PatternLongDestinationNames' | 'PatternAlightInformation' |
  'GeometryData' | 'OperatorCodeStart' | 'OperatorCodeEnd' | 'LineType' |
  'LineClass' | 'LineDescription' | 'LineFrequency' | 'LineIdxStart' |
  'LineIdxEnd' | 'TripType' | 'OperatorID' | 'RunningNo' | 'BonnetNo' |
  'OperatorID' | 'VehicleStatus' | 'Name' | 'ModalType' | 'Height' |
  'Length' | 'NoSeats' | 'NoStanding' | 'NoWheelchairBays' |
  'LowFloorVehicle' | 'CO2' | 'NOX' | 'ObersvationTime' |
  'VehicleLongitude' | 'VehicleLatitude' | 'Heading' | 'Odometer' |
  'PlannedDepartureTime' | ' CourseId'

Limiter := 'Limit=' Number

StringList := EscapedString *( ',' EscapedString )

NumberList := Number *( ',' Number )

PositiveIntegerList := PositiveInteger *( ',' PositiveInteger )

PositiveInteger := '0'^'1'^'2'^'3'^'4'^'5'^'6'^'7'^'8'^'9'
  *('0'^'1'^'2'^'3'^'4'^'5'^'6'^'7'^'8'^'9')
```

1.5.2 Query Details Table

Note that to transform a usual String into an EscapedString, the following modifications have to be made:

1. Escape ',' by '\c'
2. percent-encoded result as URL String by RFC 3986

Parameter	Type	Valid Values	Default	Multiple Values Allowed	Comment
Key	String	Any String	Null	No	Allows to specify an API key, in case the server is required to need one.
ResponseType	Comma-separated Numbers	Integer 0…9	Unset	Yes	List of Response Type Information arrays to send out
ReturnList	Comma-separated Strings	see EBNF	All return array field names defined in API	No	List of Fields to return to client.
Limit	Number	Positive Integer > 0	Unset	No	Limits the amount of returned arrays per array type to Number. Not applicable to stream mode.
EstimatedTime	Special	“<Positive Integer” or “>Positive Integer”	Null	Yes	See below
Circle	Comma-separated Numbers	GPS Coordinates plus radius, Format is Circle=Latitude,Longitude,Radius (in m),e.g. Circle=12.3121412,14.1231241,100	Unset	No	Filter return for stops in given radius around given coordinates
Rectangle	Comma-separated Numbers	GPS Coordinates of a rectangle to search for. north west and south east coordinates of the bounding box, i.e. Rectangle=NWLat,NWLon,SELat,SELon	Unset	No	Filter return for stops in bounding box of given coordinates

Table 2, Request Filter Parameter List

Additionally to the table above, all return field names in tables 3-10, that have an “x” in the “Filter” column CAN be used as a query parameter. Their data type is a comma separated

list of their Specific value. For example to filter the response for StopCode 99 and 123, the query is StopCode=99,123. The filter criterias are case insensitive.

1.5.3 Estimated Time Filtering

When filtering for Estimated Time, using a specific value (i.e. EstimatedTime=1377694632000) usually makes little sense. Therefore the client can additionally use < and > as filter criteria. These follow the same boolean logic as other filters: Multiple estimatedTime filter parameters are ANDed together, multiple values are ORed together. I.e. All predictions between 1377694675000 and 1388694675000 would be EstimatedTime=>1377694675000, EstimatedTime=<1388694675000, whereas all predictions before and after that time would be EstimatedTime=<1388694675000,>1388694675000.

1.5.4 Constraints

The client SHOULD NOT set MessageType unless “MessageText” is part of the ReturnList.

Clients SHOULD reduce the ReturnList to the absolute minimum.

The server MAY NOT support the “If-Modified-Since” HTTP feature.

1.6 RESPONSE

The server responds with an UTF-8 JSON message. This are newline separated arrays, either stop, prediction, flexible message, line geometry, pattern, trip, trip time, base version or URA version arrays.

The empty response is just a version array: [4,"2.0"]

The sequence of the field values in the prediction, flexible message and stop Information arrays MUST match the tables below. None requested fields are skipped. The sequence of the ResponseArrays in the response is undefined, except that the URA Version Array appears first. Clients MUST NOT make any other assumption about the sequence in the return.

The `ResponseTypeFilter` determines which array types are returned. The first value in each array MUST be the Type field, indicating the arrays type.

1.6.1 Response EBNF

```
Response = URAVersionArray [ResponseArray] *(Whitespace 'Response Array)

ResponseArray := StopArray ^ PredictionArray ^ FLMArray ^ BaseVersionArray
  ^ PatternArray ^ LineGeometyArray ^ TripArray ^ TripTimeArray ^
  VehicleArray ^ VehicleLocationArray


URAVersionArray := '[4,' PositiveInteger '.' PositiveInteger ','
  PositiveInteger ']'


StopArray := '[0,' StopPointName | ',' StopID |
```

```
',' StopCode1 | ',' StopCode2 |',' StopPointType |',' Provider |','
StationSubtype | ',' TotalVehicles | ',' TotalSpaces| ',' AvailVehicles|
',' AvailSpaces | ',' VehiclesAtStation | ',' Towards | ',' Bearing |
',' StopPointIndicator | ',' StopPointState | ',' Latitude | ','
Longitude | ',' StopArea | ',' LocationCoordinateX | ','
LocationCoordinateY | ',' BoroughCode | ',' StreetName | ',' PostCode |
',' ProductiveStop | ',' StopAccessible ']'


PredictionArray := '[1,' StopID | ',' VisitNumber | ',' LineID | ','
LineName | ',' DirectionID | ',' DestinationText | ',' DestinationName |
',' VehicleID | ',' TripID | ',' TripID2 |',' RegistrationNumber |',','
Realtime |',' PredictionStatus |',' VehicleLoad |',' WheelChairLoad |',','
EstimatedTime |  ',' ExpireTime |  ',' PlannedDepartureTime |  ','
CourseId ']'


FLMArray := '[2,' StopID | ',' TripID | ',' MessageUUID | ',' MessageType
| ',' MessagePriority | ',' MessageText | ',' StartTime | ',' ExpireTime
']'


PatternArray := '[5,' PatternID | ',' LineID | ',' LineName | ','
Direction |',' MainPatterBoolean |',' PatternStopIDData |','
PatternDistsData |',' PatternCumulativeDistData |','
PatternShortDestData |',' PatternLongDestData |',' PatternAlightData
|',' GeometryData |',' OperatorCodeStart |',' OperatorCodeEnd |','
LineType |',' LineClass |',' LineDescription |',' LineFrequency |','
LineIdxStart |',' LineIdxEnd ']'


PatternStopIDData := '[' [StopID] *(',' StopID ) ']'


GeometryData := '[' GeometryPoint *(',' GeometryPoint) ']'


GeometryPoint := '[' Latitude ',' Longitude ']'


TripArray = '[6,' TripID | ',' TripID2 | ',' PatternID | ',' CaldendarDay
| ',' StartTime  | ',' EndTime  | ',' TripType | ',' OperatorID | ','
OperatorName | ',' OperatorAgency | ',' GarageName | ',' GarageCode |
',' RunningNo ']'


TripTimeArray = '[7,' TripID | ',' DutyChangeOverDatas | ',' WaitTimeData
| ',' DriveTimeData ']'


DutyChangeOverData := '[' [StopID] *(',' StopID ) ']'


WaitTimeData := '[' [Time] *(',' Time) ']'


DriveTimeData := '[' [Time] *(',' Time ) ']'


VehicleArray = '[8,' VehicleId | ',' RegistrationNumber | ',' BonnetNo |
',' OperatorID | ',' VehicleStatus | ',' Name | ',' ModalType | ','
```

```
  Height | ',' Length | ',' NoSeats | ',' NoStanding | ','
  NoWheelchairBays | ',' LowFloorVehicle | ',' CO2 | ',' NOX  |','
  VehicleClass  |',' Convertible  |',' EngineType  |',' HorsePower  |','
  TrunkCapacity  |',' Consumption  |',' Color  |',' Doors  |','
  Transmission  |',' OpeningTime  |',' WinterTire  |',' AirCondition  |','
  ChildCarSeat  |',' Navigation  |',' CruiseControl  |','
  AdditionalInformation |']'


VehicleLocationArray = '[9,' VehicleId | ',' ObservationTime | ','
  VehicleLongitude | ',' VehicleLatitude | ',' Heading | ',' Odometer |
  ',' TripID ']'


BaseVersionArray := '[3,' String ']'
```

1.6.2 Response Detail Tables

The following tables list the possible response array fields, including their formal type.

The first table listing stop information, the other response reference it by using the StopID. The response is a sequence of values given by the tables, but only the values from the Return List parameter, containing fields from table 3 and 4 for predictions arrays and from table 3 and 5 for flexible message arrays. Stop information arrays contain only fields from table 3.

1.6.2.1 STOP INFORMATION TABLE

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	0	No		
1	StopPointName	String		No	X	
2	StopID	String		Yes	X	technical Stop Identifier used
3	StopCode1	String		No	X	Primary public code SMS Stop Code
4	StopCode2	String		No	X	Secondary Stop Code
5	StopPointType	String		No	X	Allows the distinction between different stop points (e.g. car sharing, bus stops, ...)
6	Towards	String		No	X	
7	Bearing	Number	Integer 0…359 degrees	No	X	
8	StopPointIndicator	String		No	X	
9	StopPointState	Number	Integer >= 0	No	X	
10	Latitude	Number	Every Number	No	X	
11	Longitude	Number	Every Number	No	X	
12	StopArea	String		No	X	
13	LocationCoordinateX	Number	Implementation depended	No	X	Alternative format for describing the coordinate, X part
14	LocationCoordinateY	Number	Implementation depended	No	X	Alternative format for describing the coordinate, y part
15	BoroughCode	String		No	X	
16	StreetName	String		No	X	

17	PostCode	String		No	X	
18	ProductiveStop	Boolean		No	X	Determines if that stop is used during normal operation
19	StopAccessible	Boolean	True/False/Null	No	X	Determines if that stop is accessible. Null if unknown
20	Provider	String		No	X	The name of the company who runs the car sharing or bike sharing
21	StationSubtype	String		No	X	In case of a sharing station: What kind of sharing station (‘Bike’, ‘Car’, etc.)
22	TotalVehicles	Number		No		In case of a sharing station: The total number of vehicles currently at the station, including reserved/unavailable vehicles.
23	TotalSpaces	Number		No		In case of a sharing station: The maximum number of vehicles which can be stored at the sharing station, i.e. the number of parking lots.
24	AvailVehicles	Number		No		In case of a sharing station: The number of vehicles (e.g. cars or bikes) which are available for sharing.
25	AvailSpaces	Number		No		In case of a sharing station: The current number of available free parking lots.
26	VehiclesAtStation	String		No		In case of a sharing station: A ‘;’-separated list of vehicle ids of the vehicles at the station.

Table 3, Stop Information Return Fields

1.6.2.2 PREDICTION INFORMATION TABLE

The following table lists the fields of a prediction array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment

0	ResponseType	Number	1	No		
1	StopID	String		Yes	X	technical Stop Identifier used
2	VisitNumber	Number	Every positive Integer	Yes	X	
3	LineID	String		No	X	Also called Contract Line Number
4	LineName	String		No	X	Also called Service Line Number
5	DirectionID	Number	1 or 2	No	X	
6	DestinationText	String		Yes	X	
7	DestinationName	String		No	X	
8	VehicleID	String		Yes	X	
9	TripID	String		No	X	Primary used identifier of a trip
10	TripID2	String		No	X	Secondary used identifier of a trip
11	RegistrationNumber	String		No	X	
12	Realtime	Boolean		No	X	Realtime or planned trip
13	PredictionStatus	String		No	X	Further Prediction status: e.g. deleted, waiting for connection, jammed, detoured etc.
14	VehicleLoad	Number	Positive float number between 0 and 1	No	X	Amount of ordinary passenger space used
15	WheelChairLoad	Number	Positive float number between 0 and 1	No	X	Amount of wheel chair passenger space used
16	EstimatedTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch
17	ExpireTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch
18	PlannedDepartureTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch
19	CourseId	String		No	X	

Table 4, Prediction Array Return Fields

Note that in ord er to have a unique identifier for a prediction the client COULD need to include the primary key of the stop information.

1.6.2.3 FLEXIBLE MESSAGE INFORMATION TABLE

The following table lists the fields of a flexible message array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	2	No		
1	StopID	String		Yes	X	Primary, technical Stop Code used
2	TripID	String		Yes	X	
3	MessageUUID	String		Yes	X	UUID to identify the message
4	MessageType	Number	Integer >= 0	No	X	See section 2.7
5	MessagePriority	Number	Integer 1...10	No	X	10 is highest Priority, 1 is lowest Priority
6	MessageText	String		No	X	
7	StartTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch
8	ExpireTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch

Table 5, Flexible message Array Return Fields

Messages CAN belong to stops, trips or both.

Clients SHOULD NOT make assumptions about the format of the String parameters coming from static data.

The server SHOULD set the ExpireTime to 0 in order to “delete” predictions and flexible message in stream mode.

1.6.2.4 PATTERN MESSAGE INFORMATION TABLE

The following table lists the fields of a pattern array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	5	No		
1	PatternID	String		Yes	X	
2	PatternVersion	Number		Yes	X	Allows changes of the Line Geometry during the
2	LineID	String		No	X	
3	LineName	String		No	X	
4	Direction	String		No	X	
5	MainPatternSwitch	Number	Every positive Integer	No	X	1 Indicates if that pattern is the main pattern of that LineID and Direction 0 indicates it is not the main pattern. Other numbers implementation specific
6	PatternStopIDs	Array	See below	No		Array of StopIDs
7	PatternDists	Array	See below	No		Array of Numbers
8	PatternCumulativeDists	Array	See below	No		Array of Numbers
9	PatternShortDestinationNames	Array	See below	No		Array of Strings
10	PatternLongDestinationNames	Array	See below	No		Array of Strings
11	PatternAlightInformation	Array	See below	No		Array of Strings
12	GeometryData	Array	See below	No		Array of arrays of GPS latitude& GPS longitude
13	OperatorCodeStart	String		No	X	Operator code at start of schedule
14	OperatorCodeEnd	String		No	X	Operator code at end of

					schedule
15	LineType	String	No	X	
16	LineClass	String	No	X	
17	LineDescription	String	No	X	
18	LineFrequency	Number	No	X	Higher number means higher frequency, exact meaning is implementation specific
19	LineIdxStart	Number	No	X	Planning System line number at start of schedule
20	LineIdxEnd	Number	No	X	Planning System line number at start of schedule

Table 6, Pattern message Array Return Fields

All PatternData Arrays are a JSON Array of StopIDs, i.e. [StopID1, StopID2…] or Numbers, Strings etc.

Note that there is only one Pattern with MainPattern = 1 for every combination of LineID and Direction.

GeometryData is a JSON Array of Location tuples ,i.e. [[0.123,50.2331],[-1.121,32.12121]…].

1.6.2.5 TRIP MESSAGE INFORMATION TABLE

The following table lists the fields of a trip array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	6	No		
1	TripID	String		Yes	X	
2	TripID2	String		No	X	Alternative identifier for trip
3	PatternID	String		No	X	Corresponding Pattern
4	CalendarDay	Number	Every positive Integer	No	X	
5	StartTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch
6	EndTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch
7	TripType	String		No	X	
8	OperatorID	String		No	X	
9	OperatorName	String		No	X	
10	OperatorAgency	String		No	X	
11	GarageName	String		No	X	
12	GarageCode	String		No	X	
13	RunningNo	Number		No	X	

Table 7, Trip message Array Return Fields

1.6.2.6 TRIP TIME MESSAGE INFORMATION TABLE

The following table lists the fields of a trip wait time array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	7	No		
1	TripID	String		Yes	X	
2	DutyChangeOvers	Array of Strings	StopIDs	No	X	Gives all StopIDs where the duty changes
3	WaitTimeData	Array	See below	No		

4	DriveTimeData	Array	See below	No

Table 8, Trip time message Array Return Fields

WaitTimeData is a JSON Array of positive Integers, i.e. [10,22,1,0,115] in s.

DutyChangeOvers is a JSON Array of Strings that give the StopIDs where a duty change over happens.

DriveTimeData is a JSON Array of positive Integers, i.e. [10,22,1,0,115] in s.

1.6.2.7 VEHICLE MESSAGE INFORMATION TABLE

The following table lists the fields of a vehicle array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	8	No		
1	VehicleId	String		Yes	X	
2	RegistrationNumber	String		No	X	
3	BonnetNo	String		No	X	
4	OperatorID	String		No	X	
5	VehicleStatus	Number		No	X	Indicates the status of the vehicle: 1 = Commissioned 2 = Planned 99 = Decommissioned vehicle Other values implementation specific
6	Name	String		No	X	The name of the vehicle. In case of an individual car this is the name of the car type (e.g. 'Ford Fiesta')
7	Modal Type	String		No	X	Indicates the modal type of the vehicle: Use PTEG values , other values implementation specific
8	Height	Number	Positive Number	No	X	Height in m
9	Length	Number	Positive Number	No	X	Length in m
10	NoSeats	Number	Positive Number or Null	No	X	Null if unknown
11	NoStanding	Number	Positive Number or Null	No	X	Null if unknown
12	NoWheelchairBays	Number	Positive Number or Null	No	X	Null if unknown
13	LowFloorVehicle	Boolean	True/False/Null	No	X	Null if unknown
14	CO2	Number	Positive Number or Null	No	X	g per km, Null if unknown

15	NOX	Number	Positive Number or Null	No	X	g per km, Null if unknown
16	VehicleClass	String		No	X	Vehicle segment classification, e.g. ‘bike’, ’motorcycle’, ’micro’, mini’, ’small’, ’medium’, ’large’, ’van’ ,’transporter’, ’suv’
17	Convertible	Boolean	True/False/Null	No		Is the car a convertible?
18	EngineType	String			X	E.g. ‘none’, ’diesel’,’ gasoline’, ’electric’, ’liquidgas’, ’naturalgas’, ‘hydrogen’, ‘hybrid’
19	HorsePower	Number	Positive Number or Null			The car’s horsepower.
20	TrunkCapacity	Number	Positive Number or Null	No		Capacity of the Trunk for cars and busses.
21	Consumption	Number	Positive Number or Null	No		A measure for energy consumption per kilometre.
22	Color	String	Any color description	No		The primary color of the car.
23	Doors	Number	Positive Number or Null	No		Number of doors of the vehicle
24	Transmission	String	‘manual’ or ‘automatic’	No	X	The transmission of the char.
25	OpeningTime	Number	Positive Number or Null	No		Minimum time required to open the car.
26	WinterTire	Boolean	True/False/Null			Is the car equipped

					with snow tires?
27	AirCondition	Boolean	True/False/Null	No	Indicates whether the car has an air conditioning.
28	ChildCarSeat	Number	Positive Number or Null		
29	Navigation	Boolean	True/False/Null	No	Has the car a navigation system onboard?
30	CruiseControl	Boolean	True/False/Null	No	Has the car a cruise control system?
31	AdditionalInformation	String	A String containing key-value-pairs separated by ';'	No	Additional information formatted as key-value pairs.

Table 9, Vehicle information message Array Return Fields

1.6.2.8 VEHICLE MODAL TYPE INFORMATION TABLES

The following table contains allowed vehicle modal types. There is no distinction between individual modes and public transport modes, so that all strings from the right-hand side of the table are to be used interchangeably without any prefix.

Examples: *"walk", "bus", "tram"*

IndividualMode possible Values	*walk* | *cycle* | *taxi* | *self-drive-car* | *others-drive-car* | *motorcycle* | *truck*
PtMode possible Values	*all* | *unknown* | *air* | *bus* | *trolleyBus* | *tram* | *coach* | *rail* | *intercityRail* | *urbanRail* | *metro* | *water* | *cableway* | *funicular* | *taxi*

Table 1:Modal Types

Optionally, submodal types can be appended to the main modal type in the following form:

<modal type>/<submodal type>

The possible submodal types depend on the main modal type as specified in the following table.

Examples: *"bus/localBus", "rail/nightRail", "tram/shuttleTram"*

RailSubmode possible Values	*local* | *highSpeedRail* | *suburbanRailway* | *regionalRail* | *interregionalRail* | *longDistance* | *international* | *sleeperRailService* | *nightRail* | *carTransportRailService* | *touristRailway* | *railShuttle* | *replacementRailService* | *specialTrain* | *crossCountryRail* | *rackAndPinionRailway*
MetroSubmode possible Values	*metro* | *tube* | *urbanRailway*
BusSubmode possible Values	*localBus* | *regionalBus* | *expressBus* | *nightBus* | *postBus* | *specialNeedsBus* | *mobilityBus* | *mobilityBusForRegisteredDisabled* | *sightseeingBus* | *shuttleBus* | *schoolBus* | *schoolAndPublicServiceBus* | *railReplacementBus* | *demandAndResponseBus* | *airportLinkBus*
TramSubmode	*cityTram* | *localTram* | *regionalTram* | *sightseeingTram* | *shuttleTram*

possible Values	
WaterSubmode possible Values	*internationalCarFerry* | *nationalCarFerry* | *regionalCarFerry* | *localCarFerry* | *internationalPassengerFerry* | *nationalPassengerFerry* | *regionalPassengerFerry* | *localPassengerFerry* | *postBoat* | *trainFerry* | *roadFerryLink* | *airportBoatLink* | *highSpeedVehicleService* | *highSpeedPassengerService* | *sightseeingService* | *schoolBoat* | *cableFerry* | *riverBus* | *scheduledFerry* | *shuttleFerryService*

TABLE 2: SUBMODAL TYPES

1.6.2.9 VEHICLE LOCATION INFORMATION TABLE

The following table lists the fields of a vehicle location array.

Sequence Nr	Field	JSON-Type	Valid Values	PK	Filter	Comment
0	ResponseType	Number	9	No		
1	VehicleId	String		Yes	X	
2	TripID	String		No	X	Currently served trip
3	ObservationTime	Number	Every positive Integer	No	X	UTC as per Unix Epoch, last time vehicle send position information
4	VehicleLongitude	Number	Every number	No	X	GPS Coordinate
5	VehicleLatitude	Number	Every number	No	X	GPS Coordinate
6	Heading	Number	Number between 0 and 360	No	X	Heading in degrees
7	Odometer	Number	Every positive Integer	No	X	In m
8	VehicleLineName	String		No	X	Currently served line (seen by passenger)
9	VehicleDeviation	Number		No		Current deviation of the vehicle in seconds
10	VehicleCurrentStopCode	String		No	X	The current stop where the vehicle drives to, halts or departs from
11	VehicleNextStopCode	String		No	X	The next stop of the vehicle's route
12	AdditionalInformation	String	A String containing key-	No		Additional information

	value-pairs separated by ';'	formatted as key-value pairs.

Table 10, Vehicle location message Array Return Fields

1.6.2.10 BASE VERSION INFORMATION TABLE

Sequence Nr	Field	JSON-Type	Valid Values	Comment
0	ResponseType	Number	3	
1	Version	String		

If the BaseVersion request field is set, the server answers MUST contain a separate array containing just the base version. This base version array can appear anywhere. In stream mode this MUST be send out everytime when the base version changes.

The client MUST NOT make assumptions about the format of the name.

1.6.2.11 URA VERSION INFORMATION TABLE

Sequence Nr	Field	JSON-Type	Valid Values	Comment
0	ResponseType	Number	4	
1	Version	String	Integer.Integer	e.g. "1.1" or "3.0"
2	TimeStamp	Number	Positive integers	Current UTC as per Unix Epoch
3	SupportedFields	Array of Strings	All API field names	Returns all fields where the server expects to return data

The server MUST send this field back as the first array in the response. It is send exactly once. The SupportedFields field needs to be enabled via ReturnList query parameter, otherwise it is omitted.

The TimeStamp can be used as a very simplistic time synchronisation for the client. (Server) Clients SHOULD rather use something like NTP.

The SupportedFields Array returns a list of API field names where the servers datasource SHOULD provide meaningful information. The server still MUST support queries for all fields, i.e. inclusive the ones not returned in the SuportedField list. However these fields might likely return null.

1.6.3 Stop Information

If the client does a request that does not contain a field from both Table 4 and Table 5, i.e. no prediction or flexible message only values, the server will return stop information arrays (message type 0) that only contain stops information. This response will also contain data for stops that do not have predictions or flexible messages. This allows retrieving all stops

for example, even if they are not served by vehicles. In order to also get those values in other requests, the StopArray Parameter can be used.

1.6.4 Response Type

The ResponseType field is:

0 for Stop Information

1 for Prediction

2 for Flexible message

3 for Base Version

4 for URA Version

5 for Pattern and Geometry

6 for Trips

7 for Trip Times

8 for Vehicle Information

9 for Vehicle Location

1.6.5 Comments

Note that by leaving out filter criteria the client can get the full set of data. E.g. leaving out the message type field will return all message types.

The server operators should make a public note which services require authorization and other constraints for the request configured. This information cannot be transferred on protocol level.

The meaning of the different stop codes is not part of this API and needs to be announced separately.

Any return may contain null response fields if the server does not have the required information available.

Whitespace-only data may be sent by the server as a heartbeat. The client MUST ignore any whitespace-only transmission.

When returning a limited response, the server SHOULD sort prediction arrays by the estimated time, returning only the most recent ones. Not that when doing subsequent limit requests, predictions might therefore drop out of the responses if the vehicle is delayed. For all other limited arrays no order is defined.

1.6.6 Constraints:

The server MAY NOT support the “If-Modified-Since” HTTP feature.

1.6.7 Stop Point State

The stop point state is transferred as a number, the values 0-3 are defined for version 1.0.Values 4-100 may be used in later versions others are up to the implementer to define:

0: “Open”,

1: “Temporary Closed”,

2: “Closed”,

3: “Suspended”

4-100: Reserved for Later API Versions

100-: Free to use by implementator

1.6.8 Info Message Type

The type of the message usually reflects the physical layout on devices. It is transferred as a number, the values 0-2 are defined for version 1.0. Values 3-100 may be used in later versions, and others are up to the implementer to define:

0: “Normal”,

1: “Special”,

2: “Full Matrix”,

3-100: Reserverd for Later API Versions

100-: Free to use by implementator

1.7 VERSIONING CONSIDERATION

The robustness principle from RFC 791 applies here as well. An increase in the minor version indicates new functions on the server side, but queries against older versions must return the same responses, i.e. a Client using URA V1.1 must get the same answer from a URA V1.2 or URA V1.3 server. An increase in the major version indicates a non-compatible change in the protocol.

1.8 AUTHORIZATION

Authorization of clients happens according to RFC 2617 digest authentication, or via an API key.

1.9 EXAMPLES (TO BE UPDATED LATER)

Note that in order to improve readability, whitespace and line breaks have been used in the examples below. The response is still valid JSON; however the client MUST NOT make assumptions about the use of whitespace, as by RFC.

1.9.1 Single Stop Full Data Example

This fetches all available data for a single stop.

1.9.1.1 QUERY STRING:

http://realtime.ivu.de:8098/instant?StopID=99&ReturnList=StopPointName,StopID,StopCode1,VisitNumber,LineID,LineName,DirectionID,DestinationText,DestinationName,EstimatedTime,MessageText,MessageType,Latitude,Longitude,ExpireTime

1.9.1.2 CONTENT RESPONSE:

[4,"1.0",12345242421]

[1, "GreenParkStation", "99", "21961", 51.12144141, 0.12141, "1", 332, "332", 2, "BrentPark", "BrentParkTesco", 1308847650, 1308857650]
[2, "GreenParkStation", "99", "21961", 51.12144141, 0.12141,0, ”Test flexible message”, 1308857870]

1.9.2 Full Prediction Stream Example

This fetches all predictions, stop information and Base Version for all stops

1.9.2.1 QUERY:

http://realtime.ivu.de:8098/stream?ReturnList=StopPointName,StopID,StopCode1,StopPointState,VisitNumber,LineID,LineName,DirectionID,DestinationText,DestinationName,EstimatedTime,Latitude,Longitude,ExpireTime,BaseVersion

Query Header containing Authorization data.

1.9.2.2 RESPONSE:

[4,"1.0",12345242421]

[3, “BaseVersion”, ”20100730”]
[1, "GreenParkStation", "99", "21961",0,51.12144141, 0.12141, "1", 332, "332", 2, "BrentPark", "BrentParkTesco", 1308847650, 1308857650] [1, "GreenParkStation", "99", "21961",0, 51.12144141, 0.12141, "1", 332, "332", 2, "BrentPark", "BrentParkTesco", 13088476120, 1308857650]....

1.9.3 Typical Mobile Example:

This fetches a limited amount of prediction data for an area around coordinates, as might be typically used by a mobile app.

1.9.3.1 QUERY:

http://realtime.ivu.de:8098/instant?ReturnList=StopPointName,LineName,DestinationName,EstimatedTime,Latitude,Longitude,ExpireTime&Circle=51.12144141,0.12141,500

1.9.3.2 RESPONSE:

[4,"1.0",12345242421]

[1, "GreenParkStation", 51.12144141, 0.12141, "332", "BrentParkTesco", 1308847650]
[1, "Tottenham", 51.12144141, 0.12141123, "331", "Aldwych", 1308847612, 1308857650]
…

1.9.4 Typical Corporate Sign Example

This fetches prediction data limited to the data needed for a corporate sign for a list of stops, identified by their StopCode.

1.9.4.1 QUERY:

http://realtime.ivu.de:8098/instant?ReturnList=StopPointName,StopPointState,LineName,DestinationName,EstimatedTime,ExpireTime&StopID=99,102,BS121

1.9.4.2 RESPONSE:

[4,"1.0",12345242421]

[1, "GreenParkStation",1, "332", "BrentParkTesco", 1308847650, 1308857650]
[1, "Tottenham",0, "331", "Aldwych", "1308847612", "1308847612"]

1.9.5 Full Stream Example

This fetches all available information, suitable to feed another URA server

1.9.5.1 QUERY:

http://realtime.ivu.de:8098/stream?ReturnList=StopPointName,StopID,StopCode1,StopPointState,VisitNumber,LineID,LineName,DirectionID,DestinationText,DestinationName,EstimatedTime,Latitude,Longitude,ExpireTime,BaseVersion,MessageUUID,MessageType,MessageText&Stream=true&StopAlso=true

Query Header containing Authorization data.

1.9.5.2 RESPONSE:

[4, "1.0",12345242421]

[3, "BaseVersion", "20100730"]

[0, "GreenParkStation", "99", "21961",0, 51.12144141, 0.12141]
[1, "GreenParkStation", "99", "21961",0, 51.12144141, 0.12141, "1", 332, "332", 2, "BrentPark", "BrentParkTesco", 1308847650, 1308857650]
[1, "GreenParkStation", "99", "21961",0, 51.12144141, 0.12141, "1", 332, "332", 2, "BrentPark", "BrentParkTesco", 13088476120, 1308857650]
[2, "GreenParkStation", "99", "21961",0, 51.12144141, 0.12141, ”12345-abcde”, 1, ”Test flexible message”, 1308857650]

....