Transport for London

TfL Live Bus & River Bus Arrivals API

Interface Documentation

Transport for London
50 Victoria Street
Westminster
London
SW1H 0TL

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

Please pass any queries to developers@tfl.gov.uk with *Bus & River Bus Arrivals API Feedback* in the subject line

Table of Contents

Document history				
Document Version	Software Version	Date	Name	Revision Type
1.0	1.0	05/04/2012	TS	Version for issue
1.1	1.0	03/05/2012	TS	Updates made following feedback from Beta
1.2	1.0	30/05/2012	TS	Updates made following internal feedback
1.3	1.0	12/10/2012	AR	Guidelines added to the fields 'DestinationName' and 'RegistrationNumber' in chapter 6.
1.4	1.0	28/03/2013	TS	Added River Bus information. Updated stop types to be displayed. Clarify use of StartTime for flexible messages.
1.5	1.0	02/08/2013	AR	Section 3.5: Information added for time synchronisation and time stamps. Section 3.6: Note on Pier stops added. Section 4: Technical detail added to query parameters. Section 4.1.1: Stream parameter removed from table. Section 4.1.3: Guidelines added for stream connectivy. Section 4.2.1 & Appendix A: Null values related to stop properties explained. Section 6: Note regarding not using 'DestinationName' removed. The long destination name can now be used, Section 6: Note added that latitude/longitude may be returned in scientific (E) notation format. Appendix B: New appendix added detailing Name, VehicleID and RegistrationNumber of all River Buses. Appendix C: New appendix added as guideline to flexible message priorities.
1.6	1.0	08/01/2015	AR	Section 4.1.2: Corrected 'ExpectedTime' and changed it to 'EstimatedTime'. Section 4.1.3: Added information about Digest Authentication RFC2617 for the Stream Interface. Section 4.2.1: Note regarding usage of 'StopPointState'. Section 5.2.1: Correction to header text. Section 6: Clarification regarding the TripID. General: Clarified that responses deviate from JSON.

2.0	1.0	28/07/2016	KC	Section 4.2.2: Added note about ‘MessageType’ = 2. Section 6: Added note about ‘MessageType’ = 2. Section 6: Removed description about SMS code to NaPTAN.
2.1	1.0	05/08/2016	KC	Section 6: Added note about 'ExpireTime' and 'TimeBeforeClear'

1 Introduction

1.1 The Live Bus & River Bus Arrivals API

TfL's Countdown system provides real-time bus and river bus arrival and service disruption information for passengers across London. Using data from iBus (https://tfl.gov.uk/corporate/about-tfl/what-we-do/buses), the system provides passengers with an accurate and complete information service for every one of London's 19,000 bus stops as well as for the TfL river bus piers. This data is currently provided over the web, via SMS and also on on-street signs.

A key part of TfL's strategy is to provide data openly. This API is designed to enable application developers to subscribe to live bus information and use this data to develop innovative services appropriate to their market and clients.

The data that is accessible via the API is taken from the same source systems as TfL's Countdown service. This ensures that the information supplied is consistent with other delivery channels, meaning that the end user is presented with consistent information.

1.2 Accessing the API

In order to access the API it is necessary to register at the TfL developer's area – http://www.tfl.gov.uk/developers

1.3 Attribution

Please do not include any TfL branding in your application or give the impression it is an official TfL application. Please add the attribution "Data provided by Transport for London" refer to https://tfl.gov.uk/corporate/terms-and-conditions/live-bus-departure-information.

1.4 Support

SLAs or guaranteed support are not offered with this API.

1.5 API Overview

The TfL Live Bus & River Bus Arrivals API is controlled via a number of different HTTP requests and parameters.

The API is based on JSON, however the responses deviate away from the JSON standard. This is primarily to optimise the performance of the API. It also allows the Streaming API to be consistent with the Instant API. This document describes and provides examples of the output format in order for developers to utilise the data.

A request is structured as follows:

```
http://server/virtualDirectory/type/version?HTTP parameters
```

2 Data served by the API

The data that is made available by this interface can broadly be put into two categories; real time data and reference data. By providing both categories of data over a single interface, developers are able to request real time data alongside the contextualising reference data in a single request. Data provided is stop centric; therefore the stop information is central to the request.

The scale of the bus network means that there is a large amount of data that can be accessed using this API. Some indicative figures to give an idea of the magnitude of the data are as follows:

- Over **19,000** bus stops
- Over **700** bus routes
- Over **8000** buses
- Approximately **130,000** bus arrival predictions at any point in time

In addition to the bus network data, the TfL River Bus network is also included in the API. This data is modelled in the same way as the bus data and therefore is provided in the same API.

Definitions of all data fields are provided in section 6.

2.1 Real time data

Live Bus & River Bus Arrivals provides the predicted time until a bus or river bus is expected to arrive at a stop. The system provides live bus/river bus arrival times, including destinations, for the next 30 minutes. This data is refreshed at source (tfl.gov.uk/countdown) every 30 seconds. It is therefore unnecessary to request this data at any interval more frequent than this.

Flexible messages provide service information and are assigned to specific bus stops. They are updated much less frequently than bus arrival information although may be added or removed at any time. It should be noted that flexible messages have both a start time and an expiry time. For operational reasons, messages are often added to the system prior to them being valid. It is therefore important that they are only displayed during the specified time window.

Stop closure information is the least variable of the real time data. Like service information messages, these closures may happen at any time.

2.2 Reference data

Reference data is primarily provided by the interface in order to give context to the real time data supplied. Additionally, developers have the option of requesting only reference data items and not real-time information.

Reference data is also sourced from TfL's Countdown system, this data is updated on a fortnightly basis in order to ensure that the data remains consistent with the rapidly changing London bus network. Reference data is versioned using a Baseversion, which is available via the API. Additional reference data, including bus timetables is available on the TfL developer area (http://www.tfl.gov.uk/developers).

Section 6 - Glossary of terms indicates whether a data item is reference data or real time data.

3 Service Behaviour

3.1 Instant and Streaming requests

The service provides two types of request to users; instant (request/response) and streaming.

Instant requests are made by the client and the server will respond with a single message. After this the client will not receive any further updates unless they make further requests.

Streaming requests are made by the client, in response the server will continually serve data to satisfy the request until the connection is terminated. Streaming data requires additional authentication owing to the potentially very large amounts of data to be transferred.

The data source for both instant and streaming requests is consistent, ensuring that the data provided to the public remains the same.

3.2 Caching

Data is cached in the system for a period of **30 seconds**. Hence there is no benefit to the developer in querying any of the data services any more frequently than once every 30 seconds.

3.3 URA Versioning

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.

3.4 HTTP status codes

In normal use, the service will provide HTTP status codes as per RFC 2616. The use of these codes is as follows:

HTTP Code	Reason
200 OK	Service is working correctly and the response contains data.
400 Bad Syntax	If the URL is malformed, such as it is not in exact conformance with this document, then the HTTP code of 400 is returned. The client should not retry and needs to change the request
401 Unauthorized	The Client did not provide, or provided incorrect, authentication details (username and password) and therefore the request was not served. Authentication is required in order to stream data.
408 Request Timeout	The server timed out waiting for the request.
416 Requested Range Not Satisfiable	The Client used a filter criteria used was not within range.

	The filter criteria used may not exist in the static data, or may be formally out of range.
500 Internal Server Error	A generic error message, given when no more specific messages are suitable.
502 Bad Gateway	There is an error in the upstream data supply or the supply is not available.

Table 1 - HTTP Status Codes

3.5 Time Synchronisation and Time Stamps

For general server time synchronisation it is recommended to use NTP and a reliable time source e.g. an internet based time source. Correct server time is mainly relevant to the Stream interface and when calculating and providing predictions as relative time i.e. the bus will arrive in 7 minutes.

The API always provides a timestamp for when the response to the request was processed. This is always in the first line of the response i.e. the 'URA Version Request' array see section 4.2.5. It is recommended to use this time stamp if calculating the relative time for predictions. However this is only practical when using the Instant interface as each response comes with it. The only option to calculate relative time when using the Stream is to use the local server time as the timestamp is only provided once when connecting to the Stream.

All API time references are provided as absolute time in Unix epoch in milliseconds (UTC).

3.6 River Buses and Pier Stops

Please note that some stops belonging to the same Pier have exactly the same location coordinates. When those stops are displayed on a map they will be shown on top of each other.

4 Technical guide to using the service

This guide relates to version 1.0 of the API.

The service is accessed by requesting a specific URL. This URL instructs the server as to what data should be returned. 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 to enable 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 it and it was specified in the HTTP request.

The server responds with a UTF-8 JSON like message. Depending on the request, the response can be composed of 5 different array types; Stop, Prediction, Flexible Message, Baseversion or URA Version arrays. Full details of the response format and these arrays are detailed in section 4.2 of this document.

4.1 Request

There are two different URLs for the service, one is for instant (request/response) usage and the other is for data streaming. The API for these services is the same; with the exception of deletion messages which are only necessary when streaming.

In order to obtain the URLs for the API it is necessary to register at the TfL developer's area – http://www.tfl.gov.uk/developers

As part of a request it is necessary to specify the parameters for the service. In specifying this string, the client instructs the server as to what data items should be returned and also provides the filters to restrict the data that is returned. Parameters should be added as comma separated values.

The full list of parameters that can be set are listed below.

4.1.1 Request parameters

Parameter	Type	Valid Values	Default	Multiple Values Allowed	Description
StopAlso	Boolean	True/False	False	No	If set to true, the service will return Stop arrays (Type 0) for stops even if they do not have predictions or flexible messages currently associated with them.

					This will only work for stop based requests. It will not work when selecting based on a route (LineID, LineName)
ReturnList	Comma-separated Strings	StopPointName,StopID,StopCode1,StopCode2, StopPointState, StopPointType, StopPointIndicator, Towards, Bearing, Latitude,Longitude,VisitNumber, TripID, VehicleID, RegistrationNumber,LineID,LineName,DirectionID,DestinationText,DestinationName,EstimatedTime, MessageUUID, MessageText,MessageType,MessagePriority, StartTime,ExpireTime, BaseVersion	StopPointName, LineName, EstimatedTime	No	List of fields to return to client.
Circle	Comma-separated Strings	GPS Coordinates plus radius, Format is Circle=Latitude, Longitude,Radius (in m),e.g. Circle=12.3121412,14.1231241,100	Unset	No	Filter response for stops within a given radius (in meters) from a specified latitude and longitude. (WGS84 coordinate system)
StopPointName	Comma-separated Strings	Every String	Unset	Yes	Filters response for only stops with that name
StopID	Comma-separated Strings	Every String	Unset	Yes	Filters response for only stops with that StopID

StopCode1	Comma-separated Strings	Every String	Unset	Yes	Filters response for only stops with that Stop Code 1
StopCode2	Comma-separated Strings	Every String	Unset	Yes	Filters response for only stops with that Stop Code 2
StopPointType	Comma-separated Strings	Every String	Unset	Yes	Filters response for only stops with that StopPointType
Towards	Comma-separated Strings	Every String	Unset	Yes	Filters response for only stops with the specific Towards value
Bearing	Comma-separated Numbers	Integer from 0…359	Unset	Yes	Filters response for only stops with the specific Bearing value
StopPointState	Comma-separated Numbers	Positive Integer	Unset	Yes	Filters response for only stops with the specific StopPointState.
VisitNumber	Comma-separated Numbers	Positive Integer	Unset	Yes	Filters response for only predictions with that sequence counter value
LineID	Comma-separated Numbers	Every String	Unset	Yes	Filters response for predictions only with that Line ID
LineName	Comma-separated Numbers	Every String	Unset	Yes	Filters response for predictions only with that Line Text
DirectionID	Number	1 or 2	Unset	No	Filters response for predictions only with that Direction
DestinationText	Comma-separated Strings	Every String	Unset	Yes	Filters response for only predictions with that Destination Text
DestinationName	Comma-separated Strings	Every String	Unset	Yes	Filters response for only predictions with that Destination Short Text
VehicleID	Comma-separated Strings	Every String	Unset	Yes	Filters response for only predictions for a specific Vehicle
TripID	Comma-separated	Positive	Unset	Yes	Filters response for only predictions for a specific

	Numbers	Integers			Journey
Registration Number	Comma-separated Strings	Every String	Unset	Yes	Filters response for only predictions for a vehicle with the specific Registration Number
StopPointIndicator	Comma-separated Strings	Every String	Unset	Yes	Filters response for stops with the specific Stop Indicator
MessageType	Comma-separated Numbers	Positive Integer	Unset	Yes	Filters response for only Message with that type.
MessagePriority	Comma-separated Numbers	Positive Integer 1..10	Unset	Yes	Filters response for Message with that Priority

Table 2 - Request Parameters

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

- Escape'&' by '\a'
- Escape',' by '\c'
- Escape result of string as JSON String by RFC 4627
- Percent-encoded result as URL String by RFC 3986

4.1.2 General Request Guidelines

- Clients should reduce the ReturnList to the absolute minimum in order to reduce load on both the client and server
- The client should not request MessageType or MessagePriority unless MessageText is part of the ReturnList
- The client should request the ExpireTime when requesting either the EstimatedTime OR MessageText fields
- The client should request the StartTime when requesting the MessageText field
- Leaving out filter criteria means that the client will get the full set of data. E.g. leaving out the message field will return all message types
- Filter criteria are not case sensitive
- The “If-Modified-Since” HTTP feature is not supported

4.1.3 Stream Interface Guidelines

Using the Streaming API means making a very long lived HTTP request. On connection, all current data matching the request will be made available immediately. After that only changes to the data will be returned as they occur.

To connect to the Streaming API, form a HTTP request and consume the resulting Stream for as long as is practical. The API service will hold the connection open indefinitely, barring server-side error, excessive client-side lag, network issues and planned system maintenance. To authenticate it is required to use Digest Authentication according to RFC2617.

The method to form an HTTP request and parse the response will be different for every language or framework, so consult the documentation for the HTTP library you are using. You must use an HTTP client that will return response data incrementally.

Once an established connection drops, attempt to reconnect immediately. If the reconnect fails, slow down your reconnect attempts according to the type of error experienced as recommended below,

- Back off linearly for TCP/IP level network errors. These problems are generally temporary and tend to clear quickly. Increase the delay in reconnects by 250ms each attempt, up to 30 seconds.

- Back off exponentially for HTTP errors for which reconnecting would be appropriate. Start with a 5 second wait, doubling each attempt, up to 320 seconds.

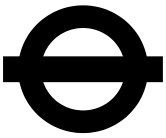
Repeatedly connecting and disconnecting to the Stream API wastes service resources and uses more data. Keep connections as stable and long-lived as possible.

Make sure to test your back off strategy. It is recommended that you get alerted when a back off situation occurs.

Also note that depending on the request made, data updates may not be available for 60 seconds or longer. If no updates are available white space only data will be sent by the server as a heartbeat every 60 seconds. The client must ignore any whitespace only transmission.

4.2 Response

The server responds with a UTF-8 JSON like message. Depending on the request, the response can be composed of 5 different array types; Stop, Prediction, Flexible Message, Baseversion or URA Version arrays. The sequence of these arrays in the response is undefined, except that the URA Version array always appears first.



An ‘empty’ response will contain only a URA Version array.

The sequence of fields returned within each response array will always follow the sequence detailed below, regardless of the order specified in the request. If a field is not specified in the ReturnList of the request then it is skipped.

In responses to Stream requests, white space only data may be sent by the server as a heartbeat. The client MUST ignore any whitespace only transmission.

4.2.1 Stop array

The stop array contains reference data about bus stops and piers. It does not contain any real time data. This array is particularly useful if real time data is not required. It is also used in order to return data for a stop when a stop does not have any predictions or flexible messages associated with it (the StopAlso request parameter should be set to ‘true’ to return these arrays).This will only work for stop based requests. It will not work when selecting based on a route (LineID, LineName)

Note that some fields may return ‘null’ values. In particular if the following fields contain ‘null’ values ‘StopCode1’, ‘StopCode2’, ‘StopPointType’ and ‘StopPointIndicator’ it usually means that the stop is a withdrawn stop or a bus stand (where the bus may stand and wait when not in service). For public display of stops it is important to follow the guideline in Appendix A.

Note also that the ‘StopPointState’ field is currently not part of any standard operational procedure. Instead the state of a stop is indicated using only the Flexible Message Array and the ‘MessageType’ field. If 'Message Type' field equals '2' the associated stop is temporarily out of service and predictions should not be presented to the customer.

The primary key of the Stop array is StopID.

Sequence Nr	Field	JSON-Type	Valid Values	Stop Primary Key
0	ResponseType	Number	For the Stop array this is always 0	No
1	StopPointName	String	As per Static Data	No
2	StopID	String	As per Static Data	Yes
3	StopCode1	String	As per Static Data	No
4	StopCode2	String	As per Static Data	No
5	StopPointType	String	As per Static Data	No
6	Towards	String	As per Static Data	No
7	Bearing	Number	As per Static Data, Integer 0…359	No

			degrees	
8	StopPointIndicator	String	As per Static Data	No
9	StopPointState	Number	Integer >= 0	No
10	Latitude	Number	Every Number	No
11	Longitude	Number	Every Number	No

Table 3 - Stop array

4.2.2 Prediction array

The prediction array contains the predicted arrival times for particular buses / river buses at stops. The array also contains the reference data for the routes, stops and vehicles that appear in the predictions.

The prediction array has a compound primary key comprised of StopID, VisitNumber, DestinationText and VehicleID.

Sequence Nr	Field	JSON-Type	Valid Values	Prediction Primary Key
0	ResponseType	Number	For the Prediction array this is always 1	No
1	StopPointName	String	As per Static Data	No
2	StopID	String	As per Static Data	Yes
3	StopCode1	String	As per Static Data	No
4	StopCode2	String	As per Static Data	No
5	StopPointType	String	As per Static Data	No
6	Towards	String	As per Static Data	No
7	Bearing	Number	As per Static Data, Integer 0…359 degrees	No
8	StopPointIndicator	String	As per Static Data	No
9	StopPointState	Number	Integer >= 0	No
10	Latitude	Number	Every Number	No
11	Longitude	Number	Every Number	No

12	VisitNumber	Number	Every positive Integer	Yes
13	LineID	String	As per Static Data	No
14	LineName	String	As per Static Data	No
15	DirectionID	Number	1 or 2	No
16	DestinationText	String	As per Static Data	Yes
17	DestinationName	String	As per Static Data	No
18	VehicleID	String	Every String	Yes
19	TripID	Number	As per Static Data	No
20	RegistrationNumber	String	As per Static Data	No
21	EstimatedTime	Number	Every positive Integer (UTC as per Unix epoch in milliseconds). To convert to UTC Epoch divide by 1000. 64 bit.	No
22	ExpireTime	Number	Every positive Integer (UTC as per Unix epoch in milliseconds). To convert to UTC Epoch divide by 1000. 64 bit. This field should always be requested if the client is requesting EstimateTime. In stream mode, this field will be set to 0 when the prediction needs to be deleted.	No

Table 4 - Prediction array

4.2.3 Flexible Message array

The flexible message array returns flexible messages that are associated with bus stops or piers. These messages inform passengers about incidents and service disruptions relevant to their journey. As with the prediction array, the flexible message array also contains reference data.

The flexible message array has a compound primary key consisting of StopID and MessageUUID.

Sequence Nr	Field	JSON-Type	Valid Values	FLM Primary Key
0	ResponseType	Number	2	No
1	StopPointName	String	As per Static Data	No
2	StopID	String	As per Static Data	Yes
3	StopCode1	String	As per Static Data	No
4	StopCode2	String	As per Static Data	No
5	StopPointType	String	As per Static Data	No
6	Towards	String	As per Static Data	No
7	Bearing	Number	As per Static Data, Integer 0…359 degrees	No
8	StopPointIndicator	String	As per Static Data	No
9	StopPointState	Number	Integer >= 0	No
10	Latitude	Number	Every Number	No
11	Longitude	Number	Every Number	No
12	MessageUUID	String	Every String	Yes
13	MessageType	Number	Integer >= 0	No
14	MessagePriority	Number	Integer 1...10 (where 1 is the highest priority). Refer to Appendix C for guidance.	No
15	MessageText	String	Any String	No
16	StartTime	Number	Every positive Integer (UTC as per Unix epoch in milliseconds). To convert to UTC Epoch divide by 1000. 64 bit.	
17	ExpireTime	Number	Every positive Integer (UTC as per Unix epoch in milliseconds). To convert to UTC Epoch divide by 1000. 64 bit. In stream mode, this field will be set to 0 when the flexible message	No

			needs to be deleted.	

Table 5 - Flexible Message array

4.2.4 Baseversion array

The Baseversion is used by LBSL to version static data. Where appropriate, it should be used when trying to join the Live Bus & River Bus Arrivals data with other LBSL data sets. Section 6 ‘Glossary of terms‘ details which data items are Baseversion controlled. These items can only change with a change of Baseversion, therefore clients may wish to use the Baseversion as an indicator as to whether they need to update this data item.

If the Baseversion request field is set, the server answers will always contain a separate array containing just the Baseversion. In stream mode this array will be send out when the Baseversion changes.

The client must not make assumptions about the format of the name

Sequence Nr	Field	JSON-Type	Valid Values
0	ResponseType	Number	3
1	Version	String	As per Static Data

Table 6 - Baseversion array

4.2.5 URA Version array

The URA version array provides the version of the URA that is being used. The array will always be the first line in the response.

The timestamp provides the time that the response was processed. It is synchronised with other timestamps provided by the Countdown system. For streaming requests, this timestamp is only provided on connection to the stream.

Sequence Nr	Field	JSON-Type	Valid Values
0	ResponseType	Number	4
1	Version	String	Integer.Integer (e.g. “1.1”)
2	TimeStamp	Number	Every positive Integer (UTC as per Unix epoch in milliseconds). To convert to UTC Epoch divide by 1000. 64 bit.

Table 7 - URA version array

5 Examples of Usage

The examples provided below are for guidance only. They provide guides as to the types of the request that can be made using this API.

Note that in order to improve readability, whitespace and line breaks have been used in the examples below. It should not be assumed that the response provided by the server will conform exactly to this format.

5.1 Instant request examples

5.1.1 Request predictions and flexible messages for a single stop

This request is for prediction and flexible message information for a single stop. This may either be a bus stop or a pier. This sort of request is likely to be used by mobile applications in order to display real-time data for a user selected stop.

Request

/interfaces/ura/instant?StopCode1=52053&DirectionID=1&VisitNumber=1&ReturnList=StopCode1,StopPointName,LineName,DestinationText,EstimatedTime,MessageUUID,MessageText,MessagePriority,MessageType,ExpireTime

Example Content Response

[4,"1.0",1334925465143] - URA Version array.

[1,"Green Park Station","52053","22","Piccadilly Cir",1334925458000,1334927227146] - first Prediction array

[1,"Green Park Station","52053","14","Warren Street",1334925830000,1334927247004]

[1,"Green Park Station","52053","22","Piccadilly Cir",1334925731000,1334926994196]

...

[1,"Green Park Station","52053","14","Warren Street",1334926824000,1334926832021]

[1,"Green Park Station","52053","22","Piccadilly Cir",1334926836000,1334926844473]

[1,"Green Park Station","52053","14","Warren Street",1334927168000,1334927176525]

[[2,"Green Park Station","21961","8a56a2ac359ff7df0136074830af4b26_99",0,3,"Test message for Green Park",1333545681000] –first Flexible Message array

5.1.2 Request predictions for multiple stops

This request is for predictions for two routes at multiple bus stops. This sort of request is likely to be used to allow a passenger that has a choice of stops that serve their destination to determine which to depart from.

Request

/interfaces/ura/instant?StopCode1=58726,51586&LineName=c10,507&ReturnList=StopPointName,LineName,DestinationText,EstimatedTime,ExpireTime

Example Content Response

[4,"1.0",1334928941477] - URA Version array.

[1,"Marsham Street","507","Victoria",1334929424000,1334930553134] - first Prediction array

[1,"Marsham Street","507","Victoria",1334929784000,1334930548807]

[1,"Marsham Street","507","Victoria",1334928961000,1334930715448]

[1,"Marsham Street","507","Victoria",1334930151000,1334930544495]

[1,"Marsham Street","507","Victoria",1334930534000,1334930593771]

[1,"Page Street","C10","Victoria",1334929908000,1334930534667]

[1,"Page Street","C10","Victoria",1334929194000,1334930548463]

5.1.3 Request stops within a specified radius

This request returns static data for all 'Open' bus stops within a radius of a specified point. This is likely to be of use when returning bus stops that are close to a mobile user's location.

Request

/interfaces/ura/instant?Circle=51.49598,-0.14191,250&StopPointState=0&ReturnList=StopCode1,StopPointName,Bearing,StopPointIndicator,StopPointType,Latitude,Longitude

Example Content Response

[4,"1.0",1334930109388] - URA Version array

[0,"Bressenden Place / Victoria Station","91545","STBC",165,"CN",51.497219,-0.141818] - first Stop array

[0,"Victoria Station, Bus Station Stand",null,null,null,null,51.496169,-0.143633]

[0,"Victoria Bus Station, Stand D",null,null,null,null,51.496103,-0.14401]

…

[0,"Victoria Station","56026","STBC",92,"F",51.496239,-0.143514]

[0,"Victoria Station","57096","STBC",338,"H",51.495648,-0.143106]

5.1.4 Request all predictions for a specific vehicle

This request returns prediction information for a specific vehicle. This is likely to be used in an application designed to allow a passenger to view the predicted arrival times (for the next 30 minutes) of their specific bus. This would give the user an indication of how long until they reach their destination.

Request

/interfaces/ura/instant?RegistrationNumber=LX59DDF&ReturnList=StopCode1,EstimatedTime,ExpireTime,Baseversion,RegistrationNumber

Example Content Response

[4,"1.0",1334932175872] - URA Version array

[3,"20120417"]– Baseversion array

[1,"52954","LX59DDF",1334933329000,1334933330949] – first Prediction array

[1,"52374","LX59DDF",1334933383000,1334933385304]

[1,"74720","LX59DDF",1334932474000,1334933795889]

[1,"49429","LX59DDF",1334932293000,1334933884537]

[1,"50476","LX59DDF",1334933045000,1334933053570]

5.2 Stream examples

5.2.1 Stream prediction data for two stops

This request is to return a stream of continuous data for two stops.

Request

/interfaces/ura/stream?Stopid=99,13551&ReturnList=Stoppointname,VehicleID,RegistrationNumber,LineName,DestinationName,EstimatedTime,ExpireTime

Example Content Response

[4,"1.0",1332280681000] - URA Version array.

[1,"Green Park Station","Z19","Finsbury Park Interchange",942,"LJ51DBV",1332280682000,1332280682000] –first Prediction array

[1,"Green Park Station","19","Finsbury Park Interchange",938,"X538GGO",1332280650000,1332280650000] –second Prediction array

…

[1,"Green Park Station","Z19","Finsbury Park Interchange",942,"LJ51DBV",1332280682000,1332280682000]

[1,"Green Park Station","19","Finsbury Park Interchange",938,"X538GGO",1332280650000,1332280650000]

[1,"Green Park Station","9","Aldwych",14811,"W141EON ",1332280976000,1332280976000]

[1,"Green Park Station","19","Finsbury Park Interchange",1578,"LF52URT,1332281123000,1332281123000]

5.2.2 Stream all data for all stops

This request is to return a stream of continuous data for all stops.

Request

/interfaces/ura/stream?ReturnList=StopPointName,StopID,StopCode1,StopCode2,StopPointState,StopPointType,StopPointIndicator,Towards,Bearing,Latitude,Longitude,VisitNumber,TripID,VehicleID,RegistrationNumber,LineID,LineName,DirectionID,DestinationText,DestinationName,EstimatedTime,MessageUUID,MessageText,MessageType,MessagePriority,ExpireTime,BaseVersion

Example Content Response

[4,"1.0",1332197936000] - URA Version array.

[3,"20111225"] - Baseversion array

[1,"Centre Common Road / War Memorial","17522","25633","490005205N","STBC","ELTHAM OR GROVE PARK",323,"N",0,51.413472,0.072582,1,"Z161","Z161",1,"North Greenwich","North Greenwich Station",18225,254131,"YN06JXY",1332197937000,1332197937000] – first Prediction array

[1,"Cedar Way","10107","16095","03700083","STBR","SLOUGH",303,null,0,51.497457,-0.556007,1,"81","81",2,"Slough","Slough Bus Station",14502,856858,"YN55NHA ",1332198209000,1332198209000] –second Prediction array

…

[1,"New Cross Gate Station","26343","22474","490000156O","STBC","PECKHAM OR BRICKLAYERS ARMS",246,"O",0,51.475048,-0.039426,1,"Z53","Z53",1,"Whitehall","Horse Guards Parade",21188,300000,"TEST_21188",1332197619000,1332197619000]

6 Glossary of terms

The API used is not specific to TfL. As a result, many of the names used are generic so that they can be populated with different data according to the transport authority using the service. For this reason it is necessary to provide a reference between the field names in this API and those used elsewhere by TfL. A description of each field is provided along with an indication whether the data item is reference or real time.

Reference data will not change within a particular Baseversion.

Field	TfL static data	Description	Data type
Bearing	Heading	Direction the vehicle is travelling in when it arrives at the bus-stop. This is expressed from 0° to 359°	Reference
DestinationName	Long_Destination_Name	The full length destination name of the trip the vehicle is on. The destination name is based on the route and end point of the trip.	Real time
DestinationText	Short_Destination_Name	The abbreviated destination name of the trip the vehicle is on. The destination text is based on the route and end point of the trip.	Real time
DirectionID	Direction	This identifies the direction of the trip that the vehicle is on. It indicates whether the vehicle is on an outbound or inbound trip.	Reference
EstimatedTime	N/A	This is the predicted time of arrival for the vehicle at a specific stop. It is an absolute time in UTC as per Unix epoch (in milliseconds). To convert to UTC Epoch divide by 1000.	Real time
ExpireTime	N/A	This is the time at which the corresponding prediction or flexible message is no longer valid and should stop being displayed. This will ensure consistency with the on street signs.It is an absolute time in UTC as per Unix epoch (in milliseconds). To convert to UTC Epoch divide by 1000. In stream mode, this time may be set to 0 to indicate that a message	Real time

		should be deleted. For predictions, 'ExpireTime' = 'EstimatedTime' + 'TimeBeforeClear' 'TimeBeforeClear' is a parameter that is set by TfL and can be unique for each stop. *N.B. as of Aug 2016, 'TimeBeforeClear' is set to 30 seconds for all stops.*	
Latitude	Location_Latitude	The latitude of the stop. This is expressed using the WGS84 coordinate system. Note that some values may be returned in scientific (E) notation format e.g. -1.87E-4.	Reference
LineID	Contract_Line_No	The identifier of a route. This is an internal identifier and is not equal to the route number displayed on the front of the bus / river bus. It should not be displayed to the public.	Reference
LineName	Service_Line_No	This is the route number that is displayed on the front of the bus / river bus and on any publicity advertising the route.	Reference
Longitude	Location_Longitude	The longitude of the stop. This is expressed using the WGS84 coordinate system. Note that some values may be returned in scientific (E) notation format e.g. -1.87E-4.	Reference
MessagePriority	N/A	Messages are assigned a priority in order for them to be ranked. Since it is possible for a stop to be assigned multiple messages it is important to ensure priority is given. Priorities are between 1 and 10 (where 1 is the highest priority). By default the message priority is set to 3.	Real time

MessageText	N/A	The text of the message. This should be displayed to the public.	Real time
MessageType	N/A	Messages are assigned a type. This is predominantly in order to define how they should be displayed on on-street signs, however can be used to alter display on other devices. 0: “Normal”, 1: “Special”, 2: “Full Matrix” – Stop is temporarily out of service and predictions should not be presented to the customer	Real time
MessageUUID	N/A	This is the unique identifier of the flexible message.	Real time
RegistrationNumber	N/A	The registration number of the vehicle that the prediction belongs to. Any registration numbers containing the following values should be excluded from any selection or publication. Prefixed with ‘X_’: These vehicles have been decommissioned from the London bus fleet. ‘NEW’ within the first five characters: These are placeholders for new vehicles soon to enter the London bus fleet. These should not appear in the prediction array as they are not active vehicles, if they do then please contact us via developers@tfl.gov.uk.	Reference
ResponseType	N/A	This indicates the response array type. There are 5 different response arrays: 0: “Stop array” 1: “Prediction array” 2: “Flexible Message array”	Reference

		3: “Baseversion array” 4: “URA Version array”	
StartTime	N/A	This is the time at which the flexible message becomes valid. It should not be displayed to the public before this time. It is an absolute time in UTC as per Unix epoch (in milliseconds). To convert to UTC Epoch divide by 1000.	Real time
StopID	Stop_Code_LBSL	This is the alphanumeric identifier of a bus stop or pier used by LBSL. It SHOULD NOT be displayed to the public.	Reference
StopCode1	SMS_Code	This is the public code for the bus stop or pier. It is displayed on the bus stop flag or pier, the Countdown website and should be used for all public facing applications.	Reference
StopCode2	NaPTAN_Code	This is the unique national identifier of the bus stop or pier – i.e. the NaPTAN AtcoCode as defined in the DfT NaPTAN data model.	Reference
StopPointIndicator	Point_Letter	The letter(s) that are displayed on top of the bus stop flag (e.g. SA). These are used to help passengers easily identify a bus stop or pier from others in the locality. It should be noted that not all bus stops or piers are assigned a point letter and that stop point indicators are not unique.	Reference
StopPointName	Stop_Name	The name of the bus stop / pier.	Reference
StopPointState	N/A	The different stop states and their definitions are provided below: 0: “Open”: Bus stop or pier is being served as usual	Real time

		1: “Temporarily Closed” : Vehicles are not serving the stop but may be serving a nearby bus stop or pier, predictions may be available 2: “Closed” : Vehicles are not serving the stop. Stop should display the closed message and **predictions should not be shown**. 3: “Suspended” : Vehicles are not serving the stop. Stop should display the closed message and **predictions should not be shown**. (On street signs may not show any messages in this scenario)	
StopPointType	Stop_Type	Indicates the type of stop as categorized by TfL. The full list of these stop point types is available in Appendix A.	Reference
Towards	Towards	Identifies the primary location(s) that are visited by routes serving a stop. The ‘towards’ text relates to the stop and, for buses, is displayed on the bus stop flag on the street.	Reference
TripID	Journey_Idx	The identifier of the specific trip that the prediction is for. Note that this is an internal iBus trip identifier. It does not correspond to the bus schedule trip identifier.	Reference
VehicleID	N/A	The unique identifier of the vehicle. This is an internal identifier and should not be displayed to the public.	Reference
Version		Either used as baseversion of the data (Baseversion array) or URA version (Version array)	Reference
VisitNumber	N/A	Indicates whether the prediction is for the first time the vehicle visits a stop on that trip. On some routes,	Reference

		where the vehicle ‘loops’ the same stop may be visited more than once.	

Table 8 - Glossary of terms

Appendix A. Stop Point Types

TfL bus stops are assigned different types depending on their usage. The table below provides a reference for the stop point types. It should be noted that the API includes stop types that **should not** be displayed to the public. These are intentionally left in the data as they help to indicate the roads that are served by the bus route.

The table indicates which stop types should NOT be displayed to the public. It is expected that users of the data adhere to this in order to avoid confusion to passengers.

In order to determine which bus stops are currently served by one or more routes the static data that TfL make available should be consulted.

Stop Point Type	Description	Public display?
null	Not a stop type but a null value.	No
STBR	Stop - Bus Request	Yes
STBC	Stop - Bus Compulsory	Yes
STZZ	Stop - Other	Yes
STBN	Stop - No Flag (Hail + Ride Time table)	Yes (note no physical stop at this location)
STBS	Stop - Live Bus Stand	Yes
STBE	Stop - Dead Bus Stand	No
STCC	Stop - Coach Compulsory	No
STTS	Stop - Taxi Stop	No
STSS	Stop - Bus Station Stop	Yes
STTP	Stop - Timing Point for CAESAR	No
STDM	Stop - Dummy (No Physical Stop)	No
STVA	Stop - Virtual - iBus Announce	Yes (note no physical stop at this location)
STCR	Stop - Coach Request	No
STDL	Stop - Bus Request Coach Req	No
STDJ	Stop - Bus Compulsory Coach Req	No
SHCP	Stop - Hail & Ride Comm Point Start	No
SHCE	Stop - Hail & Ride Comm Point End	No
SLRS	Stop London River Services	Yes. Should be used to provide River Bus information.

Table 9 - Stop Point Types

Appendix B. River Bus Information

The following tables are detailing information about the River Buses as not all are available in the API.

River Bus Name	VehicleID	RegistrationNumber
Sun Clipper	18951	1
Moon Clipper	18952	2
Sky Clipper	18953	3
Storm Clipper	18954	4
Star Clipper	18955	5
Meteor Clipper	18956	6
Cyclone Clipper	18957	7
Typhoon Clipper	18958	8
Tornado Clipper	18959	9
Monsoon Clipper	18960	10
Aurora Clipper	18961	11
Hurricane Clipper	18962	12
Twin Star	18963	13

Table 10 – River Bus Information

	Visible on Boat	Available in API
River Bus Name	Yes	No
VehicleID	No	Yes
RegistrationNumber	Yes	Yes

Table 11 – Location of River Bus information

Appendix C. Guide to Flexible Message Priorities

The following is a guideline to the Flexible Message priority field 'MessagePriority' which indicates the type and severity of an incident made available via a flexible message. It is recommended that all messages are displayed but the prominence of messages can be managed by priority. Note that priority 6 is not in use at the moment and priorities 7-10 may be included in the future. It is recommended that these priorities are still made available in a case message is given one of these priorities.

Real Time and Unplanned Incidents		
Priority	**Type of Incident**	**Examples and instructions**
1	**Emergency and network wide severe disruption** Bus and other modes	Severe weather conditions (no or very limited service) Advice to passengers during emergency incidents
2	**High severity - disruption affecting a large area of the network** Bus and other modes. Current events and future events but no earlier than 24 hours before the anticipated event **Unplanned failures to the Countdown system** **Scheduled system maintenance**	Bus operator strikes Widespread rail/LUL disruption – strikes, major line closures Future incidents: Severe advanced weather warnings expected to cause widespread disruption Other events expected to result in widespread network disruption, such as bus operator or rail strikes
3	**Major incidents – bus**	Severe delays Diversions (leaving stops unserved and/or causing significant disruption) Route suspensions and curtailments Stop closures
4	**Major incidents – other modes**	Service suspensions Part of full line closures Closure of major stations Severe delays if the resulting disruption is deemed to be widespread and long term

Table 12 - Real Time and Unplanned Incidents

Future Planned Incidents		
Priority	**Type of Incident**	**Examples**
5	**Planned bus** **Future scheduled system maintenance**	Events expected to lead to: Severe delays Diversions (leaving stops unserved or causing significant disruption) Route curtailments Stop closures (at affected stop only) Scheduled maintenance to Countdown which will affect information availability
6	**Planned other modes** **(Not in use at the moment)**	Part or full line closures Closure of major stations (Blanket general messages may be used to cover extensive weekend engineering work programmes.)
7–10	**(Not in use at the moment)**	

Table 13 – Future Planned Incidents