# Public XML Feed

Revision 1.24

May 18, 2021

**Revision History:**

| Revision | Date | Reason |
|---|---|---|
| 1.0 | Nov. 2, 2009 | Document Creation |
| 1.1 | Jan. 14, 2010 | Switched to webservices.nextbus.com |
| 1.2 | June 9, 2010 | Updated Terms & Conditions |
| 1.4 | July 13, 2010 | Clarified use of stopId versus stop tags for predictions. Also clarified the contents of the data that is returned for routeConfig command. |
| 1.5 | August 11, 2010 | Added information on restrictions to make sure bandwidth of data is limited. |
| 1.6 | December 15, 2010 | Clarification on using commands. |
| 1.7 | January 14, 2011 | Modified the direction tag variables in the requests for predictions using route, direction and stop tags. |
| 1.8 | January 21, 2011 | Added description of tripTag, affectedByLayover, and isScheduleBased attributes. |
| 1.9 | February 17, 2011 | Added description of delayed attribute. |
| 1.10 | March 4, 2011 | Changed definition of tripTag. |
| 1.11 | May 10, 2011 | Added information about e-mail groups for announcements and discussions. |
| 1.12 | June 6, 2011 | Added documentation for schedule and messages commands, cleaned up some inconsistencies in terminology, and change the font of the document. |
| 1.13 | June 16, 2011 | Added section on compression |
| 1.14 | July 11, 2011 | Added additional info for routeList command |
| 1.15 | July 13, 2011 | More explicit information on <predictions> data. |
| 1.16 | August 16, 2011 | Explanation of "_IB" and "_OB" suffixes for stops. |
| 1.17 | January 30, 2012 | Corrected URLs. |
| 1.18 | April 30, 2012 | Added branch attribute for Toronto TTC predictions. Fixed typo for predictionsForMultiStops command. Added information on attributes for the vehicleLocations command. |
| 1.19 | September 5, 2012 | Clarified for TTC that for arrival predictions are provided for streetcars at terminals while for buses departure predictions are provided. |
| 1.20 | September 11, 2012 | Added information on <prediction> attributes. |
| 1.21 | March 19, 2013 | Corrected description of dirTag for vehicleLocations command. |
| 1.23 | January 26, 2016 | Added description of lastTime element for vehicleLocations command. Update term and conditions. |
| 1.24 | May 18, 2021 | Switch to https://retro.umoiq.com |

## Terms & Conditions

The accompanying documentation and the XML Feed (Feed) is available under the terms of this LICENSE AGREEMENT (THE "AGREEMENT").

No support is to be provided by Umo IQ to external use of the Feed except for those under written contracts. Umo IQ reserves the right to terminate or limit the Feed to any or all users at any time and for any reason. For example, access to the XML Feed will be restricted if attempt to obtain data too frequently. All polling commands such as for obtaining vehicle locations should only be run at the most once every 10 seconds. Comments or suggestions with respect to the Feed can be sent to umoiq.support@cubic.com .

LICENSE GRANT.
1. The Umo IQ Incorporated grants its CUSTOMERS a non-exclusive license to use the XML Feed (Feed) and associated data. This Agreement will also apply to any upgrades provided by Umo IQ that replace and/or supplement the original Feed, unless the upgrades are under a separate license agreement, in which case the terms of the separate license will govern.

2. TERMINATION. If CUSTOMERS breach this Agreement the right to use the Feed will terminate immediately without notice, but all provisions of this Agreement except the License Grant (Paragraph 1) will survive termination and continue in effect.

3. PROPRIETARY RIGHTS. CUSTOMERS may not remove or alter any trademark, logo, copyright or other proprietary notices in or on the Feed. This license does not grant CUSTOMERS any right to use the trademarks, service marks or logos of Umo IQ or its licensors.

4. DISCLAIMER OF WARRANTY. THE FEED IS PROVIDED "AS IS" WITH ALL FAULTS. TO THE EXTENT PERMITTED BY LAW, UMO IQ AND UMO IQ' DISTRIBUTORS, LICENSORS HEREBY DISCLAIM ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION WARRANTIES THAT THE FEED IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE AND NON-INFRINGING. CUSTOMERS BEAR ENTIRE RISK IN USING THE FEED FOR ANY PURPOSES AND THE QUALITY AND THE PERFORMANCE OF THE APPLICATIONS OF THE FEED. THIS LIMITATION WILL APPLY NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES, THIS DISCLAIMER MAY NOT BE APPLICABLE.

5. LIMITATION OF LIABILITY. EXCEPT AS REQUIRED BY LAW, UMO IQ AND ITS DISTRIBUTORS, DIRECTORS, LICENSORS, CONTRIBUTORS AND AGENTS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES ARISING OUT OF OR IN ANY WAY RELATING TO THIS AGREEMENT OR THE USE OF OR INABILITY TO USE THE FEED, INCLUDING UNLIMITED DAMAGES CLAIMS FOR LOSS OF GOODWILL, WORK STOPPAGE, LOST PROFITS, LOSS OF DATA, AND COMPUTER FAILURE OR MALFUNCTION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND REGARDLESS OF THE THEORY (CONTRACT, TORT OR OTHERWISE) UPON WHICH SUCH CLAIM IS BASED. THE UMO IQ COLLECTIVE LIABILITY UNDER THIS AGREEMENT WILL NOT EXCEED THE GREATER OF $1 (ONE DOLLAR) AND THE FEES PAID BY THE CUSTOMERS UNDER THIS LICENSE (IF ANY). SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, THIS EXCLUSION AND LIMITATION MAY NOT BE APPLICABLE TO CUSTOMERS.

6. MISCELLANEOUS. (a) This Agreement constitutes the entire agreement between Umo IQ and CUSTOMERS concerning the subject matter hereof, and it may only be modified by a written amendment signed by an authorized executive of Umo IQ. (b) Except to the extent applicable law, if any, provides otherwise, this Agreement will be governed by the laws of the state of California, U.S.A., excluding its conflict of law provisions. (c) This Agreement will not be governed by the United Nations Convention on Contracts for the International Sale of Goods. (d) If any part of this Agreement is held invalid or

unenforceable, that part will be construed to reflect the parties' original intent, and the remaining portions will remain in full force and effect. (e) A waiver by either party of any term or condition of this Agreement or any breach thereof, in any one instance, will not waive such term or condition or any subsequent breach thereof. (f) Except as required by law, the controlling language of this Agreement is English. (g) You may assign your rights under this Agreement to any party that consents to, and agrees to be bound by, its terms; the Umo IQ Incorporated may assign its rights under this Agreement without condition. (h) This Agreement will be binding upon and will inure to the benefit of the parties, their successors and permitted assigns.

Note: The Feed is free-of-charge to Umo IQ’ respective CUSTOMERS

# Overview

Umo IQ Incorporated (Umo IQ) provides a Feed of the prediction and configuration information such that developers can create applications for providing passenger information to the public. The XML Feed data is accessed using URLs with parameters specified in the query string. This document provides:

- The URLs that can be used.
- Examples of the XML that is returned

Included are steps for accessing and utilizing information supplied within Umo IQ' XML feed. The XML feed data is accessed using URLs with parameters specified in the query string. This document provides the URLs that can be used and examples of the XML that is returned

Before proceeding you should have at least passing familiarity with XML and URI strings or RESTful web services.

For a brief introduction to XML please visit the sites listed below:
http://www.w3schools.com/xml/xml_whatis.asp
http://www.codeguru.com/java/article.php/c13529

For a brief introduction to RESTful web services please visit the sites listed below:
http://www.infoq.com/articles/rest-introduction
http://en.wikipedia.org/wiki/REST

## *Stability of XML Feed*

It is intended that the commands and the resulting XML data change as little as possible in the future. There will be situations though when the feed will need to change. In particular, additional XML elements and properties might be added. This should not affect a proper XML client though because any new elements should simply be ignored.

## *Compressing Data*

If the request for XML data specifies in the request header that gzip'ed data can be used then the web servers will automatically compress the XML data. This can be especially useful for Smartphone applications that use a slower cellular connection because reductions in data size of between 50% and 85% are attained.

Web browsers automatically specify that they can handle compressed data so if a data is requested via Javascript from a web browser then no additional action needs to be taken to make use of this feature. But if the application requests data using something like the wget command

the one needs to set the headers appropriately in order to receive compressed data. To do so include the following header in the requests:

```
Accept-Encoding:  gzip,  deflate
```

## *Commands*

The commands can be broken in several categories. There are configuration requests, prediction requests, a vehicle location request, and an arrival/departure times request. The arrival/depature request is not intended for the public so is not documented here.

Note: The <agency tag> used in the examples in the remainder of this document is sf-muni. To find the proper agency tag use the command
https://retro.umoiq.com/service/publicXMLFeed?command=agencyList

The format of URL command is:

```
https://retro.umoiq.com/service/publicXMLFeed?command=commandName&a=<agency
tag>&additionParams...
```

## *Error Messages*

Commands can return an Error XML object if there is some type of problem such as the system has not been initialized yet or bad parameters were used in the URL. When an Error XML object is returned, it has an attribute called shouldRetry. If the error was returned only because the server was initializing, then shouldRetry will be set to true. For this case the application should try the URL again after waiting 10 seconds. If shouldRetry attribute is set to false though, that means that there is an error due to the URL and simply retrying the URL again will not fix the problem.

An error message looks like:

```
<body>
   <Error shouldRetry="true">
    Agency server cannot accept client while status is: agency
      name = sf-muni,status = UNINITIALIZED, client count = 0, last
    even t = 0 seconds ago Could not get route list for agency tag "sf-muni".
    Either the route tag is bad or the system is initializing.
   </Error>
</body>
```

## *Limits on Amount of Data*

In order to prevent some users from being able to download so much data that it would interfere with other users we have imposed restrictions on data usage. These limitations could change at any time. They currently are:

- Maximum characters per requester for all commands (IP address): 2MB/20sec
- Maximum routes per "routeConfig" command: 100
- Maximum stops per route for the "predictionsForMultiStops" command: 150
- Maximum number of predictions per stop for prediction commands: 5
- Maximum timespan for "vehicleLocations" command: 5min

# Configuration Requests

## *Command "agencyList"*

To obtain a list of available agencies the following command should be used:

https://retro.umoiq.com/service/publicXMLFeed?command=agencyList

The resulting XML will be in the form (note: the shortTitle element is provided only if it is different than the standard title element. If no shortTitle element is provided, simply use the standard title element):

```
<body>
  <agency tag="actransit" title="AC Transit, CA"
regionTitle="California-Northern">
  <agency tag="alexandria" title="Alexandria DASH, VA"
shortTitle="DASH" regionTitle="Virginia">
  <agency tag="amerimar" title="Amerimar" regionTitle="Pennsylvania">
  <agency tag="blackhawk" title="Black Hawk Transportation Authority,
  CO" shortTitle="Black Hawk" regionTitle="Colorado">
  <agency tag="camarillo" title="Camarillo Area (CAT), CA"
shortTitle="Camarillo (CAT)" regionTitle="California-Southern">
  <agency tag="case-western" title="Case Western University, OH"
shortTitle="Case Western" regionTitle="Ohio">
  <agency tag="chapel-hill" title="Chapel Hill Transit, NC"
shortTitle="Chapel Hill" regionTitle="North Carolina">
  ...
</body>
```

**Note: For all commands below an agency tag, "a" should be included in the query string and set** to an agency name such as ***sf-muni*** (e.g., a=sf-muni).

## *Command "routeList"*

To obtain a list of routes for an agency, use the "routeList" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used.

The format of the command is:

https://retro.umoiq.com/service/publicXMLFeed?command=routeList&
a=<agency_tag>

The route list data returned has multiple attributes. These are:

- **tag – unique alphanumeric identifier for route, such as "N".**
- **title – the name of the route to be displayed in a User Interface, such as "N-Judah".**
- **shortTitle –** for some transit agencies shorter titles are provided that can be useful for User Interfaces where there is not much screen real estate, such as on smartphones. This element is only provided where a short title is actually available. If a short title is not available then the regular title element should be used.

For the example URL:
https://retro.umoiq.com/service/publicXMLFeed?command=routeList& a=sf-muni

The resulting XML is in the form:

```
<body>
  <route tag="1" title="1 - California" shortTitle="1-Calif/>
  <route tag="3" title="3 - Jackson" shortTitle="3-Jacksn"/>
  <route tag="4" title="4 - Sutter" shortTitle="4-Sutter"/>
  <route tag="5" title="5 - Fulton" shortTitle="5-Fulton"/>
  <route tag="6" title="6 - Parnassus" shortTitle="6-Parnas"/>
  <route tag="7" title="7 - Haight" shortTitle="7-Haight"/>
  <route tag="14" title="14 - Mission" shortTitle="14-Missn"/>
  <route tag="21" title="21 - Hayes" shortTitle="21-Hayes"/>
</body>
```

## *Command "routeConfig"*

To obtain a list of routes for an agency, use the "routeConfig" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used. The route is optionally specified by the "r" parameter. The tag for the route is obtained using the routeList command. If the "r" parameter is not specified, XML data for all routes for the agency is returned. Due to the large size of the resulting XML the routeConfig command is limited to providing data for only up to 100 routes per request. If an agency has more than 100 routes then multiple requests would need to be used to read data for all routes.

The format of the command is:

https://retro.umoiq.com/service/publicXMLFeed?command=routeConfi
g&a=<agency_tag>&r=<route tag>

The route data returned has multiple attributes. These are:

- **tag** – unique alphanumeric identifier for route, such as “N”.
- **title** – the name of the route to be displayed in a User Interface, such as “N-Judah”.
- **shortTitle** – for some transit agencies shorter titles are provided that can be useful for User Interfaces where there is not much screen real estate, such as on smartphones. This element is only provided where a short title is actually available. If a short title is not available then the regular title element should be used.
- **color** – the color in hexadecimal format associated with the route. Useful for User Interfaces such as maps.
- **oppositeColor** – the color that most contrasts with the route color. Specified in hexadecimal format. Useful for User Interfaces such as maps. Will be either black or white.
- **latMin**, **latMax**, **lonMin**, **lonMax** – specifies the extent of the route.

The route data returned includes lists of stops, lists of directions, and lists of paths. Configurations can be complicated, with multiple directions having different sets of stops. Therefore one cannot expect the data to contain just two simple directions for a route. The stops are provided to show the details, such as the titles for the stops, lat/lon, and also a numeric stop ID. If you are creating a map then you simply display all of the stops on the map. But if you are creating route/direction/stop selection User Interface you will need to use the direction data. The directions list all of the stops associated with a direction. If the direction is important enough to be

listed to a passenger then the “useForUI” element will be set to true. The other directions typically do not need to be shown and are therefore not provided by default. If you need the other non-useForUI directions then you need to add “&verbose” to the URL.

A stop has the following attributes:

- **tag** – unique alphanumeric identifier for stop, such as “cp_1321”. Even if the stop tags appear to usually be numeric they can sometimes contain alphabetical characters. Therefore the stop tags cannot be used as a number for telephone systems and other such applications. For larger agencies such as Toronto TTC suffixes "_IB" and “_OB" are included at the end of the stop tag for the rare situations when an agency has defined only a single stop for both directions and the stop is not an arrival at the end of the route (in cases of arrivals “_ar” is used). This means that the stop tag might not always correspond to GTFS or other configuration data. These suffixes allow duplicate stops to have the identical stopID as the original stop while preserving both unique stops in the system. "_IB" represents a duplicated inbound stop, and "_OB" represents a duplicated outbound stop.
- **title** – the name of the stop to displayed in a User Interface, such as “5th St & Main, City Hall”.
- **shortTitle** – some transit agencies define short version of the title that are useful for applications where screen real estate is limited. This element is only provided when a separate short title exists.
- **lat/lon** – specify the location of the stop.
- **stopId** – an optional numeric ID to identify a stop. Useful for telephone or SMS systems so that a user can simply enter the numeric ID to identify a stop instead of having to select a route, direction, and stop. Not all transit agencies have numeric IDs to identify a stop so this element will not always be available.

A direction has the following attributes:

- **tag** – unique alphanumeric identifier for the direction.
- **title** – the name of the direction to be displayed in the User Interface, such as “Inbound to Caltrain Station”.
- **name** – a simplified name so that directions can be grouped together. If there are several Inbound directions for example then they can all be grouped together because they will all have the same name “Inbound”. This element is not available for all transit agencies.
- **List of stops** – within the direction there is a list of stops in order. This are useful for creating a User Interface where the user selects a route, direction, and then stop in order to obtain predictions.

The paths are simply lists of coordinates that can be used to draw a route on a map. The path data can be voluminous. If you do not need the path data you should add “&terse” to the routeConfig URL and the volume of returned data will be cut approximately in half. This is especially useful for mobile apps where you want to transfer as little data as possible.

Due to the nature of the configuration there can be many separate paths, some of them overlapping. A map client should simply draw all of the paths. The paths are not necessarily in any kind of order so you should only connect the points within a path. You should not connect the points between two separate paths though.

For the example URL:

https://retro.umoiq.com/service/publicXMLFeed?command=routeConfig&a=sf-muni&r=N

the resulting XML is in the form:

```
    <body>
      <route tag="N" title="N - Judah" color="003399" oppositeColor="ffffff"
      latMin="37.7601699" latMax="37.7932299" lonMin="-122.5092" lonMax="-122.38798">
      <stop tag="KINGd4S0" title="King St and 4th St" shortTitle="King & 4th"
      lat="37.776036" lon="-122.394355" stopId="1"/>
         <stop tag="KINGd2S0" title="King St and 2nd St" shortTitle="King &
         2nd" lat="37.7796152" lon="-122.3898067" stopId="2"/>
         <stop tag="EMBRBRAN" title="Embarcadero and Brannan St"
         shortTitle="Embarcadero & Brannan" lat="37.7844455" lon="-122.3880081"
         stopId="3"/>
         <stop tag="EMBRFOLS" title="Embarcadero and Folsom St"
         shortTitle="Embarcadero & Folsom" lat="37.7905742" lon="-122.3896326"
         stopId="4"/>
         ...

         <direction tag="out" title="Outbound to La Playa" name="Outbound"
useForUI="true">
          <stop tag="KINGd4S0"/>
          <stop tag="KINGd2S0"/>
          <stop tag="EMBRBRAN"/>
          <stop tag="EMBRFOLS"/>
          <stop tag="CVCENTF"/>
         </direction>
         <direction tag="in" title="Inbound to Caltrain" name="Inbound"
useForUI="true">
          <stop tag="CVCENTF"/>
          <stop tag="EMBRFOLS"/>
          <stop tag="EMBRBRAN"/>
          <stop tag="KINGd2S0"/>
          <stop tag="KINGd4S0"/>
         </direction>
         <direction tag="in_short" title="Short Run" name="Inbound"
useForUI="false">
          <stop tag="CVCENTF"/>
          <stop tag="EMBRFOLS"/>
          <stop tag="EMBRBRAN"/>
         </direction>
         ...

         <path>
          <point lat="37.7695171" lon="-122.4287571"/>
          <point lat="37.7695099" lon="-122.42887"/>
         </path>
         <path>
          <point lat="37.77551" lon="-122.39513"/>
          <point lat="37.77449" lon="-122.39642"/>
          <point lat="37.77413" lon="-122.39687"/>
          <point lat="37.77385" lon="-122.39721"/>
          <point lat="37.7737399" lon="-122.39734"/>
          <point lat="37.77366" lon="-122.39744"/>
          <point lat="37.77358" lon="-122.39754"/>
          <point lat="37.77346" lon="-122.39766"/>
          <point lat="37.77338" lon="-122.39772"/>
          <point lat="37.77329" lon="-122.39778"/>
          <point lat="37.77317" lon="-122.39784"/>
         </path>
         <path>
          <point lat="37.76025" lon="-122.50927"/>
          <point lat="37.76023" lon="-122.50928"/>
          <point lat="37.76017" lon="-122.50928"/>
          <point lat="37.7601299" lon="-122.50927"/>
          <point lat="37.76008" lon="-122.50924"/>
```

```
      <point lat="37.76006" lon="-122.50921"/>
      <point lat="37.7600399" lon="-122.50916"/>
      <point lat="37.76003" lon="-122.50912"/>
      <point lat="37.7600399" lon="-122.50906"/>
      <point lat="37.76005" lon="-122.50902"/>
      <point lat="37.76008" lon="-122.50898"/>
      <point lat="37.76017" lon="-122.50885"/>
    </path>
    ...
  </route>
</body>
```

# Prediction Requests

Prediction requests are used to obtain arrival/departure predictions for a stop or a set of stops.

## *Command "predictions"*

To obtain predictions associated with a stop use the "predictions" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used.

There are two ways to specify the stop: 1) using a stopId or 2) by specifying the route and stop tags. The stopId is a number that is useful for application such as a telephone or SMS system. It allows a user to specify a single stopId and get information for multiple routes. In addition, a stopId is not guaranteed to be unique for a particular stop. Instead, a stopId can actually refer to multiple stops, such as when there are several bus bays at a transit terminal. A stopId is specified in the form "stopId=0001".

A stop tag on the other hand is a unique identifier for a physical stop. It is used along with a route specifier in order to indentify a stop. It is typically used when creating a User Interface where a user selects a route, direction, and then stop. It is also used when predictions are needed for only a single route served by a stop. A stop tag can contain non-numeric characters. Even for transit agencies where the stop tags appear to be numbers the application must still be able to handle alphanumeric stop tags because the configuration might change in the future. The route is specified by the "r" parameter. The tag for the route is obtained using the routeList command. The stop tag is specified by the "s" parameter. The tag for the stop is obtained using the routeConfig command.

One can also specify the useShortTitles=true parameter so that shorter names for the agency, route, direction, and stop are returned if shorter names are available. Shorter names can be useful for smaller displays such as with wireless devices.

For small agencies that have a special agreement with Umo IQ, the "predictions" command can also be called with just the "a" parameter to specify the agency. This will cause predictions for every stop for the agency to be returned.

Predictions are returned in the <predictions> element, which contains attributes that make it easy to display the results to a passenger. The attributes are:

- **agencyTitle** (string) – The name of the agency to be displayed to passenger.
- **routeTag** (string) – Identifier for the route.
- **routeTitle** (string) – Title of the route to be displayed to passenger.
- **stopTitle** (string) – Title of the route to be displayed to passenger.
- **dirTitleBecauseNoPredictions** (string) – Title of direction. This attribute is only provided if there are no predictions. The direction title is provided in this situation because no <direction> elements are available since there are no predictions. This way the User Interface can still display the title of the direction selected even when there are no predictions.

Within the <predictions> element are possibly multiple <direction> elements and possibly multiple <message> elements.

Because routes can have multiple destinations the predictions are separated by <direction> so the client can let the passenger know whether the predicted for vehicles are actually going to their

destination. Then within the <direction> element a separate <prediction> element is provided, one for each vehicle being predicted. No more than 5 predictions per direction will be provided in the feed.

The <message> elements are important to handle because they provide important status information to passengers. These messages might be configured for an entire agency, for a route, or for a set of stops. If more detailed information is required then should use the messages command. Multiple messages can be returned for a single stop. The only attribute in the message element is called text, which contains the text of the message.

The predictions are returned in both seconds and minutes. The “minute” value is what should currently be displayed. The “seconds” value can be used to determine when the minute value will change requiring an update. Predictions should only be displayed in minutes, rounding down the number of seconds. The predictions are also provided in “epochTime”. Epoch time is a standard, defined as the number of seconds elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds. It is useful for when one needs to display the prediction time as a time of day, such as “4:15pm”.

Additional attributes contained in the prediction output are:

- **isDeparture** (boolean) - For most stops predictions of when the bus will arrive are provided. But there are some stops, especially stops at the beginning of a trip, where there is a layover. This means that the bus is stopped for at least several minutes. For these situations the departure time is provided because a passenger is only interested in the departure time for such situations. Toronto TTC streetcars are an exception. At terminals arrival predictions are provided for TTC streetcars since they are managed more by headway instead of schedule. In order to specify whether a prediction is for a departure as opposed to an arrival time the additional tag element “isDeparture” is provided along with the predictions. If it is set to true then the prediction is for the departure time. Otherwise the prediction is for an arrival time. For departures Umo IQ uses the schedule to determine when the vehicle will be leaving the stop if the vehicle is on-time or running early. If the vehicle is running late then the departure time will be based on the vehicles arrival time (assuming that the vehicle will depart as soon as it arrives without any layover).
- **block** (string) - Specifies the block number assigned to the vehicle as defined in the configuration data.
- **dirTag** (string) - Specifies the ID of the direction for the stop that the prediction is for (as opposed to the direction that the vehicle is currently on). The direction ID is usually the same as a trip pattern ID, but is very different from the tripTag. A direction or trip pattern ID specifies the configuration for a trip. It can be used multiple times for a block assignment. But a tripTag identifies a particular trip within a block assignment.
- **tripTag** (string) - Specifies the ID of the trip for when the vehicle will be arriving at the stop, as defined in the configuration data. This element is only included in the XML feed if the trip ID is actually included in the configuration data.
- **branch** (string) – This attribute is provided only for the Toronto TTC agency. For TTC there are many routes with multiple “branches”, where the route has different paths. For route 107 for example there can be branches “107”, “107A”, “107B”, “107C”, etc. By using the branch information in the User Interface the passengers can see if a prediction is for a bus that is going on their desired branch.
- **affectedByLayover** (boolean) **-** Specifies whether the predictions are based not just on the position of the vehicle and the expected travel time, but also on whether a vehicle leaves a terminal at the configured layover time. This information can be useful to passengers because predictions that are affected by a layover will not be as accurate. This element is only included in the XML feed when the value is true. If the value is not set then it should be considered false.

- **isScheduleBased** (boolean) **-** Specifies whether the predictions are based solely on the schedule and do not take the GPS position of the vehicle into account. This feature is not currently available for TTC. This element is only included in the XML feed when the value is true. If the value is not set then it should be considered false.
- **delayed** (boolean) **–** Specifies if the bus is not traveling as fast as expected over the last few minutes. This is useful for determining if a vehicle is stuck in traffic such that the predictions might not be as accurate. This feature is only enabled for certain agencies. This element is only included in the XML feed when the value is true. If the value is not **set then it should be considered false. A "slowness" value is also provided when a vehicle** is delayed but this attribute is experimental and might be removed from the feed. The slowness attribute indicates how slow a vehicle is traveling over the last few minutes compared to normal.

The two URLs to obtain predictions for a physical stop identified by a numerical stop ID are shown below. The first example does not specify a route tag so predictions for all routes that serve the stop will be returned. If predictions are desired for only a single route then the optional routeTag should be specified in the URL as shown in the second example.

1) **https://retro.umoiq.com/service/publicXMLFeed?command=predictions&a=<agency_tag>&stopId=<stop id>**

2) **https://retro.umoiq.com/service/publicXMLFeed?command=predictions&a=<agency_tag>&stopId=<stop id>&routeTag=<route tag>**

Note: Can also use the query string option useShortTitles=true to have the XML feed return short titles intended for display devices with small screens.

The format of the command for obtaining predictions for a particular stop using a stop tag is as follows.

**https://retro.umoiq.com/service/publicXMLFeed?command=predictions&a=<agency_tag>&r=<route tag>&s=<stop tag>**

An example of obtaining predictions by specifying a route and stop tag is shown below.

**https://retro.umoiq.com/service/publicXMLFeed?command=prediction s&a=sf-muni&r=N&s=5205&useShortTitles=true**

The resulting XML is in the form:

```
<body>
  <predictions agencyTitle="San Francisco Muni, CA" routeTag="N"
   routeCode="1" routeTitle="N - Judah" stopTitle="Civic Center Station
   Outbound">
   <direction title="Outbound toward Ocean Beach" >
     <prediction seconds="563" minutes="9" epochTime="1229637162309"
        isDeparture="false" dirTag="N__OB2" block="9703" />
    <prediction seconds="1250" minutes="20" epochTime="1229638264817"
        isDeparture="false" dirTag="N__OB2" block="9702" />
   </direction>
   <direction title="Outbound to 34th Ave" >
     <prediction seconds="123" minutes="2" epochTime="1229637162309"
        isDeparture="true" dirTag="N__OB1" block="9708" />
    <prediction seconds="621" minutes="10" epochTime="1229637162309"
        isDeparture="true" dirTag="N__OB1" block="9707" />
   </direction>
     <message text="No Muni metro service btwn Caltrain/Embarc. &amp;amp; Castro
     after 10pm Mon-Fri. Bus service provided."/>
```

```
  </predictions>
</body>
```

Note: the predictions are grouped by direction. For situations where buses on a line have different destinations because some turn back earlier than others, the predictions presented to the user can provide this important piece of information.

## *Command "predictionsForMultiStops"*

To obtain predictions associated with multiple stops use the "predictionsForMultiStops" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used. The stops are specified by using the "stops" parameter multiple times. Each stop is separated by the "|" character and each stop is represented by a route and stop identifier, concatenated together. One can also specify the useShortTitles=true parameter so that shorter names for the agency, route, direction, and stop are returned if shorter names are available. Shorter names can be useful for smaller displays such as with wireless devices.

Currently the predictionsForMultiStops command can only be specified with stop tags. In the future we expect to expand the feed so that the command can also be used with stopIds.

Predictions should only be displayed in minutes, rounding down the number of seconds.

The format of the command for obtaining predictions for a list of stops is (where a stop specified is a route tag and a stop tag separated by the "|" character):

```
https://retro.umoiq.com/service/publicXMLFeed?command=prediction
sForMultiStops&a=<agency_tag>&stops=<stop    1>&stops=<stop    2>&stops=<stop
3>
```

For the example URL:

```
https://retro.umoiq.com/service/publicXMLFeed?command=prediction
sForMultiStops&a=sf-muni&stops=N|6997&stops=N|3909
```

The resulting XML will be in the form:

```
<body>
  <direction title="Outbound toward Ocean Beach">
   <predictions stopTitle="Civic Center Station Outbound" routeCode="1"
      routeTitle="N - Judah">
     <prediction seconds="218" minutes="3" epochTime="1229637162309"
       isDeparture="false" />
     <prediction seconds="976" minutes="16" epochTime="122963716923"
       isDeparture="false" />
    </predictions>
  </direction>
  <direction title="Outbound toward Ocean Beach">
    <predictions stopTitle="Carl St and Cole St" routeCode="1"
     routeTitle="N - Judah">
    <prediction seconds="763" minutes="12" epochTime="1229637123422"
       isDeparture="false" />
    <prediction seconds="1281" minutes="21" epochTime="1229637168293"
       isDeparture="false" />
    <prediction seconds="1521" minutes="25" epochTime="1229637302334"
       isDeparture="false" />
```

```
    <prediction seconds="2027" minutes="33" epochTime="1229637430203"
        isDeparture="false" />
    <prediction seconds="2747" minutes="45" epochTime="1229637502034"
        isDeparture="false" />
   </predictions>
  </direction>
</body>
```

## *Command "schedule"*

To obtain the schedule information for a route use the "schedule" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used. The route is specified by the "r" parameter. The tag for the route is obtained using the routeList command.

The format of the command is:

https://retro.umoiq.com/service/publicXMLFeed?command=schedule&a=<agency_tag>& r=<route_tag>

The schedule data returned has multiple elements. It begins with a route element that has these attributes:

- **tag** - unique alphanumeric identifier for the route, such as "N".
- **title** - the name of the route to be displayed in a User Interface, such as "N-Judah".
- **scheduleClass** - name of the current schedule class, which may change with the seasons (e.g. fall and spring schedules).
- **serviceClass** - indicates service date(s) when the schedule applies, which may differ on weekdays, weekend days, and holidays.
- **direction** - simplified name for travel directions, grouped together.

This is followed by a header element that contains a number of stop elements. These stop elements have one attribute and content:

- **tag** - unique alphanumeric identifier for the stop, such as "5225". Even if the stop tags appear to usually be numeric they can sometimes contain alphabetical characters. The content of the element is a title that can be displayed in a User Interface, such as "Judah St & Sunset Blvd".

This is followed by a number of tr elements, which have one attribute:

- **blockID** - specifies the block number as defined in the configuration data.

Each tr element has multiple stop elements. The stop elements have these attributes and content:

- **tag** - unique identifier for the stop.
- **epochTime** - scheduled arrival as epoch time (see above). The content of the element is the time in HH:mm:ss format, which could be displayed in a User Interface. For trips where a vehicle does not serve the stop the epochTime is set to -1 and the time data is output as “—“.

For the example URL:

https://retro.umoiq.com/service/publicXMLFeed?command=schedule&a=sf-muni&r=N

The resulting XML is in the form:

```
<body>
  <route tag="N" title="N-Judah" scheduleClass="2011JANUARY"
serviceClass="wkd" direction="Inbound">
    <header>
      <stop tag="5225">Judah St and Sunset Blvd</stop>
      <stop tag="5200">Judah St and 19th Ave</stop>
      <stop tag="3913">Carl St and Hillway Ave</stop>
      ...
    </header>
    <tr blockID="9721">
      <stop tag="5225" epochTime="540000">00:09:00</stop>
      <stop tag="5200" epochTime="840000">00:14:00</stop>
      <stop tag="3913" epochTime="1440000">00:24:00</stop>
      ...
    </tr>
    <tr blockID="9703">
      <stop tag="5225" epochTime="1680000">00:28:00</stop>
      <stop tag="5200" epochTime="1980000">00:33:00</stop>
      <stop tag="3913" epochTime="2460000">00:41:00</stop>
      ...
    </tr>
    ...
  </route>
</body>
```

This form differs from that of our other commands, in that its structure parallels that of an HTML table. The above example could very easily be reworked into this:

```
  <table>
    <tr>
      <th>Judah St and Sunset Blvd</th>
      <th>Judah St and 19th Ave</th>
      <th>Carl St and Hillway Ave</th>
      ...
    </tr>
      <td>12:09 am</td>
      <td>12:14 am</td>
      <td>12:24 am</td>
      ...
    </tr>
    <tr>
      <td>12:28 am</td>
      <td>12:33 am</td>
      <td>12:41 am</td>
      ...
    </tr>
    ...
  </table>
```

# Message Requests

Message requests are used to obtain messages that are currently active.

## *Command "messages"*

To obtain messages currently active for a route or a group of routes, use the "messages" command. Note: this command only returns currently active messages. If a message is configured just to be shown on Tuesdays for example but it is currently Wednesday then the message will not be listed. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used.

The format of the command for obtaining messages for a route or a group of routes is shown below. One can optionally specify route tags to get messages for particular routes. Multiple route tags can be specified simply by specifying them separately in the url. For example, if you want messages for routes "N" and "J" you would specify "&r=N&r=J". If a message is defined for multiple routes then it will be listed for each route. If you do not specify a route tag then messages for all routes will be returned.

https://retro.umoiq.com/service/publicXMLFeed?command=messages&a=<**agency tag**>&r=<**route tag1**>&r=<**route tagN**>

The resulting XML is in the form shown below. Messages are grouped by route. The first group will be for system wide messages. For such messages the route tag will be set to "all". If messages were configured to be sent to specific routes then each message will contain a <routeConfiguredForMessage> element. Since messages can be targeted to multiple routes each message can contain multiple <routeConfiguredForMessage> element. If a message is configured just for certain stops on a route then the <routeConfiguredForMessage> element will contain a <stop> element for each stop.

Each message includes a unique id for identifying the message. Each one also contains the login of the person who created the message. The start boundary and end boundary are only provided if the message was specified to have a start and stop time or a duration. If start and end boundaries are not listed then the message is to be displayed until it is canceled.

Messages contain a <text> element that specifies the text for the message. They can also contain a <textSecondaryLanguage> element for when there is separate text in a different language for the message. So for transit agencies in Quebec Canada the <text> element can have the French text and the <textSecondaryLanguage> element can have the English text. And a message can also contain a <phonemeText> element if optional text has been provided to improve the pronunciation of a Text To Speech (TTS) system.

Messages will also optionally contain <interval> elements if they were configured to only be displayed for certain intervals. This is used when a message is defined to be active for a long period of time, but only for certain intervals during that time. So for example, a message couple be configured for a month but it might only be active on Tuesdays and Thursdays from 7am to 9am during that period. There will be a separate interval element for each day of the week. An interval has 4 attributes. The startDay and endDay are integers specifying the day of the week. A 1 is for Monday, 2 for Tuesday, etc. The startTime and endTime are time during the day in seconds that the message is to start or stop. So a value of 3600 corresponds to 1am (3600 seconds past midnight).

For the agency STL there is an additional optional attribute called sendToBuses that can be included in the <message> element. If it is present and set to true then the message is also intended for display on board buses. If it is not present it should be considered to be false.

For the agency "Agence metropolitaine de transport" in Quebec there are two additional elements. The first one is <smsText> and it specifies corresponding text to be provided only via SMS. The second element is <priority> and it is for specifying the priority of the message, 1 being the highest priority and 5 being the lowest.

Sample XML output is shown below:

```
<body>
  <route tag="all">
    <message id="1700" creator="supervisor" sendToBuses="true"
             startBoundary="1253232780000" startBoundaryStr="Thu Sep 17
17:13:00 PDT 2009"
            endBoundary="1253233200000" endBoundaryStr="Thu Sep 17
17:20:00 PDT 2009">
      <interval startDay="2" startTime="75600" endDay="4"
endTime="79200" />
      <interval startDay="4" startTime="75600" endDay="4"
endTime="79200" />
      <text>Thanks for taking transit</text>
      <textSecondaryLanguage>Thanks for taking transit, in
Spanish</textSecondaryLanguage>
      <phonemeText>Thanks for taking traansit</phonemeText>
    </message>
  </route>
  <route tag="bshop">
    <message id="1701" creator="supervisor"
             startBoundary="1253232780000" startBoundaryStr="Thu Sep 17
17:13:00 PDT 2009"
            endBoundary="1253233200000" endBoundaryStr="Thu Sep 17
17:20:00 PDT 2009">
      <routeConfiguredForMessage tag="bshop">
        <stop tag="1324" title="7th&Main" />
        <stop tag="1325" title="9th&Main" />
      </routeConfiguredForMessage>
      <routeForMessage tag="franklin" />
      <text>Detour on Malou St on Thursday</text>
       <phonemeText>Detour on Malooo St on Thursday</phonemeText>
      <smsText>Detour on Malou on Thursday</smsText>
      <priority>3</priority>
    </message>
  </route>
</body>
```

# Vehicle Location Requests

Vehicle locations are used to draw vehicles on a map or to simply store them in a database. The vehicle locations should not be polled more than once every 10 seconds.

## *Command "vehicleLocations"*

To obtain a list of vehicle locations that have changed since the last time the vehicleLocations command was used, use the "vehicleLocations" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used. The route is specified by the "r" parameter. The tag for the route is obtained using the routeList command. The "t" parameter specifies the last time that was returned by the vehicleLocations command. The time in msec since the 1970 epoch time. If you specify a time of 0, then data for the last 15 minutes is provided.

The format of the command for obtaining vehicle location is:

https://retro.umoiq.com/service/publicXMLFeed?command=vehicleLoc
ations&a=<agency_tag>&r=<route tag>&t=<epoch time in msec>

Attributes contained in the vehicle locations output are:

- **vehicle id** (string) – Identifier of the vehicle. It is often but not always numeric.
- **routeTag** (string) - Specifies the ID of the route the vehicle is currently associated with.
- **dirTag** (string) - Specifies the ID of the direction that the vehicle is currently on. The direction ID is usually the same as a trip pattern ID, but is very different from the tripTag. A direction or trip pattern ID specifies the configuration for a trip. It can be used multiple times for a block assignment. But a tripTag identifies a particular trip within a block assignment.
- **lat/lon** – specify the location of the vehicle.
- **secsSinceReport** (int) – How many seconds since the GPS location was actually recorded. It should be noted that sometimes a GPS report can be several minutes old.
- **predictable** (boolean) – Specifies whether the vehicle is currently predictable.
- **heading** (int) – Specifies the heading of the vehicle in degrees. Will be a value between 0 and 360. A negative value indicates that the heading is not currently available.
- **speedKmHr** (double) – Specifies GPS based speed of vehicle.

There is also a lastTime element that contains the update time of the last vehicle location returned. This value can be used as the “t” parameter for the next call to the vehicleLocations command so that only GPS reports since the last time the command was called will be returned. This is useful for preventing reading in vehicle locations that have not changed, reducing bandwidth required. The single attribute for the lastTime element is:

- **time** (msec since 1970 epoch time) – Specifies time of the last update for the vehicle location data returned.

For the example URL:

https://retro.umoiq.com/service/publicXMLFeed?command=vehicleLoc
ations&a=sf-muni&r=N&t=1144953500233

The resulting XML is in the form:

```
<body>
  <vehicle id="1453" routeTag="N" dirTag="out" lat="37.7664199" lon="-
   122.44896" secsSinceReport="29" predictable="true" heading="276"/>
  <vehicle id="1549" routeTag="N" dirTag="in" lat="37.77631" lon="-
   122.3941" secsSinceReport="3" predictable="true" heading="45"/>
  <vehicle id="1517" routeTag="N" dirTag="in_short" lat="37.76035" lon="-
   122.50794" secsSinceReport="69" predictable="true" heading="267"/>
```

```
<vehicle id="1547" routeTag="N" dirTag="out" lat="37.76952" lon="-
  122.43174" secsSinceReport="28" predictable="true" heading="85"/>
<vehicle id="1404" routeTag="N" dirTag="out" lat="37.76003" lon="-
  122.50919" secsSinceReport="9" predictable="true" heading="117"/>
<vehicle id="1400" routeTag="N" dirTag="in" lat="37.76415" lon="-
  122.46409" secsSinceReport="50" predictable="true" heading="266"/>
<lastTime time="1144953510433"/>
</body>
```

## *Command "vehicleLocation"*

To obtain a vehicle location for a particular vehicle use the "vehicleLocation" command. The agency is specified by the "a" parameter in the query string. The tag for the agency as obtained from the agencyList command should be used. The vehicle Id is specified by the "v" parameter.

The format of the command for obtaining vehicle location is:

https://retro.umoiq.com/service/publicXMLFeed?command=vehicleLoc
ation&a=<agency_tag>&v=<vehicle id>

Attributes contained in the vehicle locations output are:

- **vehicle id** (string) – Identifier of the vehicle. It is often but not always numeric.
- **routeTag** (string) - Specifies the ID of the route the vehicle is currently associated with.
- **dirTag** (string) - Specifies the ID of the direction that the vehicle is currently on. The direction ID is usually the same as a trip pattern ID, but is very different from the tripTag. A direction or trip pattern ID specifies the configuration for a trip. It can be used multiple times for a block assignment. But a tripTag identifies a particular trip within a block assignment.
- **lat/lon** – specify the location of the vehicle.
- **secsSinceReport** (int) – How many seconds since the GPS location was actually recorded. It should be noted that sometimes a GPS report can be several minutes old.
- **predictable** (boolean) – Specifies whether the vehicle is currently predictable.
- **heading** (int) – Specifies the heading of the vehicle in degrees. Will be a value between 0 and 360. A negative value indicates that the heading is not currently available.

For the example URL:

https://retro.umoiq.com/service/publicXMLFeed?command=vehicleLoc
ation&a=sf-muni&v=1453

The resulting XML is in the form:

```
<body>
  <vehicle id="1453" routeTag="N" dirTag="out" lat="37.7664199" lon="-
   122.44896" secsSinceReport="29" predictable="true" heading="276"/>
</body>
```