AppNexus Console API Overview & Training

appnexus

Learning Objectives

Intended audience: AppNexus Console networks who are interested in developing an interface on top of our APIs

This training will enable you to use the Console API.

1. Introduction
2. Ways to integrate
3. JSON
4. Methods
5. Authorization
6. Throttling
7. Read-only services
8. Tools
9. Best practices
10. Breaking vs. non-breaking changes
11. In-depth examples

Audience Poll

I would rate myself as an API…

1. Expert
2. Intermediate user
3. Newcomer

appnexus

Introduction

- API philosophy
- Why use our API?
- Onboarding process

Our API Philosophy

APIs play an important role in our technology stack.

What do we use APIs to do?

1. Abstract the display of objects from functions to create, modify and delete them
2. Perform validations in order to prevent incompatible or unsafe changes possible through data base writes.
3. Provide a protective layer to ensure smooth system operation (e.g., throttling to prevent pulling too many reports at once)
4. Allow AppNexus to change how underlying objects are organized without impacting how the data is presented externally

appnexus

Why use our API?

Our UI is built entirely on the APIs, which means that any function available in the UI can be performed through one or many API calls.

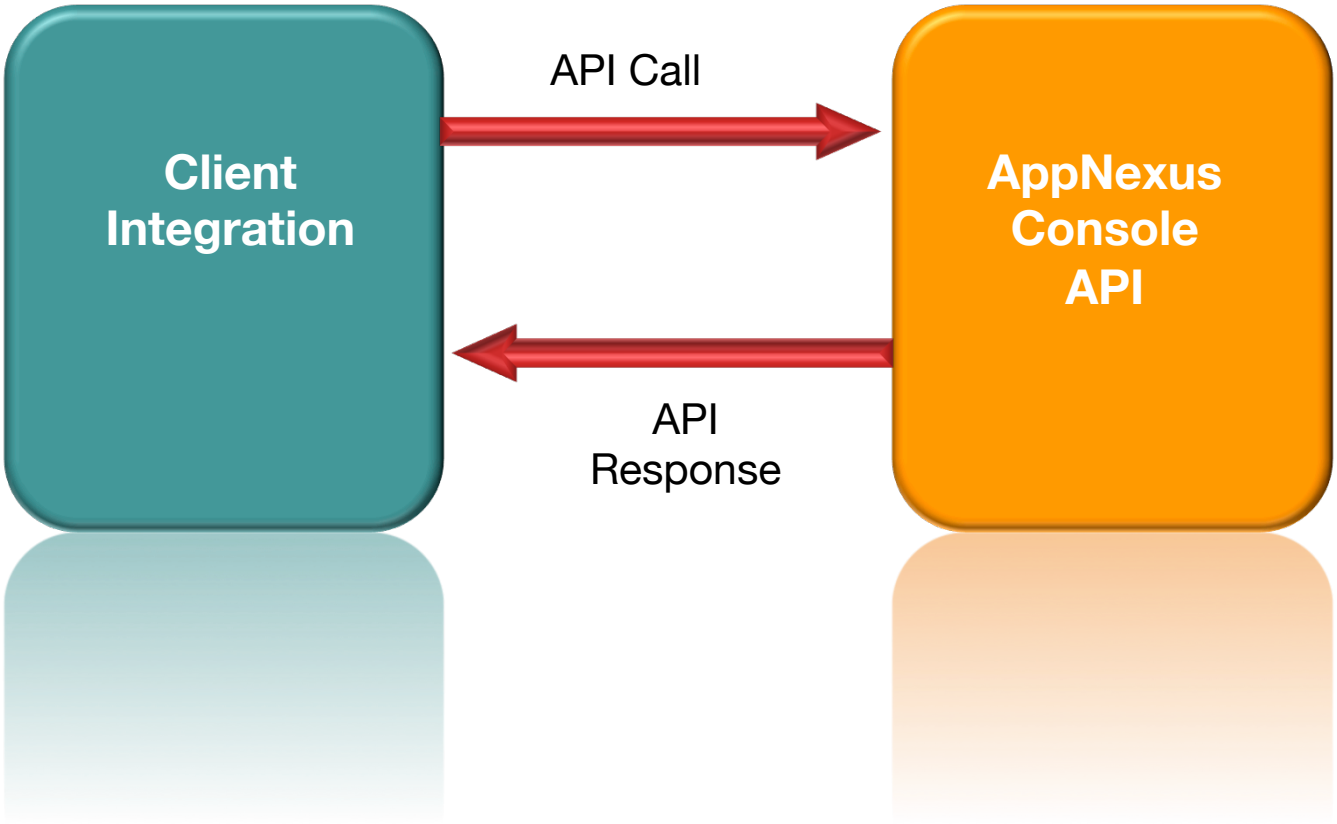
Some example use cases are:

1. Building a new custom user interface
2. Batch creation and updating of campaigns
3. Generating templates for creatives, targeting profiles, etc.
4. Extracting performance data
5. Performing analytics against reporting data (e.g., dashboards)

appnexus

API Interaction

It's easy to plug into our API



appnexus

Process for API Onboarding

In order to ensure that any integration with our APIs goes smoothly, we have developed an onboarding process that consists of:

1. Read all Wiki documentation
2. Discuss use cases with your Account Manager
3. Training sessions as a requirement to access the API
4. Sandbox API access
5. Development
6. Application review sessions to verify the integration approach
7. Production API access
8. Ongoing technical support

Wiki details:
https://wiki.appnexus.com/display/api/API+Onboarding+Process

appnexus

Ways to Integrate

- Ad-hoc
  (via cURL)
- Develop an Interface
  (various languages, we use PHP)

Ad-hoc
(via cURL)

Using the command line, you can issue single requests to the API.

cURL libraries are available which can be used within application.

- Various command line parameters can be used to modify the API request
  - `H`: Specify headers to be passed on the call.
  - `c`: The file where cookie data should be written.
  - `b`: The file from where cookie data should be read.
  - `X`: The method to be used. In our API, the methods are POST, PUT, GET and DELETE. GET it used if no method is specified.
  - `d`: Specify the data to pass for POST and PUT.

Installing cURL

cURL is a command line application which allows you to send and receive data using the HTTP protocol.

cURL supports many network protocols in addition to HTTP.

It can be downloaded at http://curl.haxx.se/download.html

- For PC, download the latest “Win32 Generic” binaries.
  - At the time of this writing, the latest version for Windows is 7.21.3
- For Mac, cURL should already be installed by default.
  - If not, download the latest “Mac OS X” binaries for your architecture.
  - At the time of this writing, the latest version for Mac is 7.21.2.

Develop an Interface
(PHP, Ruby, .NET, etc.)

- Developing an interface on top of our APIs is the ideal way to integrate.

- Programmatic interfaces are less error prone and more consistent than using manual API calls which introduce more human error.

- With an interface, you will be able to perform a number of other functions:
  - Error logging
  - Validation checks
  - Request queuing and throttling
  - Real-time feedback to requests

- JSON manipulation libraries exist for most high-level languages.

JSON

- Structure & Data Types
- Sample JSON
- Libraries for Manipulation & Other Resources

Structure & Data Types

JavaScript Object Notation (JSON) is a way to convey structured data from one system to another.

We use it for it's flexibility, language independence and the nearly ubiquitous support for this data format across systems.

Wiki details:
https://wiki.appnexus.com/display/api/Helpful+JSON+Tools

Objects are made up of attributes that are strings, numbers, Booleans (true/false), arrays and other objects. The following syntax is used:

- `"key": value` assigns the specified value to the key. The value may be a string, number, Boolean, array or another object
- `{...}` denotes an object. The object is defined within the brackets.
- `[...]` denotes an array. The array is a comma-separated list. The list may be made up of strings, numbers, Booleans, objects or sub-arrays. Please note that the array will typically contain a single data type.
- `"..."` denotes a string. Putting numbers or Booleans in quotes will affect they way they are treated by the API.

appnexus

Sample JSON

The general structure of JSON for objects in the AppNexus API is shown to the right.

Note that an object may be made up of (but not limited to) strings, numbers and other objects.

Wiki details:
https://wiki.appnexus.com/display/api/Helpful+JSON+Tools

```
{
  "object_name": {
    "id": 12345,
    "name": "name of the object",
    "array_of_ints": [123, 456, 789],
    "array_of_objs": [
      {
        "id": 123,
        "scope": "all"},
      {
        "id": 456,
        "scope": "list",
        "list_of_ids": [123, 456,
789]}
    ]
  }
}
```

appnexus

Libraries for Manipulation & Other Resources

Many languages, such as PHP, natively support JSON data structures.

Many resources exists for manipulating JSON such as:

- Validators
- Formatters
- Programming libraries

Wiki details:
https://wiki.appnexus.com/display/api/Helpful+JSON+Tools

- A brief, but comprehensive description of JSON can be found at http://json.org
- On the JSON site above, many tools are catalogued, such as:
  - Programming libraries for parsers and validators for almost all common programming languages
  - Web tools which perform parsing and validation
  - The RFC specification for JSON
  - Comparisons between XML and JSON

Methods

- GET
- POST
- PUT
- DELETE

GET Method

The GET method is used to retrieve data.

By calling a service without any filtering, the API will retrieve all accessible items.

Wiki details:
https://wiki.appnexus.com/display/api/API+Semantics

- Specifying an ID through the querystring will return a specific item
- It may be necessary to specify other filters on the querystring
  - Advertiser ID
  - Publisher ID
  - Member ID
  - Etc.
- All API services support the GET method
- If no method is specified, most libraries (cURL included), will default to the GET method.

appnexus

POST Method

The POST method is used to create objects.

The specification of the new object must be sent as part of the POST data.

Wiki details:
https://wiki.appnexus.com/display/api/API+Semantics

- When creating an object, it may be necessary to specify some filters on the querystring, similar to a GET request.
- If successful, the response will contain a status of “OK” as well as the ID for the newly created object

```
{
  “response”: {
    “status”: “OK”,
    “id”: 12345
  }
}
```

- Successful responses will also contain the full specification of an object.

appnexus

PUT Method

The PUT method is used to modify data for existing objects.

The ID of the object to be modified must be specified in the querystring.

The modified parameters of the object must be sent as part of the POST data.

Wiki details:
https://wiki.appnexus.com/display/api/API+Semantics

- When modifying an object, it may be necessary to specify some filters on the querystring, similar to a GET request.
- Successful responses will also contain the full specification of an object.
- It is not necessary to submit the full specification for an object when modifying.
  - In fact, the best practice is to submit **only** those parameters which are being modified.

appnexus

DELETE Method

The DELETE method is used to permanently remove objects (and their child object from AppNexus Console.

If an advertiser is deleted, all of the campaigns, creatives, segments, etc. associated with the advertiser are also deleted.

Wiki details:
https://wiki.appnexus.com/display/api/API+Semantics

- When deleting an object, it may be necessary to specify some filters on the querystring, similar to a GET request.
- Deleting objects may result in a “hard delete”, which completely and irrevocably removes the object from the AppNexus platform.

appnexus

Authorization

- Tokens
- Cookies vs. Headers

Authorization Tokens

Before calling any API services, you must acquire an authorization token, proving you should have access via a username and password.

Wiki details:
https://wiki.appnexus.com/display/api/Authentication

- Call the auth service at the path below

```
http://api.appnexus.com/auth
```

- Pass in the username and password as follows

```
{"auth":{"username": "USERNAME",
"password": "PASSWORD"}}
```

- If the username and password are correct you should receive a response indicating that you are logged in. The authorization token in the response is your session ID and must be passed with every API call. This can be done either through headers or cookies.

```
{"response":{"status": "OK",
"token":
"iva3vnipc8aq1vp7iumn84p5k5"}}
```

appnexus

Cookies vs. Headers

On every call to the API, it is necessary to submit the token assigned by the authorization service.

The token may be transmitted either via HTTP header or via cookie data.

Wiki details:
https://wiki.appnexus.com/display/api/Authentication

- When using headers, use the –H parameter in cURL

```
curl -H "Authorization:
iva3vnipc8aq1vp7iumn84p5k5"
"http://api.appnexus.com/
creative"
```

- If cookies are enabled, the auth service will write the token value to the cookie. Cookies can be enabled in cURL using the –c and –b parameter

```
curl -c cookies -b cookies
"http://api.appnexus.com/
creative"
```

appnexus

Throttling

- Pagination
- Filtering
- Sorting

Pagination

The API limits the response to any request to 100 objects.

Pagination will allow you to use multiple requests to retrieve more than 100 objects.

This is performed via querystring parameters on the request.

Wiki details:
https://wiki.appnexus.com/display/api/API+Semantics

- The following API call will return the first 100 campaigns under an advertiser.

```
http://api.appnexus.com/
campaign?advertiser_id=123
```

- The following API call will return the the next 100 campaigns under that advertiser.

```
http://api.appnexus.com/
campaign?
advertiser_id=123&start_element=
100&num_elements=100
```

- The response will contain information about how many "pages" there are.

```
{"response": {..., "count":384,
"start_element":null,
"num_elements":null, ...}
```

appnexus

Filtering

Filtering allows you to request a subset of objects from the API, reducing the number of “pages” which need to be requested.

This is performed via querystring parameters on the request.

Wiki details:
https://wiki.appnexus.com/display/api/API+Filtering+and+Sorting

- The following call selects campaigns under a specific advertiser and line item by filtering on object IDs

```
http://api.appnexus.com/campaign?
advertiser_id=123&line_item_id=1,2
,3
```

- The call below selects campaigns under a specific advertiser and line item by filtering on object codes

```
http://api.appnexus.com/campaign?
advertiser_code=abc&line_item_code
=li_us
```

- You can filter only recently modified objects using minimum and maximum values

```
http://api.appnexus.com/creative?
min_last_modified=2011-06-20
```

appnexus

Sorting

Sorting will instruct the API to return objects in a particular order.

When sorting paginated results, this eliminates the need to request all objects prior to ordering.

This is performed via querystring parameters on the request.

Wiki details:
https://wiki.appnexus.com/display/api/API+Filtering+and+Sorting

- An additional querystring parameter “sort” must be used to specify the field to sort by as well as the direction in which to sort
- The example below will sort campaigns by ID in descending order

```
http://api.appnexus.com/
campaign?
advertiser_id=1&sort=id.desc
```

appnexus

Read-Only Services

Read-Only Services

Read only services return reference data that remains relatively static.

The reference data is used to ensure a uniform set of data when assigning certain attributes to objects.

Wiki details:
https://wiki.appnexus.com/display/api/Read-Only+Services

- All read only services should be accessed using the GET method.
- Examples of reference data are
  - Brand
  - Browser
  - Operating System
  - Language
  - City
  - Etc.
- All read only services are fully documented on the Wiki

appnexus

Tools

- API Viewer
  (formerly the ”API Console")
- Meta

appnexus

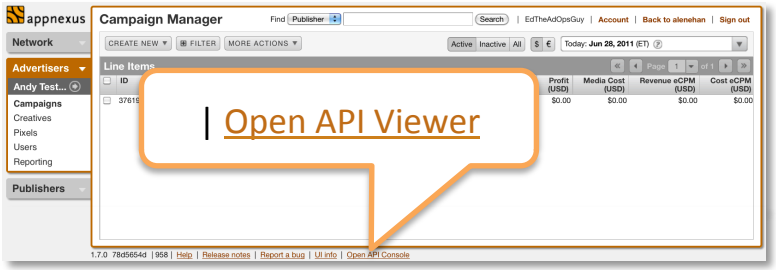
API Viewer
(formerly the ”API Console")

The API ConsVIewerallows users to explore how our UI interacts with our API to change the configuration of objects in our databases.

It can be a tremendously useful tool for both learning the AppNexus Console API (for beginners), and constructing interfaces on top of our API (for developers).

Wiki details:
https://wiki.appnexus.com/display/api/API+Console

- The Console is available via the rightmost link in the bottom menu of any AppNexus Console page.



- Once enabled, it allows you to inspect any of the API calls made during the current session

appnexus

Using the API Viewer

The two main modes of browsing the API Viewer are Page Load and AJAX

- Page Load will display the calls that were made in building the current UI page
- AJAX requests are added to the dropdown menu labeled AJAX as they are successfully executed in the UI (in chronological order).

Wiki details:
https://wiki.appnexus.com/display/api/API+Console

Browsing API Calls – the columns displayed in the API call table are defined as follows:

- Index – the order in which the API calls were made. Full details on the call can be seen by clicking on this number.
- Method – the HTTP method that was used for the API call (GET, POST, PUT, DELETE).
- URL – the URL of the API service that was called.*
- Data – the raw data sent to the API (in JSON). Click "view json…" to expand the full request
- Response – the approximate size and contents of the API response
- API / TX Time – the time the API took to process the request, as well as the transfer time, which is defined as round trip request time minus API processing time

*Note that GET and DELETE querystrings will be displayed in the Data column

appnexus

Meta

The meta function provides information about the service, rather than the objects in that service.

Wiki details:
https://wiki.appnexus.com/display/api/API+Semantics

- Information available through meta includes:
  - "name": The name of the field being described.
  - "type": The data type of the field. This could be string, number, boolean, array or object.
  - "sort_by": Whether or not the field is eligible for use in the "sort" querystring parameter.
  - "filter_by": Whether or not the field is eligible for use as a filter on the querystring.
- For each service, additional details for each field, including a description, is available on the Wiki.

Best Practices

Best Practices

- Use a configuration-driven integration.
- Throttle API calls to 100 reads per minute and 60 writes per minute. This is enforced programmatically by the API.
- Build a wrapper to the API to centralize code, perform logging, error handling, etc.
- Read the entire API Wiki before using the API.
- Do not assume that API requests are successful. Always check the “status” field on all responses. If this field is not present, that indicates that there was an error while processing your requests
- Use pagination to retrieve large quantities of objects with multiple API requests. The API limits responses to 100 objects, so multiple requests must be made for more than 100 objects.
- Use codes to easily sync with your internal system. Rather than store AppNexus IDs, objects can be identified with member-specific integration codes.
- Stagger reports so they are not all scheduled at the same time.
- Discuss with your Account Manager and Implementation Consultant what constitutes a breaking change.

Wiki details:
https://wiki.appnexus.com/display/api/Best+Practices

appnexus

Breaking vs. Non-Breaking Changes

Breaking vs. Non-Breaking Changes

Breaking Changes

- Removal of a field
- Change of a field from non-mandatory to mandatory
- Addition of new validations
- Modification of data structure (e.g., an array of IDs is converted to an array of objects)
- Modification to the throttling of requests
- Reduction in the number of objects returned by responses

Non-Breaking Changes

- Addition of a new non-mandatory field
- Addition of a new service
- Addition of metrics or dimensions to reports
- Addition of a new report type
- Deprecation of a field without removing it from the service
- Reduction or increase in the number of concurrent reports allowed

Wiki details:
https://wiki.appnexus.com/display/api/Breaking+Changes

appnexus

Breaking vs. Non-Breaking Changes

AppNexus will provide advance notice of breaking changes, but non-breaking changes roll out on a regular basis.

Wiki details:
https://wiki.appnexus.com/display/api/Breaking+Changes

appnexus

In-Depth Examples

- Campaign Hierarchy
- Reporting

Example 1: Creating the Campaign Hierarchy

1) Create the advertiser via POST

Note the use of the "code" parameter to specify an external identifier.

Wiki details:
https://wiki.appnexus.com/display/api/Advertiser+Service

```
http://api.appnexus.com/advertiser
{
  "advertiser": {
    "code": "adv_sample_123",
    "name": "Sample Advertiser",
    "timezone": "EST",
    "default_currency": "USD"
  }
}
```

appnexus

Example 1: Creating the Campaign Hierarchy

2) Create the line item via POST

Note the use of the "code" parameter to specify an external identifier.

Wiki details:
https://wiki.appnexus.com/display/api/Line+Item+Service

```
http://api.appnexus.com/line-item?
advertiser_code=adv_sample_123
{
  "line-item": {
    "advertiser_code": "adv_sample_123",
    "currency": "USD",
    "code": "line_sample123_1",
    "name": "Sample Line Item",
    "start_date": null,
    "end_date": null,
    "lifetime_budget": null,
    "lifetime_budget_imps": 1000000,
    "daily_budget": null,
    "daily_budget_imps": 100000
  }
}
```

appnexus

Example 1: Creating the Campaign Hierarchy

3) Create a creative via POST

The format of the creative must match the creative content or the content delivered by the media URL. This impacts the way our ad server writes the content to a web page.

Wiki details:
https://wiki.appnexus.com/display/api/Creative+Service

```
http://api.appnexus.com/creative?
advertiser_code=adv_sample_123
{
  "creative": {
    "code": "creative_sample_1",
    "media_url": "http://
sample_ad_code.com/12345",
    "format": "url-js",
    "width": 728,
    "height": 90
  }
}
```

appnexus

Example 1: Creating the Campaign Hierarchy

4) Create a segment via POST (if needed)

The segment would be used for targeting through the profile. This particular segment will be associated with an advertiser as a result of including the advertiser code (or ID) in the querystring and JSON.

Wiki details:
https://wiki.appnexus.com/display/api/Segment+Service

```
http://api.appnexus.com/segment?
advertiser_code=adv_sample_123
{
  "segment": {
    "advertiser_code": "adv_sample_123",
    "code": "seg_adv_sample_123_1",
    "short_name": "Sample Segment 1",
    "state": "active",
    "expire_minutes": "43200",
    "piggyback_url": null,
    "piggyback_pixel_type": null
  }
}
```

appnexus

Example 1: Creating the Campaign Hierarchy

5) Create the campaign via POST

The campaign is created as "inactive" and does not currently have a profile to control targeting.

Creatives are associated with the campaign at the time of creation.

Wiki details:
https://wiki.appnexus.com/display/api/Campaign+Service

```
http://api.appnexus.com/campaign?
advertiser_code=adv_sample_123
{
  "campaign": {
    "state": "inactive",
    "line_item_code": "line_sample123_1",
    "code": "camp_sample_123_1",
    "name": "Sample Campaign",
    "start_date": null,
    "end_date": null,
    "lifetime_budget_imps": 100000,
    "daily_budget": null,
    "daily_budget_imps": null,
    "enable_pacing": null,
    "learn_budget": null,
    "creatives": [
      {"code": "creative_sample_1"}
    ],
    ...
  }
}
```

appnexus

Example 1: Creating the Campaign Hierarchy

6) Create the targeting profile via POST

This profile targets the US, a domain list, a segment and intended audiences.

7) Add the profile to the campaign and set the campaign active via PUT

Wiki details:
https://wiki.appnexus.com/display/api/Profile+Service

```
http://api.appnexus.com/profile?
advertiser_code=adv_sample_123
{
  "profile": {
    "code": "profile_123_1",
    "country_action": "include",
    "country_targets": [
      {"country": "US"}
    ],
    "domain_list_action": "exclude",
    "domain_list_targets": [{"id": "4"}],
    "segment_targets": [
      {
        "code": "seg_adv_sample_123_1",
        "action": "include"}
    ],
    "intended_audience_targets": [
      "general",
      "young_adult",
      "children"
    ]
  }
}
```

appnexus

Example 2: Requesting Reports

1) Create the report request via POST

The reporting service will respond with a report ID that should be used to retrieve the report data.

Wiki details:
https://wiki.appnexus.com/display/api/Reporting+Service

```
http://api.appnexus.com/report?
publisher_id=1234
{
  "report": {
    "report_type":
"network_publisher_analytics", "timezone":
"EST5EDT",
    "report_interval": "yesterday",
    "columns": ["pub_rule", "imps_total",
"imps_blank", "imps_psa", "imps_psa_error",
"imps_default_error", "imps_default_bidder",
"imps_kept", "imps_resold", "imps_rtb"],
    "filters": [{"publisher_id": 1234}],
"groups": ["pub_rule_id"],
    "orders": ["pub_rule", "imps_total",
"imps_blank", "imps_psa", "imps_psa_error",
"imps_default_error", "imps_default_bidder",
"imps_kept", "imps_resold", "imps_rtb"],
    "name": "Publisher Report"
  }
}
```

appnexus

Example 2: Requesting Reports

2) Retrieve the report results via GET using the report ID

If the "execution_status" is "ready", the report data will be included in the "data" parameter.

If the "execution_status" is "pending", the report is still processing.

Wiki details:
https://wiki.appnexus.com/display/api/Reporting+Service

```
http://api.appnexus.com/report?
id=b13145af32c6b4e836d6700cb2a4d344
{
  "response": {
    "status": "OK",
    "report": {
      "name": "Publisher Report", "created_on":
"2011-04-22 15:50:19",
      "cache_hit": false, "fact_cache_hit":
false, "fact_cache_error": "did not find any
cache table for 6",
      "json_request": ..., "data":
"pub_rule,imps_total,imps_blank,imps_psa,imps_psa
_error,imps_default_error,imps_default_bidder,imp
s_kept,imps_resold,imps_rtb\r\nPub Rule A
(12345),169021,0,0,127628,41393,0,0,0,0\r\nPub
Rule B (12346),
14696851,0,107377,0,0,194943,14394531,0,0\r\nPub
Rule C (12347),764620,0,574,0,0,1871,762175,0,0\r
\nPub Rule D (12348),
1321936,0,392,0,0,70,1321474,0,0\r\nPub Rule E
(12349),552078,0,3364,0,0,0,548714,0,0\r\nPub
Rule F (12350),82973,0,0,0,0,0,82973,0,0\r\nPub
Rule G (12351),40540,0,134,0,0,28363,12043,0,0\r
\n"
    },
    "execution_status": "ready", ...
  }
}
```

appnexus

Thanks for attending!