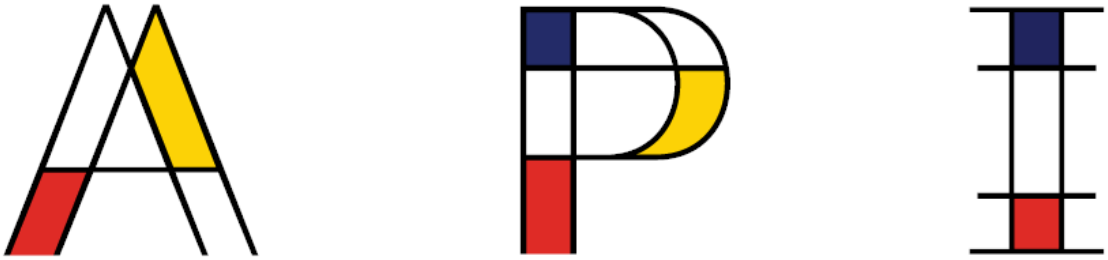
Exploring the Pure1® REST API, Part 2

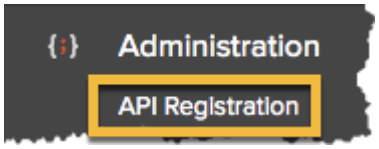


This is Part 2 of the Pure1 REST API blog series. You can read Part 1 here.

Authenticating with the Pure1 REST API

Pure1® relies on the OAuth 2.0 Token Exchange protocol (in draft mode at the time of writing) to grant OAuth2 clients access to its REST API via what is called an "access token" (on a side note, an "OAuth2 client" would be Postman or your own custom library of application).

As is the case for many OAuth 2.0 implementations, you must first sign in as an administrator and register an application in the ***Administration → API Registration*** section of your Pure1 Manage site using RSA keys.



The detailed instructions to set this up are available in the Authentication section of the Pure1 REST API Reference so please refer to them first and foremost. Among completion or your app registration, Pure1 will provide an Application ID which you can use, along with your RSA secret key, to generate valid API access tokens. We provide a Python script in the Authentication section of the Pure1 REST API Reference or

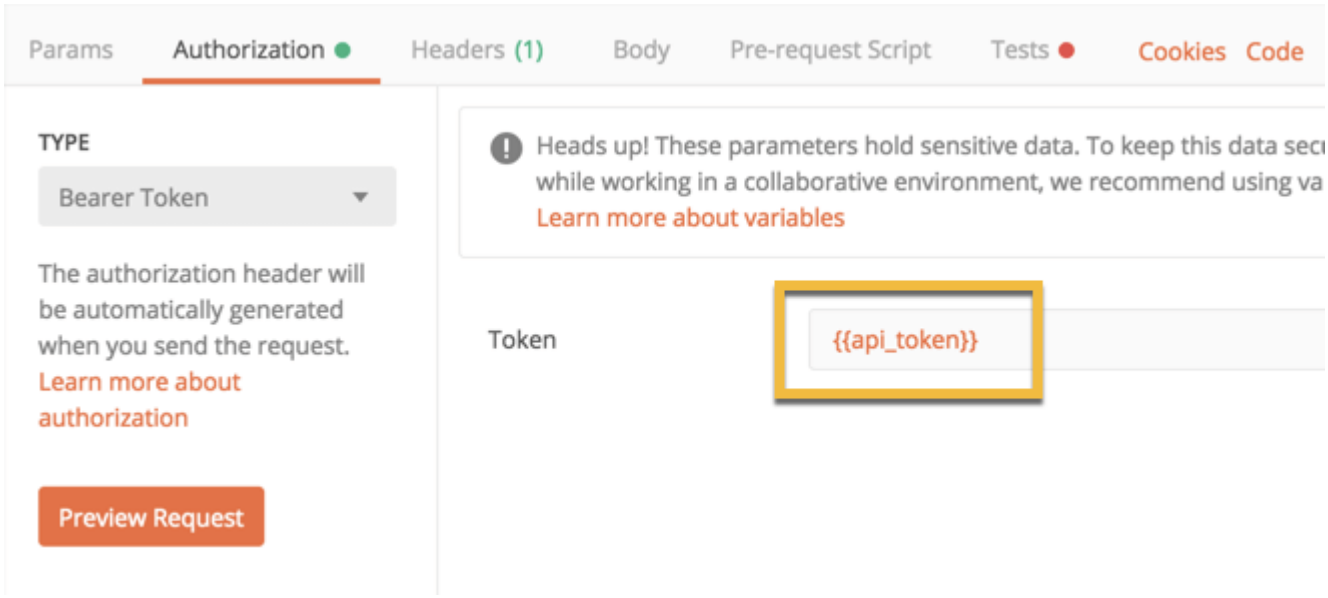
instructions to generate it with https://jwt.io and Swagger. If you are an OAuth2 developer expert, don't hesitate to share your scripts in other programming languages with the Pure/Code Slack community (instructions to join it are available at https://code.purestorage.com in the Getting Started section) and we'll be happy to spread the word out about them.

After you have installed Postman, download the Pure1 Postman collection which includes an environment aptly named "pure1". Open the pure1 environment and paste the access token you just generated into the `api_token` environment variable current value, as shown in the screenshot below:

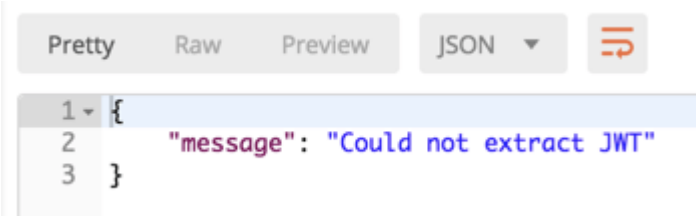
pure1 Edit

VARIABLE	INITIAL VALUE	CURRENT VALUE
api_token	eyJraWQiOiI5TEVpRURBLzlLQmxTZXpneVN0NEo0N2JtQ0xyIiwidHlwIjoiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQiOiJwdWJsaWMtYXBpIiwic3ViIjoiMjI2NCIsInJvbG...	eyJraWQiOiI5TEVpRURBLzlLQmxTZXpneVN0NEo0N2JtQ0xyIiwidHlwIjoiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQiOiJwdWJsaWMtYXBpIiwic3ViIjoiMjI2NCIsInJvbG...
array_name	sn1-x70-e05-37	sn1-x70-e05-37
array_version	5.0.1	5.0.1
array_model	FlashBlade	FlashBlade
continuation_token	MGQ3MGM0NzYtMmM5Zi00YjE0LTkwNTktZjU0ZTAzZDE0MGFh	MGQ3MGM0NzYtMmM5Zi00YjE0LTkwNTktZjU0ZTAzZDE0MGFh
env_tag_value	prod	prod
selected_array_id	0184cc69-85eb-4798-82ee-7f7324c76d18	0184cc69-85eb-4798-82ee-7f7324c76d18
selected_array_name	sn1-m50-b01-33	sn1-m50-b01-33
metric_resolution	1000000000	1000000000
selected_volume_name	EX06-21-DB1	EX06-21-DB1

All the Postman requests in that collection are configured to use that API token as an OAuth2 bearer token so there's nothing else to do (as far as authentication is concerned, at least):



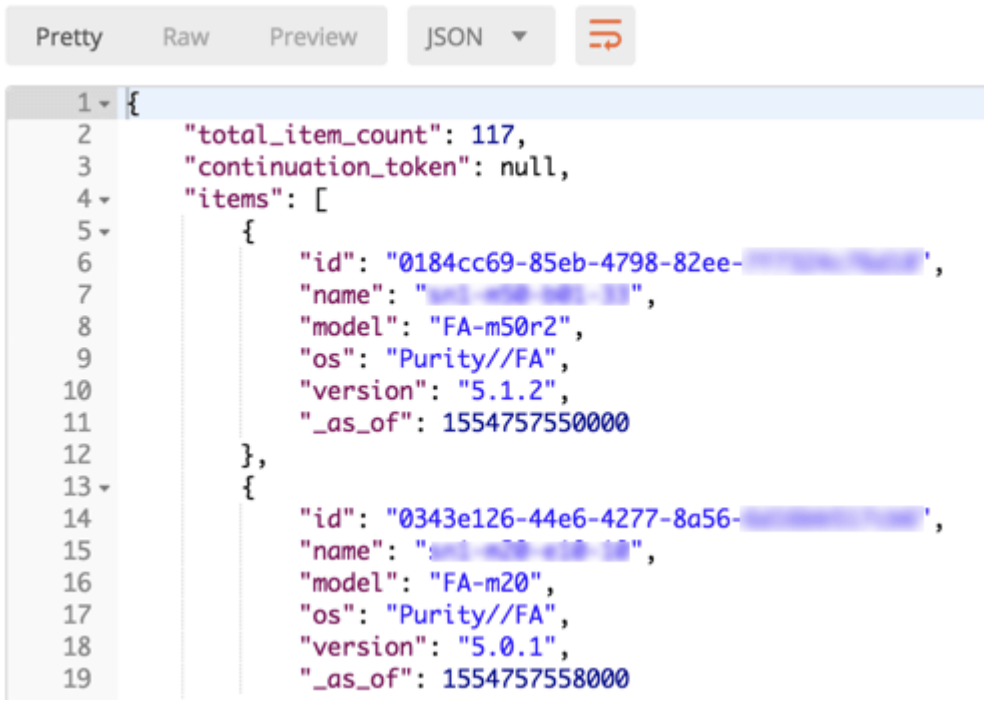
While you're at it, try the Get All Appliances request to test that your access token is properly entered. Postman will display a variety of errors if you have a malformed, invalid or expired token:



[Note that you might hit that last expiration error more frequently than the other ones, since our OAuth2

tokens are currently available for 10 hours]

If your access token is valid though, hurray! You should get a beautiful JSON list of all your Pure1-managed arrays:



The Postman magic

If you randomly play around the Postman requests available in the Pure1 Postman collection, you will most likely wonder why some filtered requests return valid data even though you don't specify the filters.

For instance, if you use the Get All Appliances request followed by the Get Appliance by Name request, you will surely notice that the second request does return a valid response, without you setting the name of the selected appliance. Well, I have both bad and good news. The bad news is that there is unfortunately no black (nor white) magic involved. The good news is that there IS a little dev magician tucked into Postman: the scripting feature available in the "Tests" tab of each Postman request indeed does wonders at extracting data from a Postman response and setting environment variables that can be used in subsequent requests.

Check out the code available in the Tests tab of the Get All Appliances request:
[crayon-6427906e0b981678388140/]
This straightforward script extracts the id and name of the first 2 results of the /arrays API response and sets them to 4 environment variables, that are used as inputs for various other sample requests such as Get Application by Name, Get Appliance by ID and many more.

Digging into the Pure1 API endpoints in Postman

As pointed out in the SmartBear's State of API 2019 report, the most important characteristic developers need in an API is ease of use. On that usability aspect, the Pure1 REST API assuredly checks the box, as all

its endpoints return results when used in their simplest forms. For instance, /arrays returns all the appliances managed by your Pure1 account, while /volumes returns all the volumes.

However, you most likely want greater granularity and smaller response payloads when requesting data through the Pure1 API. To answer these needs, the Pure1 API provides sorting, pagination and a rich filtering functionality across all of its primary endpoints. Don't take my word for it though: the official Pure1 REST API reference will always be your trusted friend and should remain your primary reference in case the data below is no longer relevant. Let's start with sorting and pagination.

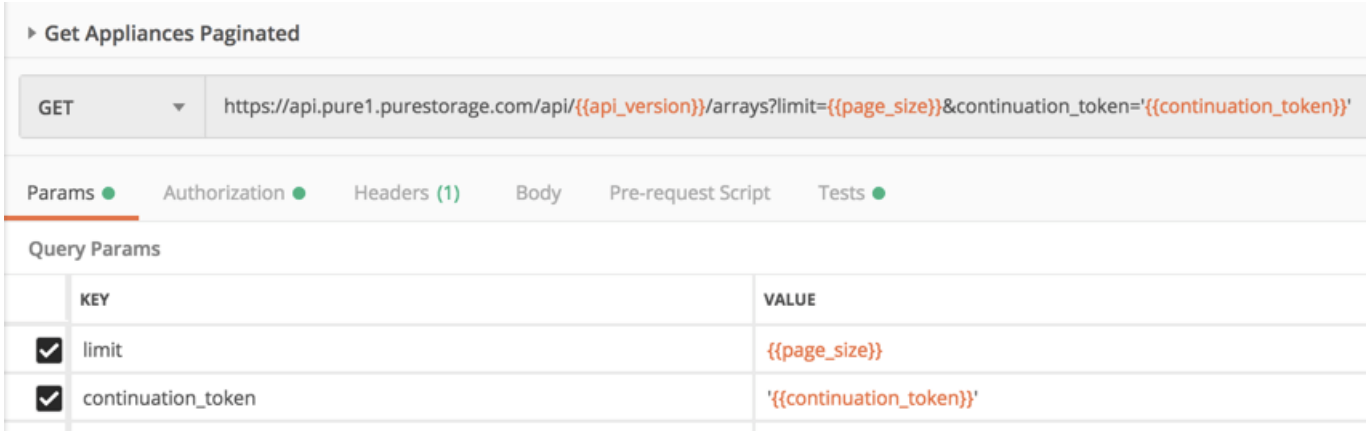
Sorting

Sorting results by property is extremely easy: you simply specify the `sort` parameter and assign it to the name of the property you want to sort by. Check out the Get Appliances Sort by Version ASC ascending order example:
[crayon-6427906e0b98a230143093/]
Sorting by descending order is equally easy and only involves adding a minus sign after the property name. See the Get Appliances Sort by Version DESC for a simple example:
[crayon-6427906e0b98c701753456/]

Pagination

The Pure1 REST API supports 2 types of pagination: seek and offset.

Seek (or token-based) pagination is as simple as adding `limit` and `continuation_token` parameters, as demonstrated in the Get Appliances Paginated request:



`page_size` is by default set to 10 in the pure1 Postman environment, and you can update it to fit your needs.

Postman's results pane displays a JSON response similar to the following screenshot, where you may notice a `continuation_token` attribute (whenever the total item count is greater than the limit):

With the use of Postman's Tests tab, we retrieve and assign this continuation token to a (different) `continuation_token` environment variable, so you can repeatedly press the Send button to easily iterate through your list of Pure1-managed appliances. You may also find similar paginated requests in the Volumes, Alerts and Audits endpoints.

The second pagination option is offset-based and entails specifying the page number (the 'offset') you would like to view at the time of request.

The main difference between the seek and offset methods is data pagination consistency and sorting. With the seek/token method, you are guaranteed to get each item once and only once if you iterate through all the pages. However, you can't sort by any property and seek pagination requests are always ordered by `id` ascending.

On the other hand, with the 'offset' method, requests are independent from each other, so you could potentially get items once, twice (or more) or never as you iterate through the pages (if matching items were added or deleted between requests). The upside of offset pagination is that you can mix it with sorting.

So when should you use seek vs. offset pagination? We recommend you use the seek pagination method if you absolutely and reliably need to get all matching items once and only once. On the other hand, we find that the offset pagination is appropriate for table-like GUIs, where the convenience of sorting and pagination bears more weight.

To conclude Part 2, check out the Get Volumes Paginated w/Offset sample request for an offset pagination example:
[crayon-6427906e0b98d279981845/]
For more details, please refer to the Pagination and Sorting sections of the official documentation.

As you may have noticed above, the request above combines both pagination, sorting… and filtering. That aptly brings us to Part 3 of our blog series, where we'll deep dive into the filtering and metrics capabilities of the Pure1 REST API. See you there soon!