Reporting API

- Make a Report Request
    - SDK
        - Getting the next pages of your report
    - REST API
        - Synchronous Example
        - Asynchronous Example
        - Pagination
- Available parameters

Make a Report Request

See the available parameters below for running a report with SDK or the API.

SDK

```
In [11]: report = springserve.reports.run(start_date="2016-09-19", end_date="2016-09-19",
dimensions=["supply_tag_id"], declared_domains=["nytimes.com", "weather.com"])
In [12]: report.ok
Out [12]: True


In [13]: data = report.to_dataframe()
```

The variable data is now a Pandas dataframe.

Getting the next pages of your report

Use the get_next_page method to get the next page of your report. The method returns True if it got the next page and False if not, indicating that you are already at the last page.

Note that calling get_next_page overwrites the current data, so when you call to_dataframe the results will only contain the data from the last page that was downloaded.

```
In[12]: report.get_next_page()
Out[12]: True
```

Use the get_all_pages method to get the remaining pages (if you're currently at page 2 it will get page 2 onwards) of your report, this will save all the data from the pages it downloads on the report object. Note that if it is a very large report it is best to get one page at a time so you don't run out of memory.

```
In[12]: report.get_all_pages()
Out[12]:
```

REST API

Report requests can be run as either synchronous or asynchronous

- Synchronous report requests will wait until the report has finished in SpringServe's system, and the response will contain the report data
- Asynchronous report requests will return immediately, and contain a status and report_id. You can then poll the API at an interval to determine if your report is ready and fetch the data.
    - We recommend a polling interval of 5 seconds

POST /api/v0/report

Headers

```
Content-Type application/json

Authorization "yourAuthToken"
```

Synchronous Example

Body (example)

```
{
        "start_date": "2015-11-01",
        "end_date": "2015-11-15",
        "interval": "day",
        "dimensions": ["supply_type"]
}
```

Required parameters: none

Response

Status code 200

```
[
  {
    "date": "2015-11-19 00:00:00",
    "supply_type": null,
    "total_requests": 392841,
    "usable_requests": 299876,
    "blocked_requests": 92965,
    "total_impressions": 40968,
    "flash_impressions": 37497,
    "vast_impressions": 3471,
    "fill_rate": 13.66,
    "flash_errors": 54093,
    "vast_errors": 7,
    "cost": 133.78,
    "revenue": 193.55,
    "profit": 59.77,
    "cpm": 3.27,
    "rpm": 4.72,
    "ppm": 1.46,
    "error_rate": 18.04,
    "usable_request_rate": 76.34
  },
  {
    "date": "2015-11-19 00:00:00",
    "supply_type": "Syndicated",
    "total_requests": 520333,
    "usable_requests": 324776,
    "blocked_requests": 195557,
    "total_impressions": 4795,
    "flash_impressions": 3633,
    "vast_impressions": 1162,
    "fill_rate": 1.48,
    "flash_errors": 18010,
    "vast_errors": 10,
    "cost": 32.32,
    "revenue": 35.5,
    "profit": 3.18,
    "cpm": 6.74,
    "rpm": 7.4,
    "ppm": 0.66,
    "error_rate": 5.55,
    "usable_request_rate": 62.42
  },
  ...
  ]
}
```

Asynchronous Example

Initial request body (example)

```
{
        "start_date": "2017-11-28",
        "end_date": "2015-11-15",
        "interval": "day",
        "dimensions": ["supply_type"],
        "async": "true"
}
```

Required parameters: none

Response (Pending report)

Status code 200

```
{
        "status":"BUILDING",
        "report_id":"15118956000004308405d5cf3f869a6208b768f0295eb",
        "page":1,
        "total_pages":0,
        "data":[]
}
```

Polling request body (example)
***NOTE that you must add the report_id to the initial payload.*

```
{
        "report_id": "15118956000004308405d5cf3f869a6208b768f0295eb",
        "start_date": "2017-11-28",
        "end_date": "2015-11-15",
        "interval": "day",
        "dimensions": ["supply_type"],
        "async": "true"
}
```

Required parameters: initial payload AND report_id

Response (Complete report)

Status code 200

```
{
        "status":"COMPLETE",
        "report_id":"15118984800009457e3a723f49e43c0a40da734e8357a",
        "page":1,
        "total_pages":0,
        "data": [
                {
                        "date":"2017-11-28 00:00:00.0",
                        "total_requests":885772,
                        "billable_requests":712521,
                        "usable_requests":857899,
                        "blocked_total":27873,
                        "blocked_requests":13902,
                        "whiteops_blocked":0,
                        "prebid_blocked_ias":0,
                        "blocked_pre_bid_ivt":0,
                        "opportunities":714907,
                        "missed_opportunities":0,
                        "total_impressions":159349,
                        "flash_impressions":79626,
                        "vast_impressions":79723,
                        "usable_request_rate":0.96853,
                        "blocked_rate_total":0.0314674656683661,
                        "blocked_request_rate":0.0156947837592518,
                        "whiteops_blocked_rate":0.0,
                        "prebid_blocked_ias_rate":0.0,
                        "blocked_rate_pre_bid_ivt":0.0,
                        "opportunity_rate":0.83332,
                        "fill_rate":0.18574,
                        "opportunity_fill_rate":0.22289,
                        "revenue":522.60105,
                        "cost":689.87713,
                        "third_party_fees":0.23219
                },
                ...
        ]
}
```

Pagination

Each page of a report is limited to 20,000 rows. To receive the 5th page of your report, you would set the body as such:

Body (example)

```
{
        "start_date": "2015-11-01",
        "end_date": "2015-11-15",
        "interval": "day",
        "dimensions": ["supply_type"],
        "page": 5
}
```

To receive all pages, you can iterate through the page numbers until you receive an empty page.

Available parameters

parameter	options (if applicable)	notes
start_date	"2015-12-01 00:00:00" or "2015-12-01"	minutes and seconds are ignored ie must be "yyyy-mm-dd hh:00:00"
end_date	"2015-12-02 00:00:00" or "2015-12-01"	minutes and seconds are ignored ie must be "yyyy-mm-dd hh:00:00"
interval	"hour", "day", "cumulative"	
timezone	"UTC", "America/New_York"	defaults to America/New_York
date_range	Today, Yesterday, Last 7 Days	date_range takes precedence over start_date/end_date
dimensions	supply_tag_id, demand_tag_id, declared_domain, detected_domain, demand_type, supply_type, supply_partner_id, demand_partner_id, supply_tag_label, demand_tag_label, key_values, country, declared_player_size, detected_player_size, demand_code, device_id, marketplace_type, buying_demand_tag_id, selling_supply_tag_id, campaign_code, environment, vpaid_type, app_name, app_bundle	domain is only available when using date_range of Today, Yesterday, or Last 7 Days to add a specific key as a dimension: "key:my_key"
the following parameters act as filters; pass an array of values (usually IDs)		
supply_tag_ids	[22423,22375, 25463]	can filter by one or more supply tag
demand_tag_ids	[22423,22375, 25463]	
declared_domains	["nytimes.com", "weather.com"]	
detected_domains	["nytimes.com", "weather.com"]	
supply_types	["Syndicated","Third-Party"]	
supply_partner_ids	[30,42,41]	
supply_tag_label_ids	[13,15,81]	
demand_partner_ids	[3,10,81]	
demand_tag_label_ids	[4, 8, 10]	
demand_types	["Vast Only","FLASH"]	

demand_codes	[10023, 13341, 12343]	
account_id	103	Relevant if you have access to multiple account ids. Cannot be a list of ids.
limit	10	Limit the number of records returned
page	10	The page requested for results over 20K rows. If there are 20,000 entries in your result, you need to call multiple pages, until you get an empty response.
keys	["my_key1", "my_key2"]	
key_values	{"my_key":"my_value"}	
environments	["desktop", "mobile"]	
buying_demand_tag_ids	[134523, 198523, 123413]	
selling_supply_tag_ids	[93471, 51235, 102345]	
vpaid_types	["both", "flash"]	
campaign_codes	[51, 42]	
country_codes	["US", "AU"]	
declared_player_sizes	["small", "medium"]	'small', 'medium', 'large', 'x-large'
detected_player_sizes	["small", "medium"]	
device_ids	[1,2]	0=Other, 1=Computer, 2=Mobile, 3=Tablet, 4=Game Console, 5=Digital Media Receiver, 6= Wearable Computer, 7= Connected TV
marketplace_type_ids	[2,3]	1 = managed, 2 = DC Sold, 3 = DC to DC, 4 = DC Bought
app_names	["solitare", "candy crush"]	
app_bundles	["com.myfitnesspal.android", "com.roku.ae"]	