chicago transit authority
Train Tracker API documentation

Introduction

CTA Train Tracker [SM] is a product currently in the beta testing phase. This document covers the arrivals API, currently under release for testing and evaluation by developers who work with CTA data.

Note that, to use this API, you must agree to our Developers License Agreement and Terms of Use. You'll also need to apply for a key.

Table of Contents

About this document

What this document covers

This document explains how to request information and what information is provided through the single arrivals API. Additional APIs may be released at a later date, and this documentation would then be updated accordingly.

What this document does not cover

This document does not cover information provided through the CTA Customer Alerts API, Google Transit Feed Specification data package nor the CTA Bus Tracker API. Visit transitchicago.com/developers for more information on these other data services from CTA.

How information gets into our system

Information in the CTA Train Tracker beta comes from data fed to CTA from its rail infrastructure (unlike buses, our current railcar fleet does not have GPS hardware). This data is then processed by software we use to monitor our rail system which also generates the predictions for train arrivals based on recent train travel times from one point to another. (The software is a product called QuicTrak®.)

Prediction data are combined with other data and polished to help present information in the most meaningful way possible.

Note: QuicTrak is a registered trademark of QEI, Inc.

Some other things you should know

This service is in beta—it may not always be available and may be changed (see DLA for complete details). Here are some notes about what you can expect from the data:

- Predictions for train arrivals are generated as trains move from one section of track to another and for estimated arrivals up to fifteen minutes beyond the train's current position.
- Predictions may be withheld in circumstances where we expect them to be inaccurate, such as during major work, reroutes or unavoidable service disruptions.

  Important tip: Use the Customer Alerts API to determine whether or not an event is affecting service and relay to your end-users when some or all predictions may be affected or unavailable due to an event that affects service.

- When no predictions are available for a station, such as because no train has yet departed a terminal, we offer up to one scheduled departure time in place of a live prediction so long as service is scheduled. Terminal departures are always represented as a scheduled departure, as live information is not presently available until a train leaves.
- Arrival predictions are available for locations where trains pick up passengers (predictions for terminal arrivals and exit-only platforms are not presently available in the data).

- Predictions for a specific train run number are not available at this time—only for arrival/departures from stops where passengers are accepted.
- Internal testing has shown train arrival accuracy averages <±1 minute from prediction times. Average actual variance from predictions may vary as traffic conditions change.
- Unscheduled express runs (where a train runs non-stop from one point on a route to another, such as to space out service following an unavoidable delay) are not indicated as express in CTA Train Tracker at this time.
- The default daily transaction limit for this API is 50,000 transactions. If you need additional transactions, contact webmaster@transitchicago.com with your request. Additionally, there is DoS protection installed on our servers which may trigger a temporary "time-out" if a large number of transactions from a single IP address.

Why we're providing these APIs

The hope is that, by publishing this data, it ends up in all sorts of places beyond CTA's sites and services. By having transit information in as many contexts as people might use it, we can extend our reach and help people make informed decisions which can improve people's experiences with transit.

If you experience any issues or have any comments regarding this service and related policies, please contact us right away. Your feedback is extremely valuable to us!

Legal notice

By using this API, you agree to our Developer License Agreement and Terms of Use, the latest version of which is included, as of the publication date of this document, in this document's appendices.

It's important that you, the developer, understand that this service is provided on an as-is basis and without any guarantees as to availability or accuracy. You must read and agree to the full Developer Terms of Use to use this API.

A few definitions…

There are a few bits of lingo that you'll find in this document (we'll try to keep it to a minimum) that we'd like to explain first, so you know what we're talking about.

Customer Alert – An entry in our Customer Alerts database which describes a condition that can affect someone's trip on CTA; see Customer Alerts API for comprehensive alert information, including a special flag for train alerts that indicates whether or not an event may cause Train Tracker to behave less-reliably than normal.

Google Transit Feed Specification (GTFS) – This is a "common format for public transportation schedules and associated geographic information." GTFS is used by hundreds of transit agencies to feed service information to Google. A GTFS package is generated, as needed, by transit agencies and can be distributed as a simple .zip file with several comma-delimited text files inside. You can read more about GTFS on Google Code. For consistency, the same route IDs and stop IDs are used throughout the Bus Tracker system, the Alerts system as are specified in the CTA GTFS feed (with a few special exceptions—see appendix).

Delay – In the context of this document, a delayed train is one that has not moved from one track circuit to another for an abnormally long period of time.

Terminal – A point of departure or terminus on a train route.

Arrivals API

Description

This API produces a list of arrival predictions for all platforms at a given train station in a well-formed XML document. Registration and receipt of an API key is required for use of this API.

(The separate Follow API produces a list of arrival predictions for a given train at all subsequent stations for which that train is estimated to arrive, up to 20 minutes in the future or to the end of its trip.)

Each separate prediction describes a single train, when it's expected to arrive, and various bits of information that explain where it's expected to arrive and certain attributes about the train.

Base URL

lapi.transitchicago.com/api/1.0/ttarrivals.aspx

Parameters:

Use URL query string method.

Name	Value	Description
mapid	**Numeric station identifier** (required if stpid not specified)	A single five-digit code to tell the server which station you'd like to receive predictions for. See appendix for information about valid station codes.
stpid	**Numeric stop identifier** (required if mapid not specified)	A single five-digit code to tell the server which specific stop (in this context, specific platform or platform side within a larger station) you'd like to receive predictions for. See appendix for information about valid stop codes.
max	**Maximum results** (optional)	The maximum number you'd like to receive (if not specified, all available results for the requested stop or station will be returned)
rt	**Route code** (optional)	Allows you to specify a single route for which you'd like results (if not specified, all available results for the requested stop or station will be returned)
key	**Alphanumeric API key** (required)	Your unique API key, assigned to you after agreeing to DLA and requesting a key be generated for you.

outputType	**"JSON"** (optional)	If outputType=JSON is added to query string, API response will be formatted as JSON instead of XML

Response fields:

Name	Description
ctatt	Root element
./tmst	Shows time when response was generated in format: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./errCd	Numeric error code (see appendices)
./errNm	Textual error description/message (see appendices)
./eta	Container element (one per individual prediction)
./../staId	Numeric GTFS parent station ID which this prediction is for (five digits in 4xxxx range) (matches "mapid" specified by requestor in query)
./../stpId	Numeric GTFS unique stop ID within station which this prediction is for (five digits in 3xxxx range)
./../staNm	Textual proper name of parent station
./../stpDe	Textual description of platform for which this prediction applies
./../rn	Run number of train being predicted for
./../rt	Textual, abbreviated route name of train being predicted for (matches GTFS routes)
./../destSt	GTFS unique stop ID where this train is expected to ultimately end its service run (experimental and supplemental only—see note below)
./../destNm	Friendly destination description (see note below)
./../trDr	Numeric train route direction code (see appendices)
./../prdt	Date-time format stamp for when the prediction was generated: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./../arrT	Date-time format stamp for when a train is expected to arrive/depart: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./../isApp	Indicates that Train Tracker is now declaring "Approaching" or "Due" on site for this train
./../isSch	Boolean flag to indicate whether this is a live prediction or based on schedule in lieu of live data
./../isFlt	Boolean flag to indicate whether a potential fault has been detected (see note below)
./../isDly	Boolean flag to indicate whether a train is considered "delayed" in Train Tracker
./../flags	Train flags (not presently in use)
./../lat	Latitude position of the train in decimal degrees
./../lon	Longitude position of the train in decimal degrees
./../heading	Heading, expressed in standard bearing degrees (0 = North, 90 = East, 180 = South, and 270 = West; range is 0 to 359, progressing clockwise)

Sample Request URL:

http://lapi.transitchicago.com/api/1.0/ttarrivals.aspx?key=a8456djbhf8475683jf7818bha81&mapid=40380&max=5

Remarks:

Stop IDs: Use stop information in GTFS Stops table for associated geolocation of stops listed here. For purposes of this API, use the parent station ID (4xxxx range of stop ID numbers) to specify a station.

Destination station ID #s: in `destSt` refer to the ultimate destination of a train per the information about each train that's on the move.

These destination station ID #s are only available once a train has departed (on schedule-based predictions, this element will show "0") and doesn't necessarily match with what will be indicated on a train's destination sign (particularly on routes that operate around the whole Loop).

Once a train leaves Midway, its ultimate destination station's ID is "Midway" because the train will make all stops to the Loop, go around it, and come back to Midway. This allows you to write your own logic on what to show, but we've already gone and done the work for you.

The `destNm` element is the public, friendly-name that should match the destination sign of approaching trains. For example, predictions a train heading toward the Loop on the Orange Line, using this element, will return "Loop" in a result set at Roosevelt, but that same train, even while still Loop-bound, will be listed as being to "Midway" for Harold Washington Library/State-Van Buren, because our system knows that once it gets to Library, it'll now be considered a Midway-bound train.

Lat/lon/heading: This information is available only for trains that are in-service (i.e., have left their terminals). Some entries are based on our written schedule, as a courtesy, for when live information isn't available. Schedule-based entries in a response will simply have an empty lat, lon and heading element as location info isn't available.

Calculating a number of minutes: See Appendix D for extended notes on this subject.

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ctatt">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="tmst" type="xs:string" />
        <xs:element name="errCd" type="xs:unsignedByte" />
        <xs:element name="errNm" />
        <xs:element maxOccurs="unbounded" name="eta">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="staId" type="xs:unsignedShort" />
              <xs:element name="stpId" type="xs:unsignedShort" />
              <xs:element name="staNm" type="xs:string" />
              <xs:element name="stpDe" type="xs:string" />
              <xs:element name="rn" type="xs:unsignedShort" />
              <xs:element name="rt" type="xs:string" />
              <xs:element name="destSt" type="xs:unsignedShort" />
              <xs:element name="destNm" type="xs:string" />
              <xs:element name="trDr" type="xs:unsignedByte" />
              <xs:element name="prdt" type="xs:string" />
              <xs:element name="arrT" type="xs:string" />
              <xs:element name="isApp" type="xs:unsignedByte" />
              <xs:element name="isSch" type="xs:unsignedByte" />
              <xs:element name="isDly" type="xs:unsignedByte" />
              <xs:element name="isFlt" type="xs:unsignedByte" />
              <xs:element name="flags" type="xs:string" />
              <xs:element name="lat" type="xs:decimal" />
              <xs:element name="lon" type="xs:decimal" />
              <xs:element name="heading" type="xs:unsignedShort" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example

What this request is asking for:
A maximum of one arrival prediction result from the station with the ID #40360. It also passes the API key for authorization (required).

XML Request:
http://lapi.transitchicago. com/api/1.0/ttarrivals.aspx?key=e3875818a4743049&max=1& mapid=40360

XML Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ctatt>
  <tmst>20110321 18:32:02</tmst>
  <errCd>0</errCd>
  <errNm />
    <eta>
      <staId>40360</staId>
      <stpId>30071</stpId>
      <staNm>Southport</staNm>
      <stpDe>Service toward Loop</stpDe>
      <rn>426</rn>
      <rt>Brn</rt>
      <destSt>31740</destSt>
      <destNm>Loop</destNm>
      <trDr>5</trDr>
      <prdt>20110321 18:31:29</prdt>
      <arrT>20110321 18:34:29</arrT>
      <isApp>0</isApp>
      <isSch>0</isSch>
      <isFlt>0</isFlt>
      <isDly>0</isDly>
      <flags/>
       <lat>41.97776</lat>
       <lon>-87.77567</lon>
       <heading>299</heading>
    </eta>
</ctatt>
```

JSON Request:
http://lapi.transitchicago. com/api/1.0/ttarrivals.aspx?key=e3875818a4743049&max=1& mapid=40360&outputType=JSON

JSON Response:

```
{
```

```
    "ctatt":{
        "tmst":"2015-04-30T20:23:53",
        "errCd":"0",
        "errNm":null,
        "eta":[
           {
               "staId":"40960",
               "stpId":"30185",
               "staNm":"Pulaski",
               "stpDe":"Service toward Loop",
               "rn":"726",
               "rt":"Org",
               "destSt":"30182",
               "destNm":"Loop",
               "trDr":"1",
               "prdt":"2015-04-30T20:23:32",
               "arrT":"2015-04-30T20:25:32",
               "isApp":"0",
               "isSch":"0",
               "isDly":"0",
               "isFlt":"0",
               "flags":null,
               "lat":"41.78661",
               "lon":"-87.73796",
               "heading":"357"
           }
        ]
    }
}
```

Follow This Train API

Description

This API produces a list of arrival predictions for a given train at all subsequent stations for which that train is estimated to arrive, up to 20 minutes in the future or to the end of its trip.

Each separate prediction describes a single train, when it's expected to arrive, and various bits of information that explain where it's expected to arrive and certain attributes about the train.

Base URL

lapi.transitchicago.com/api/1.0/ttfollow.aspx

Parameters:

Use URL query string method.

Name	Value	Description
runnumber	**Train Run Number** (required)	Allows you to specify a single run number for a train for which you'd like a series of upcoming arrival estimations.
key	**Alphanumeric API key** (required)	Your unique API key, assigned to you after agreeing to DLA and requesting a key be generated for you.
outputType	**"JSON"** (optional)	If outputType=JSON is added to query string, API response will be formatted as JSON instead of XML

Response fields:

Name	Description
ctatt	Root element
./tmst	Shows time when response was generated in format: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./errCd	Numeric error code (see appendices)
./errNm	Textual error description/message (see appendices)
./position	Container element (one per response describing train)
././lat	Latitude position of the train in decimal degrees

././lon	Longitude position of the train in decimal degrees
././heading	Heading, expressed in standard bearing degrees (0 = North, 90 = East, 180 = South, and 270 = West; range is 0 to 359, progressing clockwise)
./eta	Container element (one per individual prediction)
././staId	Numeric GTFS parent station ID which this prediction is for (five digits in 4xxxx range) (matches "mapid" specified by requestor in query)
././stpId	Numeric GTFS unique stop ID within station which this prediction is for (five digits in 3xxxx range)
././staNm	Textual proper name of parent station
././stpDe	Textual description of platform for which this prediction applies
././rn	Run number of train being predicted for
././rt	Textual, abbreviated route name of train being predicted for (matches GTFS routes)
././destSt	GTFS unique stop ID where this train is expected to ultimately end its service run (experimental and supplemental only—see note below)
././destNm	Friendly destination description (see note below)
././trDr	Numeric train route direction code (see appendices)
././prdt	Date-time format stamp for when the prediction was generated: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
././arrT	Date-time format stamp for when a train is expected to arrive/depart: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
././isApp	Indicates that Train Tracker is now declaring "Approaching" or "Due" on site for this train
././isSch	Boolean flag to indicate whether this is a live prediction or based on schedule in lieu of live data
././isFlt	Boolean flag to indicate whether a potential fault has been detected (see note below)
././isDly	Boolean flag to indicate whether a train is considered "delayed" in Train Tracker
././flags	Train flags (not presently in use)

Sample Request URL:

http://lapi.transitchicago.com/api/1.0/ttfollow.aspx?key=a8456djbhf8475683jf7818bha81&runnumber=426

Remarks:

Stop IDs: Use stop information in GTFS Stops table for associated geolocation of stops listed here. For purposes of this API, use the parent station ID (4xxxx range of stop ID numbers) to specify a station.

Destination station ID #s: in `destSt` refer to the ultimate destination of a train per the information about each train that's on the move.

These destination station ID #s are only available once a train has departed (on schedule-based predictions, this element will show “0”) and doesn’t necessarily match with what will be indicated on a train’s destination sign (particularly on routes that operate around the whole Loop).

Once a train leaves Midway, its ultimate destination station’s ID is “Midway” because the train will make all stops to the Loop, go around it, and come back to Midway. This allows you to write your own logic on what to show, but we’ve already gone and done the work for you.

The `destNm` element is the public, friendly-name that should match the destination sign of approaching trains. For example, predictions a train heading toward the Loop on the Orange Line, using this element, will return “Loop” in a result set at Roosevelt, but that same train, even while still Loop-bound, will be listed as being to “Midway” for Harold Washington Library/State-Van Buren, because our system knows that once it gets to Library, it’ll now be considered a Midway-bound train.

Calculating a number of minutes: See Appendix D for extended notes on this subject.

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="ctatt">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="tmst" type="xs:string" />
                <xs:element name="errCd" type="xs:unsignedByte" />
                <xs:element name="errNm" />
                <xs:element name="position" />
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="lat" type="xs:decimal" />
                            <xs:element name="lon" type="xs:decimal" />
                            <xs:element name="heading" type="xs:unsignedShort" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element maxOccurs="unbounded" name="eta">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="staId" type="xs:unsignedShort" />
                            <xs:element name="stpId" type="xs:unsignedShort" />
                            <xs:element name="staNm" type="xs:string" />
                            <xs:element name="stpDe" type="xs:string" />
                            <xs:element name="rn" type="xs:unsignedByte" />
                            <xs:element name="rt" type="xs:string" />
                            <xs:element name="destSt" type="xs:unsignedShort" />
                            <xs:element name="destNm" type="xs:string" />
                            <xs:element name="trDr" type="xs:unsignedByte" />
                            <xs:element name="prdt" type="xs:string" />
                            <xs:element name="arrT" type="xs:string" />
                            <xs:element name="isApp" type="xs:unsignedByte" />
                            <xs:element name="isSch" type="xs:unsignedByte" />
                            <xs:element name="isDly" type="xs:unsignedByte" />
                            <xs:element name="isFlt" type="xs:unsignedByte" />
                            <xs:element name="flags" type="xs:string" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Example

What this request is asking for:
Upcoming arrivals for run 123 (a Blue Line train). It also passes the API key for authorization (required).

(For brevity, what would be a longer series of “eta” sequence element is limited to two in this example.)

XML Request:
http://lapi.transitchicago.com/api/1.0/ttfollow.aspx?key=e3875818a474304&runnumber=123

XML Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ctatt>
  <tmst>20130515 14:11:17</tmst>
  <errCd>0</errCd>
  <errNm />
  <position>
    <lat>41.97776</lat>
    <lon>-87.77567</lon>
    <heading>299</heading>
  </position>
  <eta>
    <staId>40010</staId>
    <stpId>30001</stpId>
    <staNm>Austin</staNm>
    <stpDe>Austin to O'Hare</stpDe>
    <rn>123</rn>
    <rt>Blue Line</rt>
    <destSt>30171</destSt>
    <destNm>O'Hare</destNm>
    <trDr>1</trDr>
    <prdt>20130515 14:10:23</prdt>
    <arrT>20130515 14:11:23</arrT>
    <isApp>1</isApp>
    <isSch>0</isSch>
    <isDly>0</isDly>
    <isFlt>0</isFlt>
    <flags />
  </eta>
  <eta>
    <staId>40970</staId>
    <stpId>30187</stpId>
    <staNm>Cicero</staNm>
    <stpDe>Cicero to O'Hare</stpDe>
    <rn>123</rn>
```

```
      <rt>Blue Line</rt>
      <destSt>30171</destSt>
      <destNm>O'Hare</destNm>
      <trDr>1</trDr>
      <prdt>20130515 14:10:23</prdt>
      <arrT>20130515 14:15:23</arrT>
      <isApp>0</isApp>
      <isSch>0</isSch>
      <isDly>0</isDly>
      <isFlt>0</isFlt>
      <flags />
    </eta>
</ctatt>
```

JSON Request:
http://lapi.transitchicago.com/api/1.0/ttfollow.aspx?key=e3875818a474304&runnumber=830&outputType=JSON

JSON Response (for run 830):


```
{
   "ctatt":{
      "tmst":"2015-04-30T20:28:37",
      "errCd":"0",
      "errNm":null,
      "position":{
         "lat":"42.01588",
         "lon":"-87.66909",
         "heading":"310"
      },
      "eta":[
         {
            "staId":"40900",
            "stpId":"30173",
            "staNm":"Howard",
            "stpDe":"Howard (Red Line Terminal Arrival)",
            "rn":"830",
            "rt":"Red Line",
            "destSt":"30173",
            "destNm":"Howard",
            "trDr":"1",
            "prdt":"2015-04-30T20:27:58",
            "arrT":"2015-04-30T20:28:58",
            "isApp":"1",
            "isSch":"0",
            "isDly":"0",
            "isFlt":"0",
            "flags":null
         }
```

```
        ]
    }
}
```

Locations API

Description

This API produces a list of in-service trains and basic info and their locations for one or more specified 'L' routes.

Each separate entry describes a single train and provides coordinate, geospatial heading, certain train attributes and next stop information.

Base URL

lapi.transitchicago.com/api/1.0/ttpositions.aspx

Parameters:

Use URL query string method.

Name	Value	Description
rt	**Train route(s)** (required)	Allows you to specify one or more routes for which you'd like train location information.
key	**Alphanumeric API key** (required)	Your unique API key, assigned to you after agreeing to DLA and requesting a key be generated for you.
outputType	**"JSON"** (optional)	If outputType=JSON is added to query string, API response will be formatted as JSON instead of XML

Response fields:

Name	Description
ctatt	Root element
./tmst	Shows time when response was generated in format: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./errCd	Numeric error code (see appendices)
./errNm	Textual error description/message (see appendices)
./route name=	Container element (one per route in response), name attribute indicates route per GTFS-matching route identifiers (see appendices)
./../train	Container element (one per train in response)

./../rn	Run number
./../destSt	GTFS unique stop ID where this train is expected to ultimately end its service run (experimental and supplemental only—see note below)
./../destNm	Friendly destination description (see note below)
./../trDr	Numeric train route direction code (see appendices)
./../nextStaId	Next station ID (parent station ID matching GTFS)
./../nextStpId	Next stop ID (stop ID matching GTFS)
./../nextStaNm	Proper name of next station
./../prdt	Date-time format stamp for when the prediction was generated: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./../arrT	Date-time format stamp for when a train is expected to arrive/depart: yyyyMMdd HH:mm:ss (24-hour format, time local to Chicago)
./../isApp	Indicates that Train Tracker is now declaring “Approaching” or “Due” on site for this train
./../isDly	Boolean flag to indicate whether a train is considered “delayed” in Train Tracker
./../flags	Train flags (not presently in use)
./../lat	Latitude position of the train in decimal degrees
./../lon	Longitude position of the train in decimal degrees
./../heading	Heading, expressed in standard bearing degrees (0 = North, 90 = East, 180 = South, and 270 = West; range is 0 to 359, progressing clockwise)

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="ctatt">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="tmst" type="xs:string" />
                <xs:element name="errCd" type="xs:unsignedByte" />
                <xs:element name="errNm" />
                <xs:element maxOccurs="unbounded" name="route">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element maxOccurs="unbounded" name="train">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="rn" type="xs:unsignedShort" />
                                        <xs:element name="destSt" type="xs:unsignedShort" />
                                        <xs:element name="destNm" type="xs:string" />
                                        <xs:element name="trDr" type="xs:unsignedByte" />
                                        <xs:element name="nextStaId" type="xs:unsignedShort" />
                                        <xs:element name="nextStpId" type="xs:unsignedShort" />
                                        <xs:element name="nextStaNm" type="xs:string" />
                                        <xs:element name="prdt" type="xs:string" />
                                        <xs:element name="arrT" type="xs:string" />
                                        <xs:element name="isApp" type="xs:unsignedByte" />
                                        <xs:element name="isDly" type="xs:unsignedByte" />
                                        <xs:element name="flags" type="xs:string" />
                                        <xs:element name="lat" type="xs:decimal" />
```

```
                                        <xs:element name="lon" type="xs:decimal" />
                                        <xs:element name="heading" type="xs:unsignedShort" />
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="name" type="xs:string" use="required" />
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Example:

XML Request:

http://lapi.transitchicago.com/api/1.0/ttpositions.aspx?key=4ba28f6b2b8843bf9cef1c0fcc05f874&rt=red

XML Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ctatt>
  <tmst>20130610 15:00:21</tmst>
  <errCd>0</errCd>
  <errNm />
  <route name="red">
    <train>
      <rn>804</rn>
      <destSt>30173</destSt>
      <destNm>Howard</destNm>
      <trDr>1</trDr>
      <nextStaId>41400</nextStaId>
      <nextStpId>30269</nextStpId>
      <nextStaNm>Roosevelt</nextStaNm>
      <prdt>20130610 14:58:48</prdt>
      <arrT>20130610 14:59:48</arrT>
      <isApp>1</isApp>
      <isDly>0</isDly>
      <flags />
      <lat>41.86579</lat>
      <lon>-87.62736</lon>
      <heading>358</heading>
    </train>
    <train>
      <rn>808</rn>
      <destSt>30173</destSt>
      <destNm>Howard</destNm>
      <trDr>1</trDr>
```

```
        <nextStaId>40510</nextStaId>
        <nextStpId>30099</nextStpId>
        <nextStaNm>Garfield</nextStaNm>
        <prdt>20130610 14:58:03</prdt>
        <arrT>20130610 15:00:03</arrT>
        <isApp>0</isApp>
        <isDly>0</isDly>
        <flags />
        <lat>41.78697</lat>
        <lon>-87.6187</lon>
        <heading>81</heading>
      </train>
   </route>
</ctatt>
```

JSON Request:

http://lapi.transitchicago.com/api/1.0/ttpositions.aspx?key=4ba28f6b2b8843bf9cef1c0fcc05f874&rt=red&outputType=JSON

JSON Response:


```
{
   "ctatt":{
      "tmst":"2015-04-30T20:29:44",
      "errCd":"0",
      "errNm":null,
      "route":[
         {
            "@name":"red",
            "train":[
               {
                  "rn":"827",
                  "destSt":"30173",
                  "destNm":"Howard",
                  "trDr":"1",
                  "nextStaId":"40650",
                  "nextStpId":"30125",
                  "nextStaNm":"North/Clybourn",
                  "prdt":"2015-04-30T20:29:24",
                  "arrT":"2015-04-30T20:31:24",
                  "isApp":"0",
                  "isDly":"0",
                  "flags":null,
                  "lat":"41.90383",
                  "lon":"-87.63685",
                  "heading":"269"
               }
            ]
         }
```

```
        ]
    }
}
```

Appendix A: Route ID Quick Reference

CTA 'L'

'L' routes (rapid transit train services) are identified as follows:

- Red = Red Line (Howard-95th/Dan Ryan service)
- Blue = Blue Line (O'Hare-Forest Park service)
- Brn = Brown Line (Kimball-Loop service)
- G = Green Line (Harlem/Lake-Ashland/63rd-Cottage Grove service)
- Org = Orange Line (Midway-Loop service)
- P = Purple Line (Linden-Howard shuttle service)
- Pink = Pink Line (54th/Cermak-Loop service)
- Y = Yellow Line (Skokie-Howard [Skokie Swift] shuttle service)

Note: In the separate Customer Alerts API, alerts that apply specifically to the Purple Line Express (but not Purple Line Local/Shuttle service north of Howard) will use the additional route designator "Pexp". When integrating Customer Alert information into your code project, be sure to account that alerts applying to the Purple Line may have the route designator "P" or "Pexp" (or both).

Appendix B: Station IDs

Each bus or train stop on the CTA system, as you'll see if you look at the "stops" table in our Google Transit Feed Specification feed, has its own unique identifier. This helps to power trip planners such as the Google Maps transit directions capability in identifying individual locations and paths where vehicles make stops along a route in each direction.

Note, however, that in the GTFS data, most train stations have three entries in the stops table—one in each direction, with a third entry that covers the entire station facility, known as the "parent stop." We've numbered our stops differently, using the following convention:

0-29999 = Bus stops
30000-39999 = Train stops
40000-49999 = Train stations (parent stops)

The API accepts and responds with both train stop IDs and station IDs to allow you maximum flexibility in how you build your application.

More Help

A packed, quick reference stop list is available for download here:
Downloadable Quick Reference (.zip)

Example from GTFS

For example, Southport, on the Brown Line has three entries in our GTFS table (only relevant rows shown here):

```
stop_id,stop_code,stop_name,stop_lat,stop_lon,location_type,parent_station,wheelchair_boarding
30070,,"Southport",41.943744,-87.663619,0,40360,1
30071,,"Southport",41.943744,-87.663619,0,40360,1
40360,,"Southport",41.943744,-87.663619,1,,1
```

The first two represent specific stops in GTFS—one for each direction from the Southport station (toward Loop or toward Kimball).

The third entry is the associated parent station GTFS, which represents the entire station facility known as "Southport" inside of which these separate "stops" are grouped.

Note that while Southport's parent entry and individual stop entries all have the same basic attributes, this may vary in some stations for map clarity and to assist trip planners (particularly where bus stops also reference a parent station at a larger transfer facility like the transit center at Jefferson Park).

How these look in the API

In the Arrivals API, for example, responses for Southport might include either:

```
<staId>40360</staId>
<stpId>30071</stpId>
<staNm>Southport</staNm>
<stpDe>Service toward Loop</stpDe>
```

Or

```
<staId>40360</staId>
<stpId>30070</stpId>
<staNm>Southport</staNm>
<stpDe>Service toward Kimball</stpDe>
```

This allows you to reference GTFS for a variety of things, and also provides you additional descriptive information which helps you explain results to your customers, depending on how you wish to present them.

Parent Stop ID Quick Reference

This is a list of the parent stops and their associated ID# in GTFS. (Note: In addition to this quick reference, a machine readable CSV of all stops and parent stops is contained within the GTFS data, in stops.txt.)

Descriptive Station Name		
18th (Pink)	Ashland (Green & Pink) 41700	Cicero (Blue) 40490
35th/Archer (Orange)	Ashland (Orange) 40930	Cicero (Green) 41290
35th-Bronzeville-IIT (Green)	Ashland/63rd (Green) 41320	Cicero (Pink) 40580
43rd (Green)	Austin (Blue) 40010	Clark/Division (Red) 41420
47th (Green)	Austin (Green) 41280	Clark/Lake (Blue, Brown, Green, Orange, Purple & Pink) 40060
47th (Red)	Belmont (Blue) 41320	Clinton (Blue) 40430
51st (Green)	Belmont (Red, Brown & Purple) 41320	Clinton (Green & Pink) 41160
54th/Cermak (Pink)	Berwyn (Red) 40340	Conservatory (Green) 41670
63rd (Red)	Bryn Mawr (Red) 40880	Cottage Grove (Green) 41360
69th (Red)	California (Blue) 40570	Cumberland (Blue) 40550
79th (Red)	California (Green) 41240	Damen (Blue) 40590
87th (Red)	California (Pink) 40810	Damen (Brown) 41290
95th/Dan Ryan (Red)	Central (Green) 40450	Damen (Pink) 40830
95th/Ran Ryan (Red)	Central (Purple) 41460	Davis (Purple) 40700
Adams/Wabash (Brown, Green, Orange, Pink & Purple)	Central Park (Pink) 40380	Dempster (Purple) 41080
Addison (Blue)	Cermak-Chinatown (Red) 41000	Dempster-Skokie (Yellow) 41690
Addison (Brown)	Cermak-McCormick Place (Green) 41340	Diversey (Brown & Purple) 41440
Addison (Red)	Chicago (Blue) 41660	Division (Blue) 40370
Argyle (Red)	Chicago (Brown & Purple) 41290	Forest Park (Blue) 40390
Armitage (Brown & Purple)	Chicago (Red) 41450	Foster (Purple) 41220
		Francisco (Brown) 41460

Fullerton (Red, Brown & Purple	Lake (Red)	Purple & Pink) 41660
Garfield (Green)	Laramie (Green) 40510	Racine (Blue)
Garfield (Red)	LaSalle (Blue)	Randolph/Wabash (Brown,
Grand (Blue)	LaSalle/Van Buren (Brown,	Green, Orange, Pink & Purple) 40160
Grand (Red)	Orange, Purple & Pink) 40830	Ridgeland (Green)
Granville (Red)	Lawrence (Red)	Rockwell (Brown) 40870
Halsted (Green)	Linden (Purple) 40910	Roosevelt (Red, Orange & Green)
Halsted (Orange)	Logan Square (Blue)	Rosemont (Blue) 41020
Harlem (Blue - Forest Park Br.)	Loyola (Red) 40980	Sedgwick (Brown & Purple)
Harlem (Blue - O'Hare Br.)	Main (Purple)	Sheridan (Red) 40270
Harlem/Lake (Green)	Merchandise Mart (Brown &	South Boulevard (Purple)
Harold Washington Library-	Purple) 40850	Southport (Brown)
State/Van Buren (Brown, Orang	Midway (Orange)	Sox-35th (Red)
Purple & Pink)	Monroe (Blue)	State/Lake (Brown, Green,
Harrison (Red)	Monroe (Red)	Orange, Pink & Purple) 41090
Howard (Red, Purple & Yellow)	Montrose (Blue) 40930	Thorndale (Red)
Illinois Medical District (Blue)	Montrose (Brown)	UIC-Halsted (Blue) 41503
Indiana (Green)	Morgan (Green & Pink) 40800	Washington (Blue)
Irving Park (Blue)	Morse (Red)	Washington/Wabash (Brown,
Irving Park (Brown)	North/Clybourn (Red) 41460	Green, Orange, Pink & Purple) 40650
Jackson (Blue)	Noyes (Purple)	40400
Jackson (Red)	Oak Park (Blue) 40350	Washington/Wells (Brown,
Jarvis (Red)	Oak Park (Green)	Orange, Purple & Pink)
Jefferson Park (Blue)	Oakton-Skokie (Yellow) 41280	Wellington (Brown & Purple) 41210
Kedzie (Brown)	O'Hare (Blue)	Western (Blue - Forest Park Br.)
Kedzie (Green)	Paulina (Brown) 41070	Western (Blue - O'Hare Br.) 40310
Kedzie (Orange)	Polk (Pink)	Western (Brown)
Kedzie (Pink)	Pulaski (Blue) 41040	Western (Orange) 40930
Kedzie-Homan (Blue)	Pulaski (Green)	Western (Pink)
Kimball (Brown)	Pulaski (Orange) 41200	Wilson (Red) 40540
King Drive (Green)	Pulaski (Pink)	40150
Kostner (Pink)	Quincy/Wells (Brown, Orange,	40040

Individual Stop IDs Quick Reference

Downloadable quick reference

In addition to the stop IDs that represent a whole station, each place where a train stops has its own identifier. You may find these relevant as well. (Note: In addition to this quick reference, a machine readable CSV of all stops and parent stops is contained within the GTFS data, in stops.txt.)

Stop ID	DIR	Stop Name	Parent Stop ID

30161	E	18th (Loop-bound)	40830
30162	W	18th (54th/Cermak-bound)	40830
30022	N	35th/Archer (Loop-bound)	40120
30023	S	35th/Archer (Midway-bound)	40120
30213	N	35-Bronzeville-IIT (Harlem-bound)	41120
30214	S	35-Bronzeville-IIT (63rd-bound)	41120
30245	N	43rd (Harlem-bound)	41270
30246	S	43rd (63rd-bound)	41270
30210	S	47th (63rd-bound)	41080
30209	N	47th (Harlem-bound)	41080
30237	N	47th (Howard-bound)	41230
30238	S	47th (95th-bound)	41230
30024	N	51st (Harlem-bound)	40130
30025	S	51st (63rd-bound)	40130
30113	E	54th/Cermak (Loop-bound)	40580
30114	W	54th/Cermak (Terminal arrival)	40580
30177	N	63rd (Howard-bound)	40910
30178	S	63rd (95th-bound)	40910
30191	N	69th (Howard-bound)	40990
30192	S	69th (95th-bound)	40990
30046	N	79th (Howard-bound)	40240
30047	S	79th (95th-bound)	40240
30275	N	87th (Howard-bound)	41430
30276	S	87th (95th-bound)	41430
30089	S	95th/Dan Ryan (95th-bound)	40450
30088	N	95th/Dan Ryan (Howard-bound)	40450
30132	S	Adams/Wabash (Inner Loop)	40680
30131	N	Adams/Wabash (Outer Loop)	40680
30240	S	Addison (Forest Pk-bound)	41240
30239	N	Addison (O'Hare-bound)	41240
30277	N	Addison (Kimball-bound)	41440
30278	S	Addison (Loop-bound)	41440
30274	S	Addison (95th-bound)	41420
30273	N	Addison (Howard-bound)	41420
30229	N	Argyle (Howard-bound)	41200
30230	S	Argyle (95th-bound)	41200
30128	S	Armitage (Loop-bound)	40660
30127	N	Armitage (Kimball-Linden-bound)	40660
30033	E	Ashland (Loop-63rd-bound)	40170
30032	W	Ashland (Harlem-54th/Cermak-bound)	40170
30205	N	Ashland (Loop-bound)	41060
30206	S	Ashland (Midway-bound)	41060
30057	W	Ashland/63rd (Terminal arrival)	40290

30056	E	Ashland/63rd (Harlem-bound)	40290
30001	E	Austin (O'Hare-bound)	40010
30002	W	Austin (Forest Pk-bound)	40010
30243	E	Austin (63rd-bound)	41260
30244	W	Austin (Harlem-bound)	41260
30013	S	Belmont (O'Hare Branch) (Forest Pk-bound)	40060
30012	N	Belmont (O'Hare Branch) (O'Hare-bound)	40060
30258	S	Belmont (Loop-bound)	41320
30256	S	Belmont (95th-bound)	41320
30255	N	Belmont (Howard-bound)	41320
30257	N	Belmont (Kimball-Linden-bound)	41320
30067	S	Berwyn (95th-bound)	40340
30066	N	Berwyn (Howard-bound)	40340
30267	N	Bryn Mawr (Howard-bound)	41380
30268	S	Bryn Mawr (95th-bound)	41380
30112	S	California (Forest Pk-bound)	40570
30111	N	California (O'Hare-bound)	40570
30266	W	California (Harlem-bound)	41360
30265	E	California (63rd-bound)	41360
30086	E	California (Loop-bound)	40440
30087	W	California (54th/Cermak-bound)	40440
30055	W	Central (Harlem-bound)	40280
30054	E	Central (63rd-bound)	40280
30242	S	Central (Howard-Loop-bound)	41250
30241	N	Central (Linden-bound)	41250
30151	E	Central Park (Loop-bound)	40780
30152	W	Central Park (54th/Cermak-bound)	40780
30194	S	Cermak-Chinatown (95th-bound)	41000
30193	N	Cermak-Chinatown (Howard-bound)	41000
30382	S	Cermak-McCormick Place (63rd-bound)	41690
30381	N	Cermak-McCormick Place (Harlem-bound)	41690
30271	N	Chicago (O'Hare-bound)	41410
30272	S	Chicago (Forest Pk-bound)	41410
30138	S	Chicago (Loop-bound)	40710
30137	N	Chicago (Kimball-Linden-bound)	40710
30279	N	Chicago (Howard-bound)	41450
30280	S	Chicago (95th-bound)	41450
30187	E	Cicero (O'Hare-bound)	40970
30188	W	Cicero (Forest Pk-bound)	40970
30009	W	Cicero (Harlem-bound)	40480
30094	E	Cicero (63rd-bound)	40480
30083	W	Cicero (54th/Cermak-bound)	40420
30082	E	Cicero (Loop-bound)	40420

30122	S	Clark/Division (95th-bound)	40630
30121	N	Clark/Division (Howard-bound)	40630
30074	E	Clark/Lake (Inner Loop)	40380
30075	W	Clark/Lake (Outer Loop)	40380
30374	S	Clark/Lake (Forest Pk-bound)	40380
30375	N	Clark/Lake (O'Hare-bound)	40380
30085	W	Clinton (Forest Pk-bound)	40430
30084	E	Clinton (O'Hare-bound)	40430
30222	W	Clinton (Harlem-54th/Cermak-bound)	41160
30221	E	Clinton (Loop-63rd-bound)	41160
30291	E	Conservatory (63rd-bound)	41670
30292	W	Conservatory (Harlem-bound)	41670
30140	W	Cottage Grove (Harlem-bound)	40720
30139	E	Cottage Grove (Terminal arrival)	40720
30044	N	Cumberland (O'Hare-bound)	40230
30045	S	Cumberland (Forest Pk-bound)	40230
30116	S	Damen/Milwaukee (Forest Pk-bound)	40590
30115	N	Damen/Milwaukee (O'Hare-bound)	40590
30019	S	Damen (Loop-bound)	40090
30018	N	Damen (Kimball-bound)	40090
30041	W	Damen (54th/Cermak-bound)	40210
30040	E	Damen (Loop-bound)	40210
30010	N	Davis (Linden-bound)	40050
30011	S	Davis (Howard-Loop-bound)	40050
30133	N	Dempster (Linden-bound)	40690
30134	S	Dempster (Howard-Loop-bound)	40690
30026	N	Dempster-Skokie (Arrival)	40140
30027	S	Dempster-Skokie (Howard-bound)	40140
30103	N	Diversey (Kimball-Linden-bound)	40530
30104	S	Diversey (Loop-bound)	40530
30062	N	Division (O'Hare-bound)	40320
30063	S	Division (Forest Pk-bound)	40320
30077	W	Forest Park (Terminal Arrival)	40390
30076	E	Forest Park (O'Hare-bound)	40390
30101	N	Foster (Linden-bound)	40520
30102	S	Foster (Howard-Loop-bound)	40520
30168	S	Francisco (Loop-bound)	40870
30167	N	Francisco (Kimball-bound)	40870
30234	S	Fullerton (95th-bound)	41220
30236	S	Fullerton (Loop-bound)	41220
30235	N	Fullerton (Kimball-Linden-bound)	41220
30233	N	Fullerton (Howard-bound)	41220
30100	S	Garfield (63rd-bound)	40510

30099	N	Garfield (Harlem-bound)	40510
30224	S	Garfield (95th-bound)	41170
30223	N	Garfield (Howard-bound)	41170
30096	S	Grand/Milwaukee (Forest Pk-bound)	40490
30095	N	Grand/Milwaukee (O'Hare-bound)	40490
30064	N	Grand/State (Howard-bound)	40330
30065	S	Grand/State (95th-bound)	40330
30148	S	Granville (95th-bound)	40760
30147	N	Granville (Howard-bound)	40760
30183	E	Halsted/63rd (Harlem-bound)	40940
30184	W	Halsted/63rd (Ashland-bound)	40940
30216	S	Halsted (Midway-bound)	41130
30215	N	Halsted (Loop-bound)	41130
30189	E	Harlem (Forest Pk Branch) (O'Hare-bound)	40980
30190	W	Harlem (Forest Pk Branch) (Terminal arrival)	40980
30146	S	Harlem (O'Hare Branch) (Forest Pk-bound)	40750
30145	N	Harlem (O'Hare Branch) (O'Hare-bound)	40750
30003	E	Harlem (63rd-bound)	40020
30004	W	Harlem (Terminal arrival)	40020
30165	E	H.W. Library (Outer Loop)	40850
30166	W	H.W. Library (Inner Loop)	40850
30286	S	Harrison (95th-bound)	41490
30285	N	Harrison (Howard-bound)	41490
30173	N	Howard (Terminal arrival)	40900
30176	S	Howard (Terminal arrival)	40900
30175	N	Howard (Linden, Skokie-bound)	40900
30174	S	Howard (95th-Bound)	40900
30157	E	Illinois Medical District (O'Hare-bound)	40810
30158	W	Illinois Medical District (Forest Pk-bound)	40810
30059	S	Indiana (63rd-bound)	40300
30058	N	Indiana (Harlem-bound)	40300
30107	N	Irving Park (O'Hare Branch) (O'Hare-bound)	40550
30108	S	Irving Park (O'Hare Branch) (Forest Pk-bound)	40550
30282	S	Irving Park (Loop-bound)	41460
30281	N	Irving Park (Kimball-bound)	41460
30014	N	Jackson/Dearborn (O'Hare-bound)	40070
30015	S	Jackson/Dearborn (Forest Pk-bound)	40070
30110	S	Jackson/State (95th-bound)	40560
30109	N	Jackson/State (Howard-bound)	40560
30227	N	Jarvis (Howard-bound)	41190
30228	S	Jarvis (95th-bound)	41190
30247	N	Jefferson Park (O'Hare-bound)	41280

30248	S	Jefferson Park (Forest Pk-bound)	41280
30226	S	Kedzie (Loop-bound)	41180
30225	N	Kedzie (Kimball-bound)	41180
30207	E	Kedzie (63rd-bound)	41070
30208	W	Kedzie (Harlem-bound)	41070
30219	N	Kedzie (Loop-bound)	41150
30220	S	Kedzie (Midway-bound)	41150
30202	W	Kedzie (54th/Cermak-bound)	41040
30201	E	Kedzie (Loop-bound)	41040
30049	W	Kedzie-Homan (Forest Pk-bound)	40250
30048	E	Kedzie-Homan (O'Hare-bound)	40250
30249	N	Kimball (Terminal arrival)	41290
30250	S	Kimball (Loop-bound)	41290
30217	E	King Drive (Cottage Grove-bound)	41140
30218	W	King Drive (Harlem-bound)	41140
30117	E	Kostner (Loop-bound)	40600
30118	W	Kostner (54th/Cermak-bound)	40600
30290	S	Lake (95th-bound)	41660
30289	N	Lake (Howard-bound)	41660
30135	E	Laramie (63rd-bound)	40700
30136	W	Laramie (Harlem-bound)	40700
30262	W	LaSalle (Forest Pk-bound)	41340
30261	E	LaSalle (O'Hare-bound)	41340
30030	E	LaSalle/Van Buren (Outer Loop)	40160
30031	W	LaSalle/Van Buren (Inner Loop)	40160
30149	N	Lawrence (Howard-bound)	40770
30150	S	Lawrence (95th-bound)	40770
30203	N	Linden (Linden-bound)	41050
30204	S	Linden (Howard-Loop-bound)	41050
30197	N	Logan Square (O'Hare-bound)	41020
30198	S	Logan Square (Forest Pk-bound)	41020
30251	N	Loyola (Howard-bound)	41300
30252	S	Loyola (95th-bound)	41300
30052	N	Main (Linden-bound)	40270
30053	S	Main (Howard-Loop-bound)	40270
30091	S	Merchandise Mart (Loop-bound)	40460
30090	N	Merchandise Mart (Kimball-Linden-bound)	40460
30182	S	Midway (Arrival)	40930
30181	N	Midway (Loop-bound)	40930
30153	N	Monroe (O'Hare-bound)	40790
30154	S	Monroe (Forest Pk-bound)	40790
30211	N	Monroe (Howard-bound)	41090
30212	S	Monroe (95th-bound)	41090

30260	S	Montrose (Forest Pk-bound)	41330
30259	N	Montrose (O'Hare-bound)	41330
30287	N	Montrose (Kimball-bound)	41500
30288	S	Montrose (Loop-bound)	41500
30296	W	Morgan (Harlem-54th/Cermak-bound)	41510
30295	E	Morgan (Loop-63rd-bound)	41510
30020	N	Morse (Howard-bound)	40100
30021	S	Morse (95th-bound)	40100
30125	N	North/Clybourn (Howard-bound)	40650
30126	S	North/Clybourn (95th-bound)	40650
30079	S	Noyes (Howard-Loop-bound)	40400
30078	N	Noyes (Linden-bound)	40400
30035	W	Oak Park (Forest Pk-bound)	40180
30034	E	Oak Park (O'Hare-bound)	40180
30263	E	Oak Park (63rd-bound)	41350
30264	W	Oak Park (Harlem-bound)	41350
30298	S	Oakton-Skokie (Howard-bound)	41680
30297	N	Oakton-Skokie (Dempster-Skokie-bound)	41680
30172	S	O'Hare (Forest Pk-bound)	40890
30171	N	O'Hare (Terminal Arrival)	40890
30254	S	Paulina (Loop-bound)	41310
30253	N	Paulina (Kimball-bound)	41310
30200	W	Polk (54th/Cermak-bound)	41030
30199	E	Polk (Loop-bound)	41030
30180	W	Pulaski (Forest Pk-bound)	40920
30179	E	Pulaski (O'Hare-bound)	40920
30005	E	Pulaski (63rd-bound)	40030
30006	W	Pulaski (Harlem-bound)	40030
30185	N	Pulaski (Loop-bound)	40960
30186	S	Pulaski (Midway-bound)	40960
30028	E	Pulaski (Loop-bound)	40150
30029	W	Pulaski (54th/Cermak-bound)	40150
30007	N	Quincy/Wells (Inner Loop)	40040
30008	S	Quincy/Wells (Outer Loop)	40040
30093	W	Racine (Forest Pk-bound)	40470
30092	E	Racine (O'Hare-bound)	40470
30120	W	Ridgeland (Harlem-bound)	40610
30119	E	Ridgeland (63rd-bound)	40610
30196	S	Rockwell (Loop-bound)	41010
30195	N	Rockwell (Kimball-bound)	41010
30081	S	Roosevelt (Midway-63rd-bound)	41400
30080	N	Roosevelt (Loop-Harlem-bound)	41400
30269	N	Roosevelt (Howard-bound)	41400

30270	S	Roosevelt (95th-bound)	41400
30159	N	Rosemont (O'Hare-bound)	40820
30160	S	Rosemont (Forest Pk-bound)	40820
30155	N	Sedgwick (Kimball-Linden-bound)	40800
30156	S	Sedgwick (Loop-bound)	40800
30017	S	Sheridan (95th-bound)	40080
30016	N	Sheridan (Howard-bound)	40080
30293	N	Sheridan (Howard-Linden-bound)	40080
30294	S	Sheridan (Loop-bound)	40080
30163	N	South Blvd (Linden-bound)	40840
30164	S	South Blvd (Howard-Loop-bound)	40840
30070	N	Southport (Kimball-bound)	40360
30071	S	Southport (Loop-bound)	40360
30036	N	Sox-35th (Howard-bound)	40190
30037	S	Sox-35th (95th-bound)	40190
30051	W	State/Lake (Outer Loop)	40260
30050	E	State/Lake (Inner Loop)	40260
30170	S	Thorndale (95th-bound)	40880
30169	N	Thorndale (Howard-bound)	40880
30069	W	UIC-Halsted (Forest Pk-bound)	40350
30068	E	UIC-Halsted (O'Hare-bound)	40350
30073	S	Washington (Forest Pk-bound)	40370
30072	N	Washington (O'Hare-bound)	40370
30383	N	Washington/Wabash (Outer Loop)	41700
30384	S	Washington/Wabash (Inner Loop)	41700
30141	N	Washington/Wells (Inner Loop)	40730
30142	S	Washington/Wells (Outer Loop)	40730
30231	N	Wellington (Kimball-Linden-bound)	41210
30232	S	Wellington (Loop-bound)	41210
30042	E	Western (Forest Pk Branch) (O'Hare-bound)	40220
30043	W	Western (Forest Pk Branch) (Forest Pk-bound)	40220
30129	N	Western (O'Hare Branch) (O'Hare-bound)	40670
30130	S	Western (O'Hare Branch) (Forest Pk-bound)	40670
30284	S	Western (Loop-bound)	41480
30283	N	Western (Kimball-bound)	41480
30061	S	Western (Midway-bound)	40310
30060	N	Western (Loop-bound)	40310
30143	E	Western (Loop-bound)	40740
30144	W	Western (54th/Cermak-bound)	40740
30106	S	Wilson (95th-bound)	40540
30105	N	Wilson (Howard-bound)	40540
30385	S	Wilson (Loop-bound)	40540

30386	N	Wilson (Linden-bound)	40540

Appendix C: Route Direction Code Quick Reference

In the Arrivals API response, you'll see a train direction ("trDr") element. These values represent what you might describe as an "operational" direction—it's not expressive of the physical direction of a train at its current location so much as the big-picture route direction (even though it often coincides).

How a train's direction is defined here loosely translates to a "northbound" (trDr=1) or "southbound" (trDr=5) direction, it can be a little tricky to imagine how that applies to routes such as the Blue Line.

Also, note that this value represents the current operational direction of the train as of when the prediction was generated, not which operational direction it'll have once it reaches a given station that it's predicted to reach. For example, an Orange Line train that's at Halsted on its way to the Loop will be shown as such, even for predictions at Harold Washington Library-State/Van Buren by which time it'll have changed to being a train to Midway the moment it enters the Loop.

Here's a quick reference to help you understand what these values mean, on a per-route basis:

Red Line
1 = Howard-bound
5 = 95th/Dan Ryan-bound

Blue Line
1 = O'Hare-bound
5 = Forest Park-bound

Brown Line
1 = Kimball-bound
5 = Loop-bound

Green Line
1 = Harlem/Lake-bound
5 = Ashland/63rd- or Cottage Grove-bound (toward 63rd St destinations)

Orange Line
1 = Loop-bound
5 = Midway-bound

Purple Line
1 = Linden-bound
5 = Howard- or Loop-bound

Pink Line
1 = Loop-bound
5 = 54th/Cermak-bound

Yellow Line
1 = Skokie-bound
5 = Howard-bound

Appendix D: Insight into Polishing Your Output from the Experts

While the raw information alone is powerful, it's helpful to interpret results and present them in such a way that can make them more meaningful for your customers.

Delays

The CTA Train Tracker service looks at how long it's been since a train has moved from one track circuit to the next and identifies delays if a train appears to not be moving.

The "isDly" element is an expression of whether or not we've detected the likelihood that a train is delayed. If the value of isDly = 1, then consider indicating that the train is delayed rather than simply representing the last prediction, which might be growing stale (which you could compare timestamps to determine, independently).

Schedule faults

The isFlt element in the results indicates what we call a "schedule fault" in the context of Train Tracker. A fault on an ETA that is schedule-based (isSch=1) indicates that the scheduled arrival time given might not be feasible to serve due to the lack of a scheduled departure having occurred. Our system is designed to do some math in order to calculate whether or not a scheduled arrival is feasible based on minimum travel times from a terminal to where the arrival is being estimated for.

Note that this doesn't necessarily mean that there are delays or that service is not good at the time—it only means that a train didn't leave when the planned schedule had provisioned. Transit systems are complex and delays are sometimes unavoidable; our transportation managers use a variety of strategies, including making on-the-go schedule modifications, to maintain service levels (particularly during peak periods when trains leave every few minutes) and provide the best possible service. This is normal and provides a better service to our customers.

Events that affect prediction quality

Some construction or unplanned events that affect service (particularly if they cause trains to be routed differently than normal) can affect the quality of predictions in Train Tracker. While we work to improve back-end exception handling, we can alert users that the quality of predictions may be affected by a current event (or altogether unavailable). In certain situations where we know the information may be unreliable, we may temporarily stop offering predictions for all or portions of a route, which will also be reflected in the API. Our Customer Alerts API Alerts Feed contains a special "Train Tracker impact" flag (element "`ttim`") to indicate when works are affecting prediction quality.

Calculating a number of minutes until arrival from this data

To calculate the number of minutes until arrival (so you can say "4 min" instead of 2:35 p.m., for example, we recommend comparing `arrT` to `prdt` – this will give you the number of minutes we calculated from when a train last moved into a new track circuit until when it should get to a station.

The reason for this is because the arrival time value is actually based on how long it should take to get from where the train was when its location was last updated in our prediction database, weighted in such a way to improve prediction accuracy, which happens on a frequent cycle.

The reason we represent this information in date-time form is for added flexibility in defining your own logic, as it allows you to show clock times or make computations as you feel is best for your code project, and can be more meaningful if your app doesn't update very frequently.

Thus, it gives you the control to compare `arrT` to `prdt` if you do have the capability to update frequently, but, if not, you might then also consider other, more advanced scenarios, where you weight that comparison against the age of your last update. (It's your app, so it's really all up to you!)

Additionally, note that output for the `arrT` element, at least at this stage in the beta (we are open to your feedback as developers), is a minimum of 60 seconds from most recent prediction generation (trains disappear from result sets once they actually reach the station).

Due Trains

We show "due" or "approaching" on trains which are expected to arrive shortly because "1 min" is a very short period of time, and encourage you to consider doing the same.

Accessibility

We work very hard to design our Web services to be fully accessible to people with a wide range of levels of ability, including people with limited or no vision, limited mobility, cognitive disabilities and more.

While it's up to you how to implement our data in your product and to what lengths you go to cater to audiences who are less able to interact with technology than others, we strongly encourage you to take into consideration accessibility implications on whatever platform you write for and to make sure your incorporation of public transit information helps the whole public, to the best of your ability.

We encourage you to catch up on accessibility standards such as those laid out in the Federal Section 508 guidelines, the Illinois Information Technology Accessibility Act (IITAA), as well as staying up on modern Web or software accessibility best practices. It'll lead to better products for you and everyone who might benefit from them.

Appendix E: Error Codes

Error codes are given in the event an unexpected request was made or if an error in processing occurred. A good response comes with an error code of 0.

Arrivals API Error Codes

Error Code	Error Name	Error Explanation
0	OK	No error.
100	Required parameter [value] is missing.	The query string does not contain one of the required parameters, currently: "mapid or stpid", "key".
101	Invalid API key	The value for the required parameter "key" is not a valid API key.
102	Maximum Daily CTA Train Tracker API usage exceeded.	The number of successful API Requests using the supplied "key" have exceeded the maximum daily value.
103	Invalid mapid: [value]	At least one of the supplied values for the "mapid" parameter is not valid. The first invalid id is returned.
104	Mapid's need to be integers: [value]	At least one of the supplied values for the "mapid" parameter is not an integer value. The first invalid id is returned.
105	Maximum of mapid's you can request is 4.	A maximum of 4 values may be specified for the parameter "mapid". More than 4 were supplied.
106	Invalid Route Identifier: [value]	At least one of the supplied values for the "rt" parameter is invalid. Supported values are: "Red", "Blue", "Brn", "G", "Org", "P", "Pink", "Y".
107	Maximum of rt's you can request is 4.	A maximum of 4 values may be specified for the parameter "rt". More than 4 were supplied.
108	Invalid stpId specified: [value]	At least one of the supplied values for the "stpId" parameter is invalid. The first invalid value is returned.

109	Maximum of stpid's you can request is 4.	A maximum of 4 values may be specified for the parameter "stpId". More than 4 were supplied.
110	Invalid max specified: [value]	A non-integer value was specified for the "max" parameter.
111	Parameter 'max' must be a positive integer.	A value less than 1 was specified for the "max" parameter. The value must be an integer greater than zero.
112	Stpid's need to be integers: [value]	At least one of the supplied values for the "stpid" parameter is not an integer value. The first invalid id is returned.
500	Invalid parameter: [value]	The query string contains a parameter that is not supported by the train tracker API, currently supported parameters are: "mapid", "key", "rt", "stpid", "max".
900	Server Error	A server error occurred.

Follower API Error Codes

Error Code	Error Name	Error Explanation
100	Required parameter [value] is missing.	One or more of the required parameters is missing. For this API, the required parameters are: “runnumber”, and “key”
101	Invalid API key.	The supplied API key was not a valid API key.
102	Maximum daily Train Tracker API usage exceeded.	The daily usage limit for the supplied key has been exceeded.
500	Invalid parameter [value].	Valid parameters for this API are: “runnumber”, and “key”.

501	No trains with runnumber [value] were found.	The indicated train may have left service or may simply be incorrect.
502	Unable to determine upcoming stops.	The indicated train has an unexpected exit station id, and the system cannot reliably determine which predictions to report.
503	Unable to find predictions	The train exists, however none of the available predictions are for active stations.

Train Locations API Error Codes

Error code	Message	Description
100	Required parameter [value] is missing.	One of the required parameters (rt, key) was not provided.
101	Invalid API key.	The API key given in the parameter 'key' was either not found or inactive.
102	Maximum daily Train Tracker API usage exceeded.	Key usage has exceeded daily limits. Limits are reset at midnight.
106	Invalid route identifier: [value].	Valid route identifiers are: red, blue, brn, g, org, p, pink, y
107	Maximum number of rt's per request is 8.	No more than 8 routes can be issued per request. Note duplicates are counted but not returned.
500	Invalid parameter: [value]	The indicated parameter is not valid. Valid parameters are: rt, key.