

Roadrunner Transportation Systems, Inc.
Tracking Web API Developer's Guide
Documentation Date: 06/11/2018

Tracking Web API Interface

Tracking Web API Overview

The tracking web API will assist customers in receiving detailed information regarding freight shipped through Roadrunner Transportation Systems. The new tracking web API is [REST](#) based and located at <https://webservices.rts.com/TrackWebApi/api/values>. The API now allows tracking via PRO, Pickup Number, BOL and PO. When tracking, you will not have to distinguish between search items by type. However, only authenticated requests will extend the search to include PO and BOL search items. GPS tracking position data, if available, is now included in the API response. All responses from the API are JSON formatted.

Issuing Tracking Requests

All requests are standard HTTP GET REST calls. For example, to search by fictional pro number 123456789 you would issue a GET request to:

<https://webservices.rts.com/TrackWebApi/api/values/123456789>

Additionally, you will be able to search multiple shipments by comma delimiting the values like such, <https://webservices.rts.com/TrackWebApi/api/values/123456789,1111,AAA121>. The search values do not need to be of the same type (mixing PRO's, Pickups, BOLs and PO is acceptable). Any combination of the four search types is possible. To search by BOL and PO the user must be authenticated. The response back will look like the following:

```
"SearchResults": [  
  
  {  
  
    "SearchItem": "31991",  
  
    "Shipment": {  
  
      "DRAvail": true,  
  
      "ProNumber": "439364498",  
  
      "PickupNumber": "23793030",  
  
      "CustomerNumber": "XXXXXX",  
  
      "BOLNumber": "XXXXXX",  
  
      "BOLReceived": true,  
  
      "PODReceived": false,  
  
      "PONumber": "XXXXXX",  
  
    }  
  
  }  
  
]
```

```
"OrderNumber": "XXXXXX",

"OperationalStatus": "PICK",

"Status": "P",

"InspectionAvailable": false,

"ProDateTime": "2017-09-22T00:00:00",

"DeliverDateTime": "2017-09-28T00:00:00",

"SpecInst1": "Some comment",

"SpecInst2": "another comment",

"SpecInst3": "",

"Scac": "RDFS",

"Location": "CHR",

"Dest": "LAX",

"Manifest": null,

"BillToAccount": "",

"Pieces": 1,

"Weight": 145,

"ApptDateTime": "2017-09-28T00:00:00",

"DeliveredDateTime": "2017-09-28T00:00:00",

"ProjectedDeliveryDateTime": "2017-09-29T00:00:00",

"HAWB": null,

"Origin": {

  "Name": "VS AMERICA",

  "Address1": "8810 AIR APRK W DR",
```

```
    "Address2": "STE A",

    "City": "CHARLOTTE",

    "State": "NC",

    "PostalCode": "28214"

  },

  "Consignee": {

    "Name": "CULVER NEWLIN",

    "Address1": "500 E RINCON ST STE 100",

    "Address2": "",

    "City": "CORONA",

    "State": "CA",

    "PostalCode": "92879"

  },

  "PickupTerminal": {

    "TerminalName": "Customer Service",

    "TerminalTollFreePhone": "(855) 776-3567"

  },

  "Comments": [

    {

      "ActivityCode": null,

      "StatusComment": "09/22/2017 12:07 PM Pickup Request Received.",

      "StatusDateTime": "2017-09-22T12:07:00",

      "Status": null,
```

```
    "Reason": null

  },

  {

    "ActivityCode": "PU",

    "StatusComment": "09/22/2017 02:24 PM Shipment was picked up",

    "StatusDateTime": "2017-09-22T14:24:00",

    "Status": null,

    "Reason": null

  },

  {

    "ActivityCode": "CLO",

    "StatusComment": "09/23/2017 04:45 AM Trailer Closed - ready for dispatch",

    "StatusDateTime": "2017-09-23T04:45:00",

    "Status": null,

    "Reason": null

  },

  {

    "ActivityCode": "DSP",

    "StatusComment": "09/23/2017 06:45 AM Trailer dispatched from CHARLOTTE, NC to LOS ANGELES, CA.",

    "StatusDateTime": "2017-09-23T06:45:00",

    "Status": "L1",

    "Reason": "NS"

  },

}
```

```
{

  "ActivityCode": "ARV",

  "StatusComment": "09/25/2017 02:25 PM Trailer arrived at terminal in LOS ANGELES, CA",

  "StatusDateTime": "2017-09-25T14:25:00",

  "Status": "01",

  "Reason": "NS"

},

{

  "ActivityCode": "UNL",

  "StatusComment": "09/26/2017 12:01 AM Trailer unloaded at terminal in LOS ANGELES, CA",

  "StatusDateTime": "2017-09-26T00:01:00",

  "Status": null,

  "Reason": null

},

{

  "ActivityCode": "CLO",

  "StatusComment": "09/26/2017 09:15 AM Trailer Closed - ready for dispatch",

  "StatusDateTime": "2017-09-26T09:15:00",

  "Status": null,

  "Reason": null

},

{

  "ActivityCode": "DSP",
```

```
    "StatusComment": "09/26/2017 09:15 AM Trailer dispatched from LOS ANGELES, CA to MIRA LOMA, CA.",
    "StatusDateTime": "2017-09-26T09:15:00",
    "Status": "L1",
    "Reason": "NS"
  },
  {
    "ActivityCode": "ARV",
    "StatusComment": "09/26/2017 11:00 AM Trailer arrived at terminal in MIRA LOMA, CA",
    "StatusDateTime": "2017-09-26T11:00:00",
    "Status": "01",
    "Reason": "NS"
  },
  {
    "ActivityCode": "UNL",
    "StatusComment": "09/27/2017 03:39 AM Trailer unloaded at terminal in MIRA LOMA, CA",
    "StatusDateTime": "2017-09-27T03:39:00",
    "Status": null,
    "Reason": null
  },
  {
    "ActivityCode": "ARQ",
    "StatusComment": "09/27/2017 06:00 AM Appointment Pending",
    "StatusDateTime": "2017-09-27T06:00:00",
```

```
        "Status": null,

        "Reason": null

    },

    {

        "ActivityCode": "OFD",

        "StatusComment": "09/28/2017 08:05 AM Shipment out for delivery",

        "StatusDateTime": "2017-09-28T08:05:00",

        "Status": "OD",

        "Reason": "NS"

    },

    {

        "ActivityCode": "DEL",

        "StatusComment": "09/28/2017 10:00 AM Shipment delivered to consignee",

        "StatusDateTime": "2017-09-28T10:00:00",

        "Status": "D1",

        "Reason": "NS"

    }

],

"Positions": [],

"ReferenceNumbers": [

    {

        "StopID": "XXXXXX",

        "Qual": "PU",
```

```
        "Nbr": "XXXXXX"

    }

],

"SchedArriveEarly": "2017-09-22T12:00:00",

"SchedArriveLate": "2017-09-22T16:30:00",

"ActualDeparture": "2017-09-22T14:24:00",

"OrderDate": "2017-09-22T12:07:00",

"PickedUp": true

},

"DisplayBOL": true,

"DisplayPOD": true,

"DisplayInvoice": true,

"DisplayInspection": true,

"DisplayWI": false

}

]

}
```

When available, industry standard codes will also be included in the comments section represented by “Status” and “Reason”.

Issuing Image Requests

All requests are standard HTTP POST REST calls. For example, to get all four types of images a user would post the following JSON to <https://webservices.rts.com/TrackWebApi/api/images>.

```
[

{

  "ImageType": "POD",
```

```
"ProNumber": 333333333
},
{
  "ImageType": "BOL",
  "ProNumber": 444444444
},
{
  "ImageType": "WI",
  "ProNumber": 555555555
},
{
  "ImageType": "INVOICE",
  "ProNumber": 666666666
}
]
```

There are 4 types of images returned: BOL, Delivery Receipt (POD) , Invoice, and Weight cert. (WI). The user must be authenticated to get BOL, WI, and INVOICE. The response back will look like the following:

```
[
  {
    "Images": [
      {
        "FileName": "POD_333333333_1.jpg",
        "Base64Image": "base64string"
      },
      {
        "FileName": "POD_333333333_2.jpg",
        "Base64Image": "base64string"
      }
    ]
  },
  ]
```

```
"ProNumber": 333333333,
"HasError": false,
"ErrorMessage": null
},
{
  "Images": [
    {
      "FileName": "BOL_444444444_1.jpg",
      "Base64Image": "base64string"
    }
  ],
  "ProNumber": 444444444,
  "HasError": false,
  "ErrorMessage": null
},
{
  "Images": [
    {
      "FileName": "WI_555555555_1.jpg",
      "Base64Image": "base64string"
    }
  ],
  "ProNumber": 555555555,
  "HasError": false,
  "ErrorMessage": null
},
{
  "Images": [
    {
```

```
        "FileName": "INVOICE_666666666_1.jpg",
        "Base64Image": "base64string"
    }
],
"ProNumber": 666666666,
"HasError": false,
"ErrorMessage": null
}
```

All images will be base64 encoded strings and of type JPG.

Nonce Authentication

Overview

To authenticate requests the tracking API utilizes a custom authorization scheme called AMX. The AMX scheme represents an authorization header value consisting of four different parameters. These parameters are as follows:

- **Application Identifier.** This value is pre-computed and given to the client at some earlier time.
- **HMAC SHA256 hashed signature string.** HMAC (Hash-based Message Authentication Code) authentication provides a mechanism for calculating a message authentication code by utilizing the HMAC SHA256 hashing algorithm. This cryptographic algorithm is used to generate unique hashes per request. See [Generating a Signature](#) on how to compute this value.
- **Nonce.** A per request arbitrary value that is used only once.
- **Time stamp** in [Unix Time](#).

To build the AMX header value the four parameter strings listed above need to be concatenated and be formatted as such (braces are just placeholders):

```
{Application Id}:{Hashed Signature}:{Nonce}:{Time Stamp}
```

In addition to the Application Identifier the client will also be issued an API secret key. The API key is used as an initializer to the hashing function that generates the HMAC SHA256 hashed signature. This key should be kept safe and only the client and the server should know the value. Unlike Basic authentication, the API key is never sent over the network. The primary purpose of using HMAC authentication is to verify integrity, authenticity, and identity of the message sender.

Generating a Signature

Build raw signature string

Five parameters are required to build the raw signature string. Some of these parameters are the same as mentioned above. Here, the primary goal is to build the hashed signature portion of the authorization header value. The order in which you calculate each parameter value isn't important but the order in which the signature is built is *very* important. Once all the values have been determined the order in which they are concatenated is as follows (braces are just placeholders):

```
{Application Id}{HTTP Method}{Request URI}{Time Stamp}{Nonce}
```

Parameters:

1. **Application Identifier.** The client will be given an Application Id. When building the raw signature string the Application Identifier must be capitalized. Case will affect the final hashed string and if the client uses different casing than the server the hashes will not match.
2. **HTTP method (GET).** This parameter needs to be capitalized.
3. **Request URI.** The full address of the API call URL encoded. Do not safe encode by changing '+' to '-' and '/' to '_'. Also percent encodings must be capitalized like %2F instead of %2f.
4. **Time stamp** in [Unix Time](#).
5. **Nonce.** A nonce is an arbitrary identifier used to make http requests unique and can only be used once.

Once the raw signature is built it is used, along with the client's API key, to compute the HMAC SHA256 hash string. Finally, the client takes the hashed signature and builds the final authorization string that is stored in the request's authorization header. The final authorization header value has the following structure (braces are just placeholders):

```
{Application Id}:{Hashed signature string}:{Nonce}:{Time Stamp}
```

Below is a complete code example showing a fictional function `GetSignature(...)` which takes as input the client-provided values and returns the hashed signature string.

Example (C#)

```
// INPUT
// appID: 7E437B64-81FD-4EA9-B354-66F10749E5D1
// apiKey: UEBz3dvcMqhKg==
// apiUrl: http://webservices.rrts.com/TrackWebAPI/api/values/159364694
// httpVerb: GET

public string GetSignature(string appID, string apiKey, string apiUrl, HttpVerb httpVerb)
{
    // Get the encoded url
    var requestUri = System.Web.HttpUtility.UrlEncode(apiUrl);
    // requestUri: http%3A%2F%2Fwebservices.rrts.com%2FTrackWebAPI%2Fapi%2Fvalues%2F159364694

    var requestHttpMethod = httpVerb.ToString();
    // requestHttpMethod: GET
}
```

```

// Calculate UNIX time
var epochStart = new DateTime(1970, 01, 01, 0, 0, 0, 0, DateTimeKind.Utc);
var timeSpan = DateTime.UtcNow - epochStart;
var requestTimeStamp = Convert.ToInt64(timeSpan.TotalSeconds).ToString();
// requestTimeStamp: 1511305282

// Create random identifier for each request
var nonce = Guid.NewGuid().ToString("N");
// nonce: f338e9afc1924b54bea54dbfacacb83e

// Create the raw signature string
var signatureRawData = $"{appID.ToUpper()}{requestHttpMethod}{requestUri}{requestTimeStamp}{nonce}";
// signatureRawData: 7E437B64-81FD-4EA9-B354-66F10749E5D1GET
http%3A%2F%2Fwebservicess.com%2FTrackWebAPI%2Fapi%2Fvalues%2F1593646941511305282f338e9afc1924b54bea54dbfacacb83e

var secretKeyByteArray = Encoding.ASCII.GetBytes(apiKey);
var signature = Encoding.UTF8.GetBytes(signatureRawData);
using (var hmac = new HMACSHA256(secretKeyByteArray))
{
    var signatureBytes = hmac.ComputeHash(signature);
    var requestSignatureBase64String = Convert.ToBase64String(signatureBytes);
    // requestSignatureBase64String: eSK1ShSi7kGaaQAh7iaLoGWJZu29gp6Lo9Ct7LQT4k=
    return $"{appID.ToUpper()}: {requestSignatureBase64String}: {nonce}: {requestTimeStamp}";
}
}

// OUTPUT 7E437B64-81FD-4EA9-B354-
66F10749E5D1:eSK1ShSi7kGaaQAh7iaLoGWJZu29gp6Lo9Ct7LQT4k=:f338e9afc1924b54bea54dbfacacb83e:1511305282

```

When the client sends the request, it must set the request's authorization header value. Whichever language is used to send the request it must have the authorization scheme set to the value: "amx" and the value is the string returned from the GetSignature(...) method above. The case of the scheme does not matter.

Retrieving Application ID and API Key

To get your account specific Application Id and API Key, follow the steps below

1. Sign in to freight.rrts.com <https://freight.rrts.com/layouts/15/fba/siteLogin.aspx>
2. Retrieve your Application ID and API Key by navigating to <https://freight.rrts.com/Tools/Tracking/Pages/default.aspx>

Top Activity Codes. These are the most used activity codes and messages that will be coming back. More activity codes may be added or used that are not on this list.

Activity Code	Message
PU	Freight picked up from shipper location
CLO	Trailer closed – ready for dispatch
DSP	Trailer dispatched to destination terminal
ENR	Trailer enroute: <location>
ARV	Trailer arrived at terminal in <location>
UNL	Trailer unloaded at destination terminal
OFD	Shipment out for delivery
APT	Appointment set for delivery to consignee on <date>
RFD	Shipment refused by consignee
ARQ	Appointment Pending
DEL	Shipment delivered to consignee
BRN	Customs
TRF	Released to Delivery
DUD	Projected Delivery Date of <date>
SLC	Sailing Date
WTR	Weather Delay
TDC	Trailer Spotted at customer