Ex 2 - Tachyon v5.2 - Using - Postman as a Consumer

On this page:

Installing Postman

Now that we have Tachyon configured to allow external API calls. We need to install our API tool. The next steps will show you how to install Postman. This is the API interface we will use to first illustrate the things that can be done with the APIs. This course is not designed to teach you how to use Postman but instead the concepts here will translate to the API tool of your choice.

1ETRNAP

1. Open the *SkyTap Shared Drive* shortcut on the desktop. Navigate to **1E Tachyon - Course Content\1E Tachyon 5.1 Course Content** and download **Postman-win32-7.14.0-Setup.exe**. When the download finishes move the file to **c:\temp**
   The password to the OneDrive for the lab content is Passw0rd

1ETRNW102

2. Log into **1ETRNW102** as **1ETRN\Tachyon_AdminPP**
3. Navigate to **\\1ETRNAP\temp** and copy **Postman-win32-7.14.0.exe** to **c:\Sources**
4. Double-click the **Postman** executable you just copied and install *Postman*
5. On the *Create Account* screen click the **Skip signing in and take me straight to the app** at the very bottom (it appears greyed out but click on it anyway)
6. In the upper right click the *Wrench* icon and go into **Settings**. In *General* the third item down is *SSL Certificate Verification* - turn this off.
   We could import our certificate into Postman and leave this turned on but for simplicity sake we will turn this off.

Using Postman

1ETRNW102

7. From the *LaunchPad* click on the **Plus sign** to Open an *Untitled Request*
8. Let's first test our connection to Tachyon. In the *Enter Request URL* field in the center next to *Get* type in **https://Tachyon.1etrn.local/Consumer/SystemInformation**
9. Click the *Authorization* tab below that line. In the *Type* field choose **NTLM Authentication (Beta)** and type in **1ETRN\Tachyon_adminpp** and **Passw0rd**
   We must use an account with permissions to each of the API endpoints we will be making calls to. In production you will want to allow your consumer principal access to the least amount of securables as necessary for the functions that the Consumer will be doing. In our lab we will be using our Tachyon Global Administrator account just to show an example of most of the endpoints that are available.
10. Click on **Headers** and in the *Key* field type in **X-Tachyon-Consumer** in the *Value* field type in **Post** click **Send**
11. Notice the **ErrorCode** - *Consumer.UnlicensedConsumer.*
    Each of the calls must have a header that names the exact *Consumer* name registered in *Tachyon*
12. Amend Post in the value field to **Postman** click **Send**
    The consumer value is not case sensitive so in our lab postman or POSTMAN will work, but Pastman will not.
13. You should get a status of *200 Ok* and see the **Tachyon System Information** in the **Body** area

Examples of API Calls

In Tachyon there many different calls that we can do. Think of the things you can do in the Settings Application (system configuration type items) and things you can do in the Explorer Application (interacting with the devices that are reporting into Tachyon). We will do some of each of these things.

Device API Calls

1ETRNW102

14. Click the **Plus sign** to open a *new request*
15. In the *Request URL* Field type in **https://Tachyon.1etrn.local/Consumer/Devices** leave the type as Get
    You will need to change your Authorization to NTLM each time (your credentials will still be in there saved by Postman - this may not be the case for other tools) and you will need to fill in your header information each time since each request is sent and handled as an individual item.
16. Click on **Headers** and in the *Key* field type in **X-Tachyon-Consumer** in the *Value* field type in **Postman**
17. Click **Send**
    You will see a status of *200 Ok* each time and the body of the response will contain the data that we asked for in the *Get* statement
18. Notice the listing of all of our Devices in the body
19. Now we will Get a specific device click the **Plus sign** to open a new request and type in **https://Tachyon.1etrn.local/Consumer/Devices/fqdn/1ETRNW71.1ETRN.Local**
20. Click on **Headers** and in the *Key* field type in **X-Tachyon-Consumer** in the *Value* field type in **Postman** click **Send**
21. Notice the *Status 400 Bad Request*. FQDN for this API call must be encoded in Base64 format
    The Tachyon SDK Documentation will provide you with guidance when specfic formats or parameters are required. A link to the SDK documentation is included in the Links to Developer Resources page at the end of this Lab Guide.
22. Open a new tab in *Chrome* and navigate to **base64encode.org** - Select **Encode** at the top and Type in **1ETRNW71.1etrn.local** in the top form and click on the green **Encode**. Copy the characters from the bottom pane into your *Postman Request URL* (replacing the 1ETRNW71.1ETRN.Local). Click **Send**
    In this example your base64 encoded value will be **MWV0cm53NzEuMWV0cm4ubG9jYWw=**
23. Notice the *Status 200* and the details of our device

Defining Scope For the Query

Next we will define a scope for our query to limit our results to a particular type of device. In this example we are only wanting to return our Servers.

24. Perform a *Post* on **https://Tachyon.1etrn.local/Consumer/Devices/Scope**
25. Set *NTLM Authorization* and Add your **X-Tachyon-Consumer** header value to **Postman**. Also add a second header in the Key field start typing in **Content** and choose **Content-Type** in the Value field start typing in **application** and Choose **application/json**. This is the header we need to add when passing some parameters with our request.
26. Click on the *Body* tab and choose the **raw** radio button
27. In the Code Block type in the following

```
{
        "Attribute" : "DevType",
        "Operator" : "Like",
        "Value" : "Server"
}
```

28. Click **Send**
29. Notice our results pane - only the Server device types are returned.
    The scope attributes that are available are very specific. See the link in the Resources page at the end of this course for additional attributes that we did not use in our examples.

Device Management Groups

30. Base64 encode **1ETRNAP.1etrn.local** and use in the next step request
31. Now perform the following *Get* **https://Tachyon.1etrn.local/Consumer/Devices/FQDN/<Base64 Encoded Name>/ManagementGroups**
32. Scroll down and notice the *Management Groups* that this device is a member of

Device Criticality Mapping

33. Now perform a *Get* on **https://Tachyon.1etrn.local/Consumer/Devices/CriticalityMapping**
    This returns the *Device Criticality* from the *Criticality Mapping Table* in the **Tachyon Master DataBase**. This tells us that our *Critical* devices are set to **5**

Instruction API Calls

Now we will work with our API calls that do things with Instructions and Instruction Sets.

34. Perform a *Get* on **https://Tachyon.1etrn.local/Consumer/InstructionSets**
35. Notice the listing of all of our *Instruction Sets* that have been uploaded to Tachyon
36. Now perform a *Get* on **https://Tachyon.1etrn.local/Consumer/InstructionDefinitions**
37. Notice all of the instruction definitions. Look at all of the Instruction Set IDs. You will need to use these in later labs.
38. Perform a *Get* on **https://Tachyon.1etrn.local/Consumer/InstructionSets/Id/16**
39. Notice it returns our *Device Criticality Instruction Set*
40. Perform a *Get* on **https://Tachyon.1etrn.local/Consumer/InstructionDefinitions?instructionType=Question**
41. Notice that this returns all of the instructions that are Questions. Only the instructions that the user can see per their permissions are returned. The ones that the account does not have permission to are omitted from the return.
42. Perform a *Get* on **https://Tachyon.1etrn.local/Consumer/Instructions/InFlight/InstructionSet/16**
43. Notice we have **0 Running** instructions for our *Device Criticality Instruction Set*
44. Open *Chrome* and navigate to **Explorer Application**. Ask the question **What is the Criticality of my Devices?** and change the *Gather duration* to **30 minutes**
45. Go back to *Postman* and **resend** the last **Get**
46. Notice we now have **1 InFlight** instruction for that *instruction set*
    When you are deleting instructions you must be careful to check for any that are InFlight as you do not want to delete any InFlight instructions. Instead cancel the instruction before you delete it.

Delete an Instruction Set without deleting the Instructions

The next exercise will show you how to delete an instruction set without deleting the instruction definitions.

47. Open the *Settings Application* in *Chrome* and navigate to **Instructions - Instruction Sets** and select the *Misc Instruction Set.*
48. Note that we have **35** instructions in the *Misc Instruction Set* and **0** instructions in the *Unassigned Instruction Set*
49. Back in *Postman* perform a *Delete* on **https://Tachyon.1etrn.local/Consumer/InstructionSets/Id/18?deleteContent=false**
    Adding the deleteContent set to False will leave our Instruction Definitions but they will move to the Unassigned Instruction Set. deleteContent=true will also delete any instruction definitions that are members of the instruction set. We could have also used https://Tachyon.1etrn.local/Consumer/InstructionSets/Id/18/Clear to remove them all from an instruction set and keep the definitions.
50. Navigate back into the **Settings Application** and notice that we now have **35** instructions in the *Unassigned Instruction Set* and no *Misc Instruction Set.* You may have to refresh your page.

Create a New Instruction Set

We will now create an instruction set to replace the one we just deleted.

51. Perform a *Post* on **https://Tachyon.1etrn.local/Consumer/InstructionSets**
52. Change to *NTLM Authorization* and Add your standard header of **X-Tachyon-Consumer** with a value of **Postman**. Still in Headers also add a header of **Content-Type** with a value of **application/json**
    You can start typing in the header and Postman intellisense will allow you to select the one that matches your typing - using this is suggested as it will eliminate syntax errors.
53. Click the *Body* tab and choose the *Raw* radio button. Ensure that *json* is selected in the far right column (in Orange)
54. Type in the following in the *Body* Code Block

```
{
"Name" : "Msc",
"Description" : "Replaces our deleted Instruction Set"
}
```

55. Click **Send**
56. Notice our Status *201 Created*
57. Navigate to the *Settings Application* and ensure that the *Instruction Set* was created. Notice it contains **0** Instruction Definitions.

Add a Specific Instruction Definition to an Instruction Set

You will need to type in **https://Tachyon.1etrn.local/Consumer** in your Request URL field including the rest of the URL but in the next exercises we will refer to the URL as **<RestOfURL>** as this is the way it appears both in our Documentation and in Swagger.

58. Back in *Postman* open a new tab and perform a *Get* on **InstructionSets** find the Instruction Set ID of the new *Msc* instruction set we created
59. Perform a *Post* on **InstructionSets/Contents**
60. Add our *header* for a payload **(Content-Type with a Value of application/json)**
61. Select **Body** and choose **Raw**
62. Type the following in your *Body* code block

```
{
"SetID" : <enter the number from your Get>,
"InstructionDefinitionIds" : [56,57]
}
```

63. Navigate to Settings application and check to make sure your instructions (Delete all coverage tags and Delete all freeform tags) have been moved into your instruction set.

Create a Question

Now we will use Postman to ask a question. We will refer to it as creating a question. We are going to ask What is the Criticality of My Devices?

64. Perform a *Post* on **Instructions**
65. Add your additional *header* of **Content-Type value = application/json**
66. In your *body - raw* code block type in the following

```
{
"DefinitionId" : 28,
"InstructionTtlMinutes" : 120,
"ResponseTtlMinutes" : 120}
```

67. You should get a *Status* of **201 Created**
68. Open the *Explorer Application* and Notice that the Question is Running
    You could also look at the Tachyon.ConsumerAPI.log on the Tachyon Server (1ETRNAP)

Create a Question and Set Coverage to Specific Devices

Now we are going to ask a question "What is the location of my devices?" and limit the coverage to specific devices.

69. In *Postman* open a *new request* and perform a *Post* on **Instructions/Targeted**
    Fqdn list is limited by MaxRequestLength which defaults to 4MB
70. Add your Additional *Header* for a payload **(Content-Type value application/json)**
71. In your *body - raw* code block type in

```
{
        "DefinitionId" : 29,
        "InstructionTtlMinutes" : 15,
        "ResponseTtlMinutes" : 15,
        "Devices" : ["1ETRNW71.1ETRN.Local", "1ETRNW72.1ETRN.local", "1ETRNW73.1ETRN.local"]
}
```

    Here we are editing our *Instruction Time To Live* and our *Response Time To Live* and giving the Question a *coverage* of our three Windows 7 devices.
72. Click **Send** to create the Question
73. You should see the *Status* of **201 Created**

Check the Log for Status

1ETRNAP

74. Open the *Tachyon.ConsumerAPI.log* and see the details of the Request you just sent
75. Notice the line for *"Question Created"* with a Payload *"Get the location of my devices?"* and *Coverage* of our *Windows 7 devices*

Check the Responses in Explorer

1ETRNW102

76. Open the *Explorer Application* and check that the instruction is running
    We can still see the responses in Explorer because we did not offload them. If we had set them up for offloading they would be visible only in the application that was the source of our offloading.

Create a Question that Requires Parameters

Now we will create a question that requires us to enter some values at the time of issue.

1ETRNW102

77. Back in *Postman* click the **Plus sign** to open a *new request* window
78. Add your *Authorization* and your two *headers*
79. Perform a *Post* on **Instructions**
80. In your *Body - Raw* tab type in the following in your code block

```
{
        "DefinitionId" : 127,
        "InstructionTtlMinutes" : 15,
        "ResponseTtlMinutes" : 15,
        "Parameters" : [{
                "Name" : "hive",
                "Value" : "HKLM"
},{
                "Name" : "subkey",
                "Value" : "Software\\1E"
}]
}
```

Here we are creating a question What are all the subkeys for a specific reg key? Notice our parameters we have to define these exactly as called for the in the instruction definition. We are using the [ ] to enclose the entire array and each pair of name and value are in { } (curly braces) comma delimited.

81. Click **Send** to issue our request
82. Notice the *Status* **201 Created**
83. Check the log and the responses in *Explorer application*
    Notice that this Instruction is not *aggregated*. We will look at an instruction in a later exercise that is aggregated so that you can see the differences.

Create an Action

Now will we create an action and see the changes that are made to our devices. We will be setting a registry entry on our Windows 7 devices. Our instruction requires multifactor authentication and approval.

1ETRNW102

84. Back in *Postman* open a **new request** to perform a *Post* on **Instructions** and fill in the *Authorization* and the two *headers* that we need to add a payload to our request
85. For Authentication use the **Tachyon_Admin1** account. Change that in your *Authorization* tab in Postman
86. In the **Body - Raw** code block type in

```
{
        "DefinitionID" : 135,
        "InstructionTtlMinutes" : 15,
        "ResponseTtlMinutes" : 15,
        "Parameters" :
[{
        "Name" : "hive",
        "Value" : "HKLM"
},{
        "Name" : "subkey",
        "Value" : "Software\\1E"
},{
        "Name" : "name",
        "Value" : "Common"
},{
        "Name" : "valuetype",
        "Value" : "REG_SZ"
},{
        "Name" : "value",
        "Value" : "Postman"
}]
}
```

Consult the instruction definition when you are working with parameters for guidance. Notice that each pair (Name and Value) must correspond to the parameters of the instruction definition. For example, we are setting HKLM\Software\1E\Common to a string "Postman". Tachyon matches each pair to the proper label in the instruction definition. If we set name to Name we get an error on that parameter.

87. Click **Send**
88. You should see *Status* **201 Created**
89. Check *Tachyon.ConsumerAPI.log* for success messages
    Make note of the Request ID - you will need it for our approval.

1ETRNW71

90. Log into **1ETRNW71** as **1ETRN\Tachyon_Admin1**
91. Open *Live Mail* and get the authorization code for the instruction number you collected from the ConsumerAPI.log

92. Open *Explorer* and navigate to **Instructions** - **History**. Notice the instruction waiting for authentication. Open that instruction and click on the *Provide authentication code* button
93. Enter your authentication code from the email

1ETRNW102

Now we will approve the Action

94. Perform a *Post* on **Approvals/Instruction**
95. For authentication use **1ETRN\Tachyon_AdminPP** account
96. In the Body code block type in

```
{
        "InstructionId" : "XX",
        "Comment" : "Any comment here",
        "Approved" : "true"
}
```

97. Check the Log and the Explorer application for the results.
98. Open *Regedit* and navigate to **HKLM\Software\1E\** there will be a name of Common with the data set to *Postman*
    If you are working quickly through the labs you can come back to checking the registry for the results, if the key is not changed on this device yet.

Aggregation

Earlier we look at creating the question from instruction definition 127. This pulls all of the subkeys for a registry key. This instruction is not aggregated. Now we will look at an instruction that is aggregated. You will need to understand how your instruction returns the responses in order to be able to handle them to act on the data that is returned.

1ETRNW102

99. Create a question *What processes are running* by performing a *Post* on **Instructions**
100. After adding your *Headers* and your *Authorization* type the following in your *Body* field

```
{
        "DefinitionId" : 96,
        "InstructionTtlMinutes" : 120,
        "ResponseTtlMinutes" : 120
}
```

101. Click **Send**
102. Open Explorer application and take a look at the responses that are returned from each of the questions.
    As you can see the Instruction that is aggregated has the count of each instance of the item (in this case the processes that are running). Our registry key question has just the listing of each item. So we see the same line item for each device (for example HKLM\Software\1E will appear 7 individual times in our responses because we have 7 different devices in our lab). Each type of return has usefulness when you are acting upon the data that you are returning.

Showing both aggregated and non-aggregated responses

There are times when you want to have both the counts of the items and the listing of all items. We can do this by setting the KeepRaw flag.

103. Back in **Postman** in the **Body - Raw** section add the following line to your code block under "ResponseTtlMinutes" : 120 (you will need to add a comma to this line)

```
"KeepRaw" : 1
```

    Make sure to leave your closing brace below this new line
104. Open **Explorer** and navigate to **Instructions - History** and select the latest issue of the **What processes are running** instruction
105. Now you see the ability to see either the Raw table view or the Aggregated table view by switching the view using the buttons above the **Filter Results** button
106. Open the first **What processes are running** instruction and notice that we only have one view to see for these responses

Permissions

For our first step for permissions we will look at all of the permissions for a Role and then the permissions for a Principal

108. In *Postman* perform a **Get** on **Roles**
109. Notice "ID" : 7 is our Global Actioner
110. Now perform a **Get** on **Principals** this shows all of the users that are configured in Tachyon
111. Perform a **Get** on **Principals/Role/7** this will show you all of the users who are a Global Actioner in Tachyon
112. Perform a **Get** on **SecurableTypes** this returns all of the available securable types
     Notice for Id **1** this is our *instruction set* securable type. We have actioner, approver, questioner and viewer available for instruction sets. Refer to Swagger in order to see all of the API calls and the details for each call when you are creating your automation. There are many other things that can be done that we will not cover in this course.

Extra Credit

Take a look at the Resources section at the end of this lab guide. Navigate to the Consumer API Reference page and familiarize yourself with the different types of calls that you can make.

Lab Summary

In this lab we learned how to use Postman to interact with the Tachyon Platform. We learned about system activities (things we would do in the settings application) and also about interacting with the devices that are reporting into Tachyon (things we could do with the Explorer application). We also looked at aggregation and had our first look at Permissions.

Next Page

Ex 3 - Tachyon v5.2 - Using - Offloading Responses