

Blue Planet

Introduction to REST APIs and Blue Planet Orchestrator APIs

Lab Guide



LEGAL NOTICES

THIS DOCUMENT CONTAINS CONFIDENTIAL AND TRADE SECRET INFORMATION OF CIENA CORPORATION AND ITS RECEIPT OR POSSESSION DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE. REPRODUCTION, DISCLOSURE, OR USE IN WHOLE OR IN PART WITHOUT THE SPECIFIC WRITTEN AUTHORIZATION OF CIENA CORPORATION IS STRICTLY FORBIDDEN.

EVERY EFFORT HAS BEEN MADE TO ENSURE THAT THE INFORMATION IN THIS DOCUMENT IS COMPLETE AND ACCURATE AT THE TIME OF PUBLISHING; HOWEVER, THE INFORMATION CONTAINED IN THIS DOCUMENT IS SUBJECT TO CHANGE.

While the information in this document is believed to be accurate and reliable, except as otherwise expressly agreed to in writing CIENA PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OR CONDITION OF ANY KIND, EITHER EXPRESS OR IMPLIED. The information and/or products described in this document are subject to change without notice. For the most up-to-date technical publications, visit www.ciena.com.

Copyright © 2015-2018 Ciena® Corporation – All Rights Reserved

The material contained in this document is also protected by copyright laws of the United States of America and other countries. It may not be reproduced or distributed in any form by any means, altered in any fashion, or stored in a database or retrieval system, without the express written permission of Ciena Corporation.

Ciena®, the Ciena logo, Z22™, Z33®, Z77®, Blue Planet® and other trademarks and service marks of Ciena appearing in this publication are the property of Ciena. Trade names, trademarks, and service marks of other companies appearing in this publication are the property of the respective holders.

Security

Ciena cannot be responsible for unauthorized use of equipment and will not make allowance or credit for unauthorized use or access.

Contacting Ciena

Corporate headquarters	410-694-5700 or 800-921-1144	www.ciena.com
Customer technical support/warranty		
In North America	1-800-CIENA24 (243-6224) 410-865-4961	
In Europe, Middle East, and Africa	800-CIENA-24-7 (800-2436-2247) +44-207-012-5508	
In Asia-Pacific	800-CIENA-24-7 (800-2436-2247) +81-3-6367-3989 +91-124-4340-600	
In Caribbean and Latin America	800-CIENA-24-7 (800-2436-2247) 410-865-4944 (USA)	
Sales and General Information	410-694-5700	E-mail: sales@ciena.com
In North America	410-694-5700 or 800-207-3714	E-mail: sales@ciena.com
In Europe	+44-207-012-5500 (UK)	E-mail: sales@ciena.com
In Asia +81-3-3248-4680 (Japan)		E-mail: sales@ciena.com
In India	+91-124-434-0500	E-mail: sales@ciena.com
In Latin America	011-5255-1719-0220 (Mexico City)	E-mail: sales@ciena.com
Training	877-CIENA-TD (243-6283) or 410-865-8996	E-mail: techtnng@ciena.com

For additional office locations and phone numbers, please visit the Ciena website at www.ciena.com.

Change History

Release	Revision	Publication Date	Reason for Change
1	0.0	12/19/2017	Initial Release
1	b	04/11/2018	Updated content to align with updated 849 course

Contents

Blue Planet REST APIs Lab Guide	5
Course Software	5
Task 1: (Optional) Install cURL	7
Task 2: Token Authentication	8
Task 3: Display information about the Number Pool product using cURL.....	13
Task 4: Use Swagger to create an instance of a resource	14
Task 5: Use cURL and the BPO UI to verify the resource was created.....	17
Task 6: Delete the resource you created with cURL	19

Blue Planet REST APIs Lab Guide

Course Software

- Blue Planet Orchestrator
- Swagger
- cURL
- Google Chrome or Mozilla Firefox
- Text editor of your choice

Course Setup Information

Blue Planet Orchestrator and Swagger

This course also makes use of the Blue Planet Orchestrator, Swagger and the command line utility cURL. You will need a production installation of Blue Planet Orchestrator available to you, or an installation of the DevOps toolkit to complete this course.

Blue Planet Orchestrator Address Replacement

You will need to note the address of the installation of Blue Planet Orchestrator you are using to complete the labs successfully. This course uses the example address of <https://localhost/> to represent the instance of Blue Planet Orchestrator you are working with. You will need to replace localhost with the address of your instance of Blue Planet Orchestrator

The Blue Planet Orchestrator is located at:

[https://\[YourProductionAddressOrDevOpsToolkitAddress\]/login/?next=%2F](https://[YourProductionAddressOrDevOpsToolkitAddress]/login/?next=%2F) .

Swagger is accessible through the Hanover Blue Planet server at:

[https:// \[YourProductionAddressOrDevOpsToolkitAddress\]/bpocore/swagger#](https://[YourProductionAddressOrDevOpsToolkitAddress]/bpocore/swagger#) .

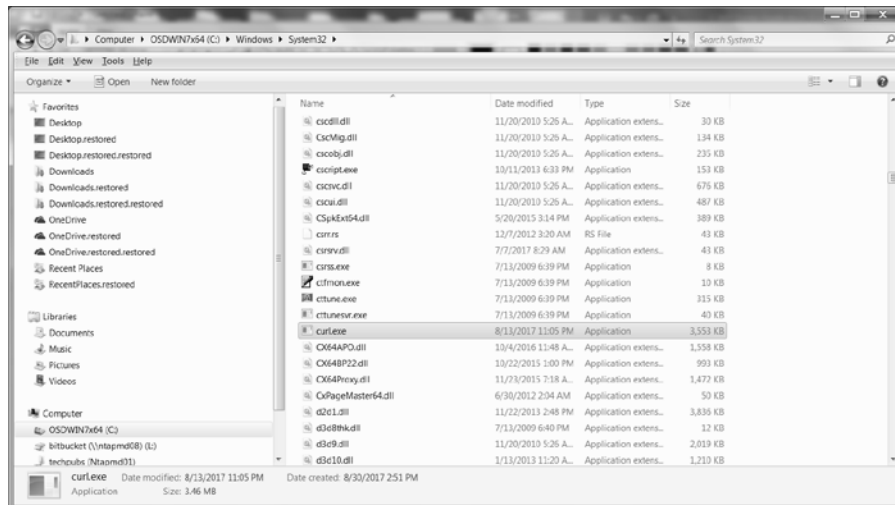
Credentials

You must note the credentials that are used to access your installation of Blue Planet Orchestrator. The example credentials used to access Blue Planet Orchestrator in this class are (U: 855student, P: 855student). Any exercise that refers to these credentials will need to be replaced with your own credentials.

cURL

If cURL is not installed on your computer, you may download it from <https://curl.haxx.se/download.html>. Windows users, it is suggested you download the Win64 x86_64 7zip 7.55.1 package. The package name will be curl-7.55.1-win64-mingw.7z. Unpack the 7zip file and copy curl.exe to a location on your Windows machine that is mapped to your operating system search path, such as c:\windows\system32.

Win64 - Generic					
Win64 ia64 CAB	7.55.1	binary	SSL	Stefan Kanthak	
Win64 x86_64 7zip	7.53.1	binary	SSL	Darren Owen	
Win64 x86_64 zip	7.53.1	binary	SSL SSH	Marc Hörsken	849 KB
Win64 x86_64 CAB	7.55.1	binary	SSL	Stefan Kanthak	
Win64 x86_64 7zip	7.55.1	binary	SSL SSH	Viktor Szakáts	1.86 MB
Win64 2000/XP x86_64 MSI	7.46.0	binary	SSL SSH	Edward LoPinto	
Win64 2000/XP x86_64 zip	7.46.0	binary	SSL SSH	Edward LoPinto	



Task 1: (Optional) Install cURL

cURL is a project and its primary purpose and focus is to make two products:

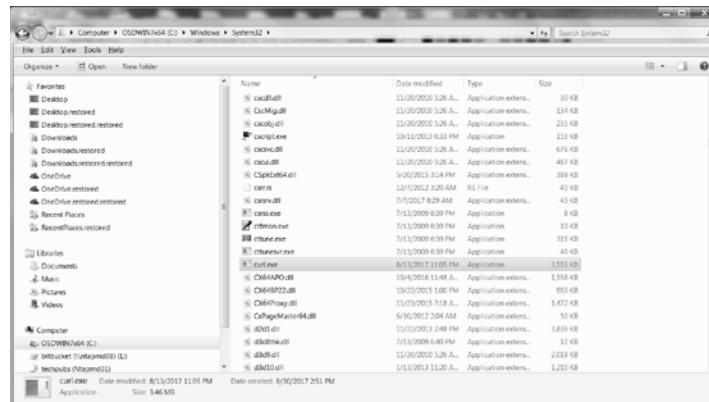
- curl, the command-line tool
- libcurl the transfer library with a C API

Both the tool and the library do Internet transfers for resources specified as URLs using Internet protocols. Documentation for cURL may be found at <https://ec.haxx.se/curl-does.html>.

1. Navigate to <https://curl.haxx.se/download.html>.
2. Select the appropriate installation package for your platform and download the package. (Windows users, it is suggested that you download the Win64 x86_64 7zip 7.55.1 package. (*Nix users may already have cURL installed. If so, simply use your installed command. Optionally, *Nix users may use a package manager to install cURL)

Win64 - Generic					
Win64 ia64 CAB	7.55.1	binary	SSL	Stefan Kanthak	
Win64 x86_64 7zip	7.53.1	binary	SSL	Darren Owen	
Win64 x86_64 zip	7.53.1	binary	SSL SSH	Marc Hörsken	849 KB
Win64 x86_64 CAB	7.55.1	binary	SSL	Stefan Kanthak	
Win64 x86_64 7zip	7.55.1	binary	SSL SSH	Viktor Szakáts	1.86 MB
Win64 2000/XP x86_64 MSI	7.46.0	binary	SSL SSH	Edward LoPinto	
Win64 2000/XP x86_64 zip	7.46.0	binary	SSL SSH	Edward LoPinto	

3. Once downloaded, unpack the package to a location on your filesystem. Make sure the curl.exe command is unpacked to a location that is mapped to your Operating System search path. (For instance, Windows users should ensure that curl.exe is unpacked to a location such as c:\Windows\system32)



Task 2: Token Authentication

When interacting with Blue Planet Orchestrator through swagger, authentication credentials must be passed as part of the query.

If a user is logged into swagger, then credentials are created and passed as part of the REST requests. However, users may manually generate authentication tokens and pass them manually as part of a request.

This is immensely beneficial if the requests to Blue Planet Orchestrator are generated programmatically such as a cURL or wget call from the shell or as an API call from inside of a Python or Java application, or if the user is manually generating packets for testing purposes using a packet generation/capture tool such as Wireshark.

Step 1: Exchange Login Credentials for a Token

Token authentication exchanges the user login credentials (user name, password, and tenant name) for a short-lived token (24-hour default). The token is included in the HTTP Authorization header and grants the

caller access appropriate for the corresponding user roles. The header value is in the format: token <token_value>. To get the token, post the user credentials directly to the UAC application (/tron/api/v1/tokens).

Tron is the original name for the Blue Planet User Access Control and is used frequently in Blue Planet API's and Directory structure. Tron and UAC (User Access Control) are synonymous.

Note, that if you log into the Swagger UI with your credentials, and authentication token is generated automatically and passed with requests you make within the UI. These exercises are designed to teach you how to make calls to the API's independent of the Swagger UI.

Both the swagger user interface and cURL may be used to exchange login credentials for an authentication token. We will look at both methods here.

Exchanging Credentials For A Token With The Swagger UI

1. Open Google Chrome, or Mozilla Firefox and navigate to the Swagger UI at <https://localhost/bpocore/swagger#> (please replace server ip or name if necessary)
2. In the left-hand Components navigation pane, click on the Authentication Component. The right-hand side of the screen should update with the two Authentication API's, tokens and verification.
3. Click on the tokens API. This is where you may exchange credentials for a token or delete a token. The POST form will be used to exchange credentials for a token. The DELETE form may be used to delete or destroy a token.



4. Click on the POST link. This is where you will exchange credential data for your token. User authentication information needs to be added to the userAuthInfo form field. This JSON data containing your credentials are passed to the UAC for processing. The resulting dataset will contain an authentication token that may be added to any successive request. A JSON template has been created for you and is in the supplemental_files directory that was supplied as part of your course kit. This JSON template will be used to supply your credential data in exchange for a token.
5. Navigate to your supplemental_files directory and open it. Locate the template.JSON file associated to your login credentials and open it. Copy the contents of the file to your clipboard. The content should be like the following code snippet.

```
{
  "name": "855student",
  "password": "855student",
  "domain": {
    "name": "master"
  }
}
```

- Move back to the Swagger UI, and paste the contents of the file into the userAuthInfo form field. Scroll down and click the “Try it out” button to pass your credentials.

Components

- Application
- Asset manager
- Authentication
- Component registry
- Diagnostics
- Events
- Import/Export
- Market
- Policy manager
- Resource providers
- Rest server

Authentication
APIs to access the authentication service

tokens: Authentication tokens (i.e., proxy interface to UAC) Show/Hide List Operations Expand Operations

DELETE /tokens Delete or invalidate a token in the authentication system (i.e., logout)

POST /tokens Create a token in the authentication system (i.e., login)

Implementation Notes
The granted token is provided in the X-Subject-Token header of the response.

Response Class (Status 200)
Model **Model Schema**

Response Content Type **application/json**

Parameter	Value	Description	Parameter Type	Data Type
userAuthInfo	<pre>{ "name": "849student", "password": "849student", "domain": { "name": "master" } }</pre>	User's authentication information. Specify either tenant id or name in request. Parent id is ignored.	body	Model Model Schema <pre>{ "id": "string", "name": "string", "password": "string", "tenant": { "id": "string", "name": "string", "parentId": "string", "isMaster": false } }</pre>

Parameter content type: **application/json**

Click to set as parameter value

- If successful, you should receive a Status Code 201 (found in the Response Code section). This indicates a successful request. You will also receive a bevy of salient data, including:
A fully configured cURL command in the Curl section, that you may use to make a similar request from the command line using cURL if you have cURL installed.
The path to the requested API in the Request URL section
The Response JSON data in the Response Body section
The Response Header data in the Response Headers section. This is where your authentication token is stored.
- Locate the Response Header section and locate the “x-subject-token” key. The value of this key is your authentication token and should look like a hex decimal number. Your return headers will be like the following snippet. The token is bold.

```
{
  "c-trace": "784dd6dd-1b81-4ba0-acf2-14bd1fffa546",
  "date": "Tue, 26 Sep 2017 20:11:56 GMT",
  "server": "spray-can/1.3.3",
  "content-type": "application/json; charset=UTF-8",
  "content-length": "161",
  "x-subject-token": "99b6b922dd24c73de939"
}
```

- Copy the value of the “x-subject-token” key into a text file. This is your 24-hour authentication token and you will be using it in successive labs.

Exchanging Credentials For A Token With cURL

Developers may also use command line tools such as cURL to interact with Blue Planet Orchestrator. This is extremely useful when developers should test commands without using Swagger, or are going to be embedding REST calls in external applications such as an RA, Python Applications, or Java Applications. In this lab developers will be using cURL to exchange credentials for an authentication token.

Credentials will be passed to the authentication API using cURL, and the return data and headers will contain the authentication token. This process is not dissimilar from the previous lab using Swagger. In fact, part of the result of the previous lab was a cURL command that may be used to retrieve your authentication token from the command line using cURL. There are some notable differences between the cURL command that is supplied by Swagger, and the one that we will be using.

First, the cURL `-i` command line option will be used with your requests. This flag indicated that the response headers should be included in the output. This is important because the response headers contain the `"x-subject-token"` response header which contains your authentication key.

The cURL `-k` command line option will also be used with your requests. This cURL option indicates that an insecure connection will be used to interact with the API. In production, we would have registered a certificate with cURL to make secure requests, but that is not necessary working in our control environment.

1. Open a command prompt or shell.
2. Navigate to any working directory you are using (`/home`, `\documents`) to store data. We will make a working folder to store the results of cURL commands for later use.
3. Create a directory called 855 (`mkdir 855`)
4. Open the 855 directory. Leave your command prompt / shell open.

You are now going to use cURL to exchange credentials for a permission token. The first cURL call will use the `-x` proxy flag which will allow you to specify the request method, the `-i` insecure flag, and the `-k` include header output flag. There is a sample command stored in the `supplemental_files/codesnippets.txt` file. [username is the credentials you are using in this course]

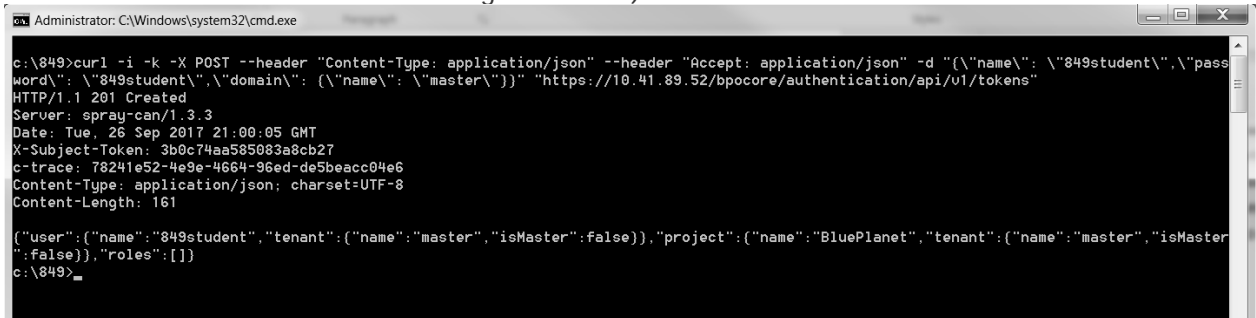
5. Open the `supplemental_files/codesnippets.txt` file with a text editor. Locate the comment `"-cURL command without file redirection"` under the `"=== TASK 2: TOKEN AUTHENTICATION ==="`. Underneath this comment is the cURL command you are going to use to exchange your permission token. This version of the command sends the output of the command to your screen so you may view the results. Later in this task, you will redirect the output to a file so you have access to your permission token.
6. Copy the cURL command, and paste it into the command window you have open in the 855 directory. The result should look like the following screen shot.

```
curl -i -k -X POST --header "Content-Type: application/json" --header
"Accept: application/json" -d "{\"name\": \"855student\", \"password\":
\"855student\", \"domain\": {\"name\": \"master\"}}"
```

```
"https://localhost/bpocore/authentication/api/v1/tokens"
```

```
c:\849>curl -i -k -X POST --header "Content-Type: application/json" --header "Accept: application/json" -d "{\"name\": \"849student\", \"password\": \"849student\", \"domain\": {\"name\": \"master\"}}\" \"https://10.41.89.52/bpocore/authentication/api/v1/tokens"
```

- Execute the command and preview the results in your command window. Be sure to note the value of the "x-subject-token" response header. This is your permission token. (Results of the command call should be like the following screenshot)

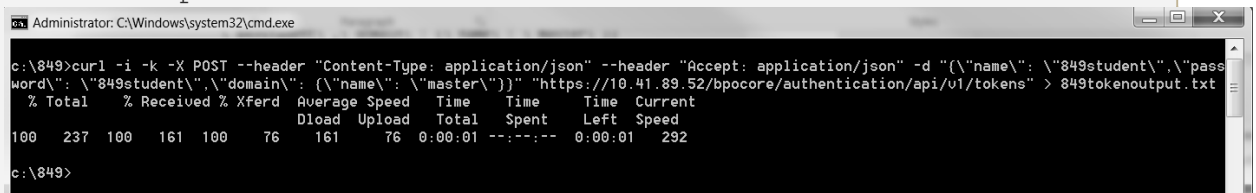


```
Administrator: C:\Windows\system32\cmd.exe
c:\849>curl -i -k -X POST --header "Content-Type: application/json" --header "Accept: application/json" -d "{\"name\": \"849student\", \"password\": \"849student\", \"domain\": {\"name\": \"master\"}}\" \"https://10.41.89.52/bpocore/authentication/api/v1/tokens"
HTTP/1.1 201 Created
Server: spray-can/1.3.3
Date: Tue, 26 Sep 2017 21:00:05 GMT
X-Subject-Token: 3b0c74aa585083a8cb27
c-trace: 78241e52-4e9e-4664-96ed-de5beacc04e6
Content-Type: application/json; charset=UTF-8
Content-Length: 161

{"user":{"name":"849student","tenant":{"name":"master","isMaster":false},"project":{"name":"BluePlanet","tenant":{"name":"master","isMaster":false},"roles":[]}}
c:\849>
```

- Move back to the supplemental_files/codesnippets.txt file that is open in your text editor. Locate the comment "-cURL command with file redirection" under the "=== TASK 2 TOKEN AUTHENTICATION ===". Underneath this comment is the cURL command you are going to use to exchange your permission token. This version of the command sends the output of the command to a text file in the current working directory so you may store the results.
- Copy the cURL command, and paste it into the command window you have open in the 855 directory. The result should look like the following screen shot.

```
curl -i -k -X POST --header "Content-Type: application/json" --header "Accept: application/json" -d "{\"name\": \"855student\", \"password\": \"855student\", \"domain\": {\"name\": \"master\"}}\" \"https://localhost/bpocore/authentication/api/v1/tokens" > 855tokenoutput.txt
```



```
Administrator: C:\Windows\system32\cmd.exe
c:\849>curl -i -k -X POST --header "Content-Type: application/json" --header "Accept: application/json" -d "{\"name\": \"849student\", \"password\": \"849student\", \"domain\": {\"name\": \"master\"}}\" \"https://10.41.89.52/bpocore/authentication/api/v1/tokens" > 849tokenoutput.txt
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 237 100 161 100 76 161 76 0:00:01 --:--:-- 0:00:01 292

c:\849>
```

- Open the 855tokenoutput.txt file located in the current working directory, using a text editor. Locate the "x-subject-token" response header. The value of this key is your 24-hour authentication token and you will be using it in successive labs. The results in the file should be like the following results.

```
HTTP/1.1 201 Created
Server: spray-can/1.3.3
Date: Tue, 26 Sep 2017 21:08:16 GMT
X-Subject-Token: 27a554a722e304bd5c18
c-trace: f19dc321-0881-4780-85f2-a54bcd800c9b
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 161
```

```
{ "user": { "name": "855student", "tenant": { "name": "master", "isMaster": false } }, "
  project": { "name": "BluePlanet", "tenant": { "name": "master", "isMaster": false
  } }, "roles": [ ] }
```

11. The value of the X-Subject-Token will be added to subsequent cURL calls as part of the header that is passed to the REST Server. Subsequent exercises will use the token
`--header "Authorization: token [YourTokenHere]"`
12. Feel free to leave Swagger, the command prompt / shell, and your text files open. You will be using these utilities in successive labs.
13. Give someone around you a high-five. You now know how to exchange credentials for an authentication token using Swagger, and cURL.

Task 3: Display information about the Number Pool product using cURL

In this exercise, you are going to use a cURL command to generate a report of information about the NumberPool product. This information will be used, in later labs, to create a resource instance from the NumberPool product, verify that your resource was created, and delete the resource you created.

If it is not open already, open a command shell, and navigate to the 855 directory you created.

1. Open the `supplemental_files/codesnippets.txt` file.
2. Navigate to the section titled `=== TASK 3: Display information about the Number Pool product using cURL ===`
3. Take notice of the cURL command. The command uses an exact string match query to return information about the product with the `resourceTypeId` of `tosca.resourceTypes.numberpool`. You will be creating a resource based on this product in the next lab.
4. Replace the value of the `Authorization: token [tokenhere]` with your token. Be sure to omit the square brackets. The brackets are only there to denote where you should be making the replacement.
5. Copy the cURL line into your command shell and execute the command.
6. Preview the results in a text viewer, they should be like the following:

```
HTTP/1.1 200 OK
Server: spray-can/1.3.3
Tue, 26 Sep 2017 05:08:14 GMT
c-trace: e35e10fb-3a55-40e0-8e7b-020f29c057c2
Content-Type: application/json; charset=UTF-8
Content-Length: 318

{"items":
```

```
[{"id": "f97d1a40-3ff1-5a9d-9bcc-83f65d96ef56",
  "resourceTypeId": "tosca.resourceTypes.NumberPool",
  "title": "Number Pool",
  "description": "Creates a pool of numbers from which to assign values to
    resources.",
  "active": true,
  "domainId": "built-in",
  "constraints": {},
  "providerData": {}
}]
, "total": 1,
  "offset": 0,
  "limit": 1000
}
```

7. Take note of the `id` value. This is the value of the `productId` key that you will be using to generate a resource in the next lab. Each instance of an object has a unique `id`, however, in the next exercise, you will be creating a resource based from this product. The template for the resource needs to have several parameters specified. Chief is its own globally unique identifier, its `id`. Second is the `id` of the product the resource is going to be based on. That value is the `id` of this product you just returned. In the subsequent lab, the value of the `productId` (the product the resource is based on) will be the `id` of this product.

Task 4: Use Swagger to create an instance of a resource

In this lab, you will use Swagger to generate an instance of a resource from the `NumberPool` product you just interrogated. The resource will be built from a pre-supplied template that needs minimal modifications. Generating a resource using a template will put you into a position to be able to manage the resource using `cURL` commands, Swagger, and the BPO UI

1. Please open Google Chrome, or Mozilla Firefox if it isn't open, and navigate to the Swagger interface at <https://localhost/bpocore/swagger#> (please replace server ip or name if necessary)
2. In the left-hand Components navigation pane, click on the Market Component. The right-hand pane should list off the market components.
3. In the right-hand pane, please click on resources.
4. You are going to generate a new resource based on the `NumberPool` product. To accomplish this, you will use a JSON template to supply the appropriate data to the API using Swagger. Once the resource is created, you will verify its existence using a `cURL` command, as well as the BPO UI.
5. Locate the POST link attached to `/resources` Create a new resource on the market, and click on POST.

6. Here you will supply the contents of the supplied template to the resource parameter under the Parameters section. If you are successful, a success exit status code 201 will be returned.
7. Open the `supplemental_files/codesnippets.txt` file.
8. Navigate to the section titled `=== TASK 4: Use Swagger to create an instance of a resource ===`
9. Copy the contents of the JSON object and paste it into the text area attached to the resource parameter in Swagger. The code you are copying looks like this:

```
{
  "id": "5a01344c-f051-4c39-8b63-ce23c49031a1",
  "autoClean": true,
  "createdAt": "2017-11-07T04:19:24.950Z",
  "description": null,
  "desiredOrchState": "active",
  "differences": [],
  "discovered": false,
  "label": "MyTestPool",
  "nativeState": null,
  "orchState": "active",
  "orderId": null,
  "productId": "f97d1a40-3ff1-5a9d-9bcc-83f65d96ef56",
  "properties": {
    "highest": 4,
    "lowest": 1
  }
}
```

e/swagger#

Response Content Type application/json ▼

Parameters

Parameter	Value	Description	Parameter Type	Data Type
resource	<pre>{ "id": "5a01344c-f051-4c39-8b63-ce23c49031a1", "autoClean": true, "createdAt": "2017-11-07T04:19:24.950Z", "description": null, "desiredOrchState": "active", "differences": [], "discovered": false, "label": "MyTestPool", "nativeState": null, "orchState": "active", "orderId": null, "productId": "f97d1a40-3ff1-5a9d-9bcc-83f65d96ef56", "properties": { "highest": 4, "lowest": 1 } }</pre>	Specification defining a resource	body	Model Model Schema

Parameter content type: application/json ▼

validate false (default) ▼

Whether to perform custom validation in addition to built-in schema and accessor validations

query boolean

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Bad Request		
401	Unauthorized		
403	Forbidden		
409	Conflict		
500	Internal Server Error		
503	Service Unavailable		

[Try it out!](#) [Hide Response](#)

Curl

10. (Please note, if you decide to create more than one instance of these resources, you will need to change the value of the id: parameter in the code. These values are hex decimal values a support the values of 1-9a-f in each position. It is easiest to change the last value in the id which is currently 1 to other another, higher value to generate more instances)
11. Swagger will return an exit status code of 201 if you are successful. If you aren't successful, go back and troubleshoot your issue. Beyond the obvious issues such as a power outage or your services aren't running, make sure you are using a unique id, check the validity of your JSON at [JSONLint.com](https://jsonlint.com), make sure you copied the template as is, or any other issue you may have introduced.

Your successful results should look something like this:



12. Look at the value of the label parameter for your NumberPool resource. It should be MyTestPool. You will use this label as a lookup filter in a subsequent lab to verify your resource was created. If you changed the label before you created the resource, take note of the change because you will need that label in the subsequent labs.

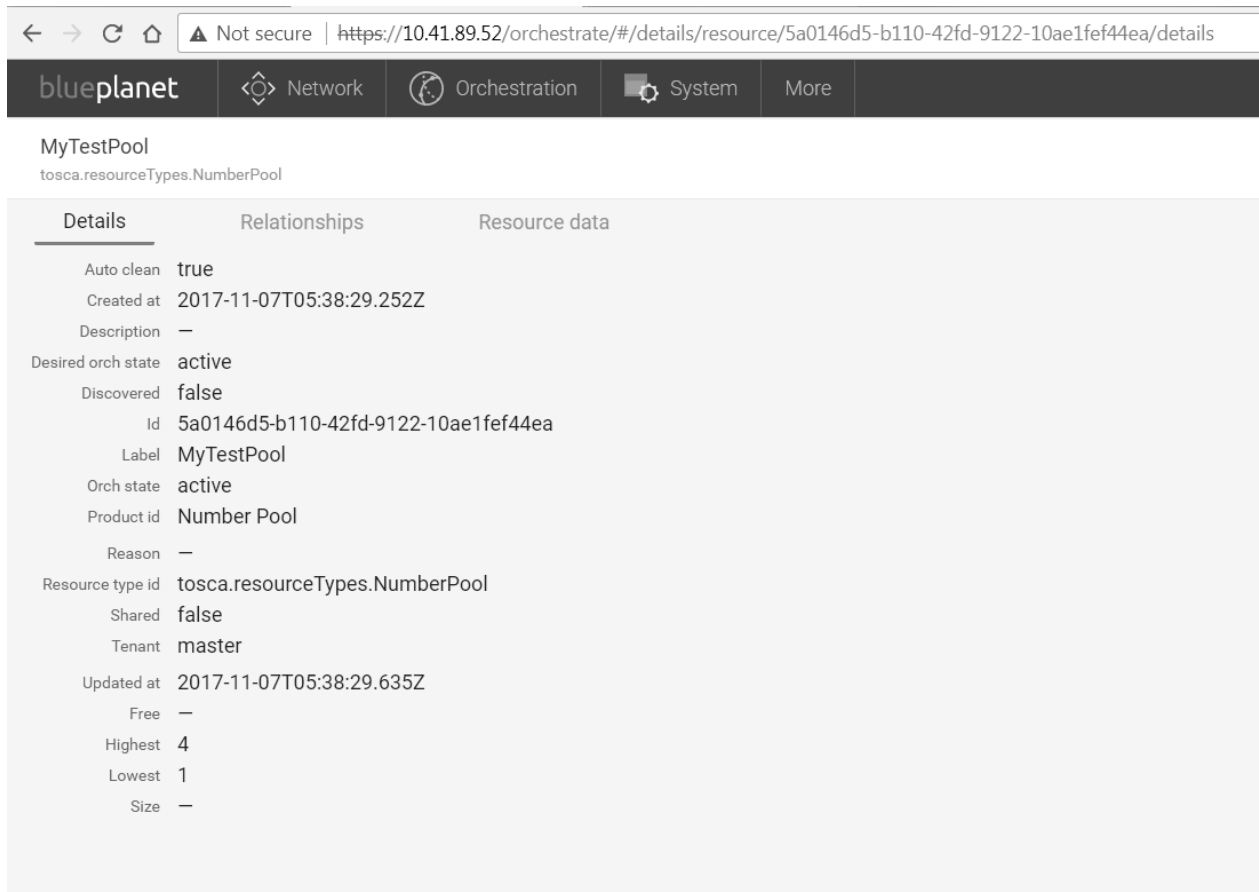
Task 5: Use cURL and the BPO UI to verify the resource was created

1. In this lab, you will use both the BPO UI and cURL commands to verify that your resource was properly created. You may also use Swagger if you like, but this lab will not explicitly ask you to use Swagger.
2. You will now use the BPO UI to verify that the instance of the NumberPool was created.
3. Please open your instance of BPO at <https://localhost/> or whatever IP/server name you are using.
4. Authenticate with the credentials supplied by your instructor, or your own credentials if you are running on your own machine.
5. Click on the Network link at the top, followed by clicking the All Resources link.

6. In the Resources search window, please search for the label “MyTestPool” (without quotes). Make sure that the sort by window is set to label. You should get a result for your resource. If not.... Go back to the previous lab and troubleshoot the process of creating the resource. A successful return should yield a result such as this.



7. Feel free to click through on your resource. Take note of the Details tab. Also, take notice that the id for the resource is in the address bar. The Details tab and the Resource data tab are very useful for viewing the metadata of your resource. However, it is convenient to know that the id of the resource is accessible from the address bar when you are viewing the resource.



8. In the next portion of this exercise, you will use a cURL command using an exact text search to return the metadata attached to the NumberPool resource you created.
9. Open the `supplemental_files/codesnippets.txt` file.

10. Navigate to the section titled `=== Task 5: Use cURL and the BPO UI to verify the resource was created ===`
11. Copy the cURL command and paste the command into your command shell.
12. Run the command. The output will be available in the `resMyTestPool.txt` file.
13. Open `resMyTestPool.txt` with a text editor and view the content. The file should contain a 200 OK status code, and the details for your resource. The content will be like the following.

```
HTTP/1.1 200 OK
Server: spray-can/1.3.3
Tue, 26 Sep 2017 06:46:40 GMT
c-trace: d197aae3-5dde-405d-a909-bc16eba5933e
Content-Type: application/json; charset=UTF-8
Content-Length: 551

{"items":[{"id":"5a0146d5-b110-42fd-9122-10aelfef44ea","label":"MyTestPool",
"resourceTypeId":"tosca.resourceTypes.NumberPool","productId":"f97d1a40-3ff1-5a9d-9bcc-83f65d96ef56","tenantId":"0dd6fdee-cell-46e8-a825-9257658f6a91","shared":false,"properties":{"highest":4,"lowest":1},"discovered":false,"differences":[],"desiredOrchState":"active","orchState":"active","reason":"","tags":{"providerData":{"state":[1,4,0]},"updatedAt":"2017-11-07T05:38:29.635Z","createdAt":"2017-11-07T05:38:29.252Z","autoClean":true}], "total":1,"offset":0,"limit":1000}
```

Feel free to leave your browser, and command shell open. You will be using those tools in the next lab.

Task 6: Delete the resource you created with cURL

1. Deleting a resource is simple. The Delete API requires the id of the resource you want to delete. So long as you have the id and the resource exists, the API will delete the resource.
2. Look at the previous lab and use either method you used to verify the existence of the resource to obtain the id for the resource you created.
3. Save the id in a text file. You will need to use it in the subsequent cURL command.
4. Open the `supplemental_files/codesnippets.txt` file.
5. Navigate to the section titled `=== Task 6: Delete the resource you created with cURL ===`
6. Replace the [ID] component in the URL with the resource's id.

```
curl -i -k -X DELETE --header "Accept: application/json" --header
"Authorization: token d3936103133d34e4ab10"
https://localhost/bpocore/market/api/v1/resources/[ID]?validate=false"
```

7. Check the output of the delete command. A successful call will return a 204 No Content status code like the following.

```
HTTP/1.1 204 No Content
Server: spray-can/1.3.3
Tue, 26 Sep 2017 07:13:27 GMT
c-trace: 32bd4380-7257-4a27-a65b-d3c5732a61a3
Content-Length: 0
Verify that the resource no longer exists by using either method from the
previous lab.
```

8. Upon verification, feel free to save and close your work. Get a cup of coffee, and give someone a high five. You did good work today.