

20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Stream API - Getting Started

*Updated: 2018-07-31*

**Overview 3**

**Technology 4** RESTful Web Services 4 GraphQL Web Services 4

**API Data Points 4**

**How to Access 5** API Key 5 IdToken 5 Authentication Challenges 6

**GraphQL API 7** How to Access 7 Success 7 Limitations 7

Throttling 7 Response Size 8 Errors 8 Server Errors 8 Query Errors 8 Partial Response Errors 8 Securing The Key and Token 8

**GraphQL IDE 10** Security 10 Documentation 11 Build & Test 11 IntelliSense 12 School Access 13 Applying to GraphQL API 13

**Feedback 14**

****

20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Overview

This document provides all of the information required in order to access the Stream API. More information on what Stream API is can be found here: https://www.calendar-stream.com/

BigTeams provides each school a unique identifier which we call the SCID.

Events are categorized in our system by one of the following types: Sport, Practice, Scrimmage, School, Other, and Try Outs.

All events are required to have an Event Date and an Official Start Time, which could also be set as TBA.

All events are required to have a location set, either Home, Away, or N/A.

Sport and Practice events are required to have a team association. A team is considered a combination of a Gender, Level, and Sport, each of which have a unique identifier as well.

School and Other events are required to have an Event Title, which is optional for Sport and Practice events. Sport events are required to have at least one opponent.

Facilities (Big Gym, Small Gym, etc) are optional on all events. Additional facility data included are From, To, and Dismissal times.

Transportation (Bus #1, Van #1, etc) is optional on all events. Additional transportation data included are Depart, Return, and Dismissal times.

For a complete list of all the available data, see the schema found in the Documentation section. *.*

**

20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Technology

The Stream API is comprised of RESTful web services and GraphQL web services.

RESTful Web Services

Authentication is required in order to access the web services. Authentication is performed using a POST to a RESTful web service.

GraphQL Web Services

The primary means of data retrieval is available via GraphQL. GraphQL is a query language built specifically for APIs. More information can be found here: http://graphql.org/.

Integrating with GraphQL is accessible via a POST URL which is covered under the How to Access section further below.

The documentation on the GraphQL schema can be found here:

https://developers.schedulestar.com/

Documentation is also part of the same utility used to build and test your queries which is available via a web-based IDE covered under GraphQL IDE.

API Data Points

Below is a list of all endpoints covered in this document for quick reference. POST https://stream.schedulestar.com/auth/ More Information POST https://stream.schedulestar.com/auth/challenge/ More Information POST https://stream.schedulestar.com/graphql/ More Information GET https://stream.schedulestar.com/graphql/ide More Information



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

How to Access

Access to the Stream API requires two pieces of authentication information in order to ensure a safe and secure API exchange.

API Key

An API key will be assigned to you by BigTeams. Your API key is private and should never be shared with anyone. It is up to your implementation to ensure the safety and security of your API key.

The API Key is provided to all web services via a HTTP header:

x-api-key: [your API key]

IdToken

A user and temporary password will be assigned to you by BigTeams based on your primary email address. Once we setup your user account, you will receive an email with the subject “Your Schedule Star temporary password”. You can authenticate this user in order to obtain an IdToken. It is up to your implementation to ensure the safety and security of your IdToken. Please note that you must login with 7 days of receiving your temporary password or it will expire.

Authentication is performed via the following REST API endpoint:

POST https://stream.schedulestar.com/auth/

Headers:

Content-Type: application/json

x-api-key: [your API key]

Request Body (json):

{

"email": "[your email address]",

"password": "[your password]"

}



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

The first time you successfully authenticate you will receive a response body containing a challenge of type “NEW\_PASSWORD\_REQUIRED”. This is simply stating that you must set a new password in order to proceed. Please keep this response body handy as you will need several pieces of information from the response in order to respond to the challenge.

Authentication Challenges

You will need to send a request to another endpoint in order to respond to new password challenge.

POST https://stream.schedulestar.com/auth/challenge/

Headers:

Content-Type: application/json

x-api-key: [your API key]

Request Body (json):

{

"challengeName": "NEW\_PASSWORD\_REQUIRED",

"challengeResponses": {

"USERNAME": "[this is the USER\_ID\_FOR\_SRP value from the authentication response]",

"NEW\_PASSWORD": "[the new password you wish to use]" },

"session": "[this is the Session value from the authentication response]"

}

**NOTE: Passwords must be a minimum of 8 characters and must contain a lowercase letter, an uppercase letter, and a number.**

Once you have successfully responded to the challenge you will receive a response body containing three tokens: AccessToken, RefreshToken, and IdToken. The IdToken is the one we care about right now. Now when you authenticate again via the /auth/ endpoint you will receive these same three tokens.

**NOTE: These tokens expire every so often so you will need to build it into your process to retrieve new tokens once yours has expired.**

****

20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Now that you have your IdToken you can access the other Stream API web services. The IdToken is provided to web services as a HTTP header:

Authorization: [your IdToken]

GraphQL API

How to Access

In order to query for Stream data you will need to make a request to the following API endpoint: POST https://stream.schedulestar.com/graphql/

Headers:

Content-Type: application/json

x-api-key: [your API key]

Authorization: [your IdToken]

Request Body (json):

{"query": "[your GraphQL query]"}

In order to build you GraphQL query we provide a GraphQL IDE which allows you access to the schema documentation and a way to build and test your queries. More information about the IDE can be found on the next page.

Success

Successful request queries will return a HTTP status code of “200 OK” along with a response body containing JSON content.

Limitations

Throttling

Throttling is a way to control the amount of requests so that they do not overload the API making it inaccessible to other consumers



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Requests from your API key are limited to 50 requests per second. If the limit is exceeded you will receive a HTTP status code of “429 Too Many Requests”. Your implementation should check the status code of responses and handle appropriately.

It is recommended that you implement a caching mechanism to help control the number of API requests you are making so that you do not exceed your throttling rate.

Response Size

The response body size is limited to a maximum of 6MB. If the response exceeds this size you will receive an “Internal Server Error” in your response. If you receive this error message, try reducing your limit and use paging or add/change your filter criteria.

Errors

Server Errors

If the request encounters an *unexpected error*, the response will return a HTTP status code in the 500 range. Your implementation should check the status code of responses and handle appropriately.

Query Errors

If you send a request with an invalid query schema you will receive a response back with a HTTP status code of “400 Bad Request” and a response body containing JSON with the error message. Your implementation should check the status code of responses and handle appropriately.

Partial Response Errors

You may encounter a case where the primary response was successful and returns the expected data, however, an underlying call has errored. This will present itself as a successful response with an “errors” section in the JSON response body. Your implementation should check for the presence of the “errors” key and handle appropriately.

Securing The Key and Token

When you are ready to implement calls to the Stream API, it is not recommended to access the API directly via your front end as there is no way to secure the API key and IdToken from prying



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

eyes. If you wish to access the Stream API from the front end of your web-based or native mobile application, you should create a server-side proxy endpoint which handles the Stream API call so that you do not expose your keys and tokens.



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

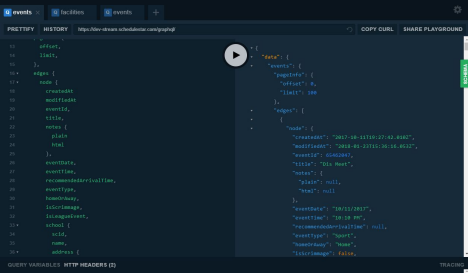
866.448.9438

GraphQL IDE

Stream API comes with an implementation of a web-based GraphQL IDE called GraphQL Playground. Details on GraphQL Playground can be found here:

https://github.com/graphcool/graphql-playground.

The GraphQL Playground provides schema documentation, query building and query testing capabilities all in one convenient location.

You can reach the Stream API IDE here: https://stream.schedulestar.com/graphql/ide 

Security

You will need to provide the x-api-key and Authorization values via the HTTP Headers option in order to access the schema documentation and to build and test your queries.



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

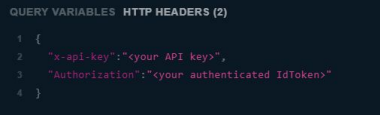
866.448.9438

{

"x-api-key":"<your API key>",

"Authorization":"<your /auth/ IdToken>"

}

*NOTE: You will need to refresh the page after providing the headers so that the utility can load the schema documentation.*

The GraphQL IDE uses the same GraphQL API POST endpoint in order to execute your queries so you know that what you are building will work in your implementation as well.

Documentation

The documentation is available from the right-hand side of the screen and is labeled “Schema”. 

Build & Test

To test your queries, enter your query on the left side. Click the Play button to execute your query and your results will appear on the right side.



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

To get you started, here is a sample query:

query {

events(scid: "VA201866618276915") {

pageInfo {

offset

limit

}

edges {

node {

createdAt

modifiedAt

eventId

title

eventDate

eventTime

eventType

homeOrAway

}

}

}

}

IntelliSense

Within this utility, the query uses Intellisense based on the documented API.



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438



School Access

All query searching requires you provide an “SCID”. The SCID is a unique identifier assigned to each school. You must know the SCID of the school in order to request their data.

If you do not have a list of SCIDs already, you can try out the API with our two testing schools. ● Bedrock - SCID: VA201866618276915

● Blue Snail - SCID: FL334146618276779

Applying to GraphQL API

Once you have a working query built and tested in the GraphQL IDE, you can copy/paste the query into the request body of your GraphQL API request.

{"query": "[your GraphQL query]"}

*You will need to remove any line breaks from the formatted query.*

**

20098 Ashbrook Pl, Suite 155 Ashburn, VA 20147

866.448.9438



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Samples

Here are some sample queries to get you started.

Connection Test

The simplest query you can run to test that you have a valid connection would be the below: {

"query": "{\_\_typename}"

}

Which should provide this result:

{

"data": {

"\_\_typename": "Query"

}

}

If you receive the above result, then you are connecting to the API properly.

Basic Query

If you would like to test actual data retrieval, please try this:

{

"query": "query {events(scid: \"VA201866618276915\") { pageInfo { offset limit } edges { node { createdAt modifiedAt eventId title eventDate eventTime eventType homeOrAway }}}}"

}

Query Variables

You can also pass the parameters in separately, similar to SQL query parameters. Here is the identical query as above but using query variables instead:

{

"query": "query($scid: String!) {events(scid: $scid) { pageInfo { offset limit } edges { node { createdAt modifiedAt eventId title eventDate eventTime eventType homeOrAway }}}}",

"variables": {

"scid": "VA201866618276915"

}

}



20098 Ashbrook Pl, Suite 155

Ashburn, VA 20147

866.448.9438

Feedback

BigTeams is open to any and all feedback. Please email philip@bigteams.com and/or the person you have generally been in contact with.

If any bugs are found, please report them promptly so they can be addressed.