IBM MQ Web

Messaging Administrator

Summary: Using the IBM MQ Console and REST interfaces

Lab 5 - Using the IBM MQ Console and REST Interfaces

Overview

View an overview of using IBM MQ Web

Note:
This lab utilizes the **Windows 10 x64** Skytap image. Start this image and then login with the userid of **iibdemo** and a password of **passw0rd** before you begin this lab.

You may use the same demo environment that you used for Lab 1. If you are doing this lab consecutively following Lab 1, you can continue with that environment. If you are doing this lab independent of Lab 1, you may need to create a new demo environment. In that case click the link below.

See the Environment Setup section

IBM MQ version 9 introduced new web based administrative interfaces for managing your IBM MQ queue managers. These interfaces include a browser-based Web UI as well as a set of REST APIs. The Web UI, referred to as the IBM MQ Console, supplies a user-friendly alternative to the IBM MQ Explorer and **runmqsc** command shell interfaces.

These interfaces are supported by a WebSphere Liberty Profile application server instance referred to as the MQWeb Server. The MQWeb Server component is packaged along with the other IBM MQ components and may be installed on any server that hosts a queue manager. The MQWeb Server is configured and started separately from your IBM MQ queue managers.

As a browser-based tool the IBM MQ Console offers certain advantages over command shell or eclipse-based tools like the IBM MQ Explorer. Advantages include avoiding the overhead of installing and

maintaining software remotely as well as being available across a much wider variety of platforms and devices.

The MQWeb Server also supports REST APIs that may be used to interact with IBM MQ objects such as queue managers and queues. Information is sent to, and received from, the administrative REST API in JSON format. The IBM MQ REST APIs are integrated with IBM MQ security which is enabled by default. You must configure security before you can use the REST APIs.

Configuring the MQWeb Server

In order to configure the MQWeb Server component you must first determine how you intend to configure security for the IBM MQ Console and the IBM MQ REST API interfaces. Security for the Console and the REST API is configured by editing the MQWeb Server configuration file named **mqwebuser.xml**.

The MQWeb Server supports the following authentication mechanisms:

- Basic registry
- LDAP registry
- Local O/S registry
- System Authorization Facility for z/OS

Roles can be assigned to IBM MQ Console users to determine what level of access they are granted to the widgets that are provided with the IBM MQ Console.

> **Note:**
> It is important to understand that the IBM MQ Object Authority Manager still controls *authorizations* to work with MQ objects. You will have an opportunity to explore this point later in this lab.

After a user is assigned a role there are a number of methods that can be used to authenticate the user. With the IBM MQ Console users can log in with a user name and password, or they can use client certificate authentication. With the IBM MQ REST APIs users can use basic HTTP authentication, token based authentication, or client certificate authentication.

Configure Security

> **Note:**
> For this lab you will configure security using the Basic registry. Refer to the IBM Knowledge Center for IBM MQ for information on how to configure security using the other options that are available. Note that the use of the Basic registry is not recommended for a production environment.

Complete the following steps to configure a Basic registry.

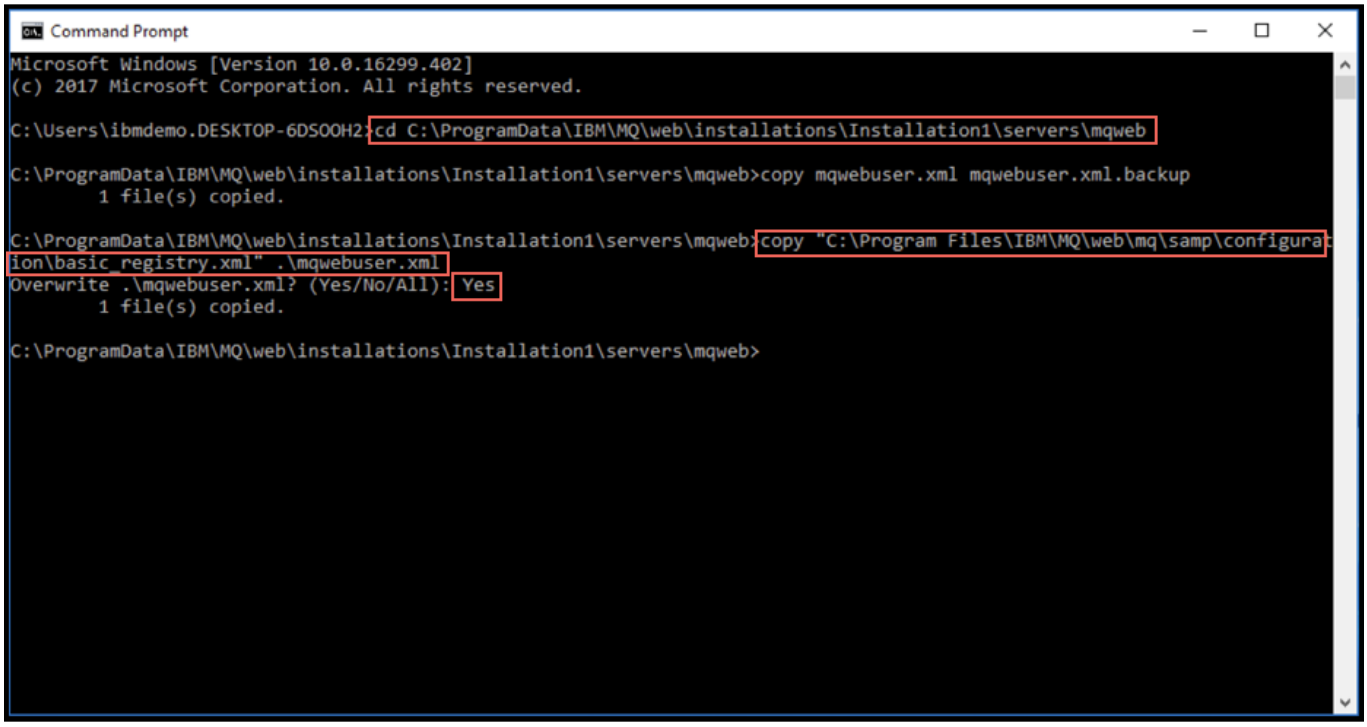
1. A default **mqwebuser.xml** file is provided when IBM MQ is first installed. Enter the following commands to create a backup copy of that file.

```
cd C:\ProgramData\IBM\MQ\web\installations\Installation1\servers\mqweb
copy mqwebuser.xml mqwebuser.xml.backup
```

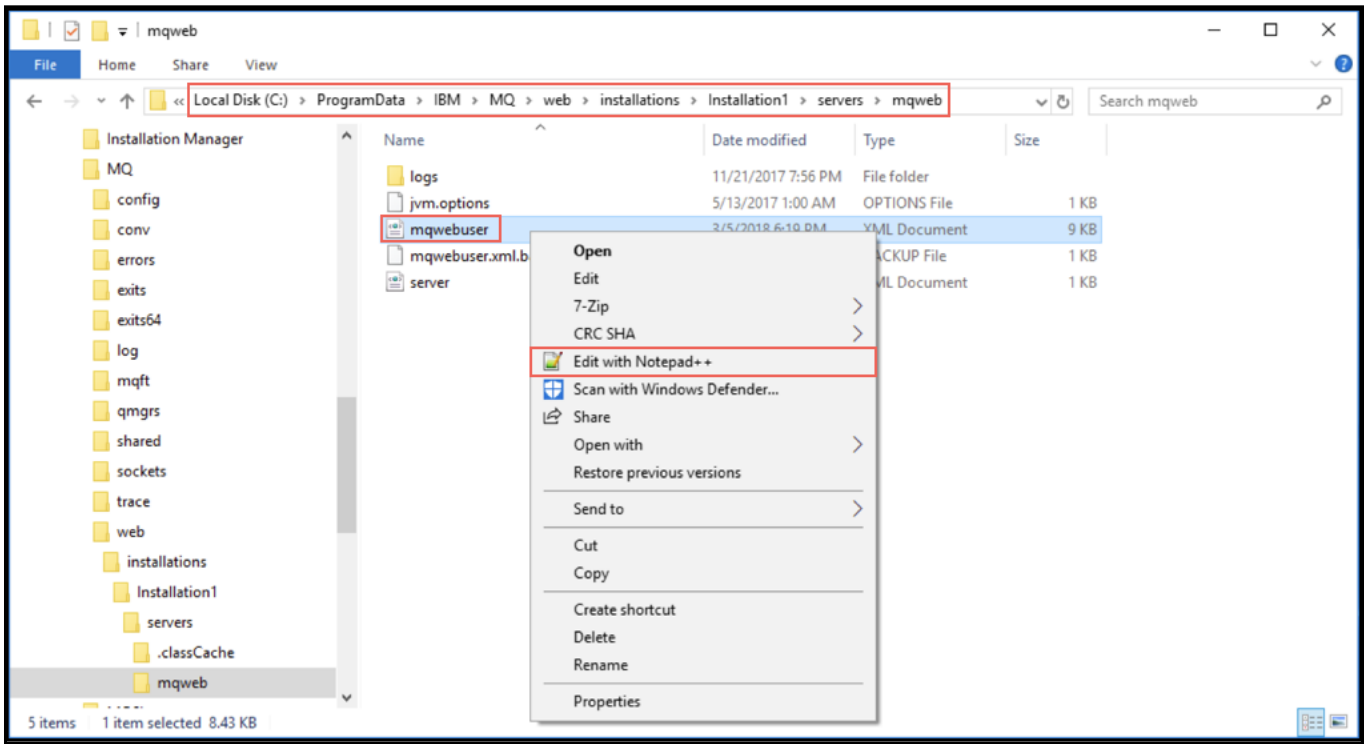
2. Enter the following command to copy the Basic registry template to the MQWeb Server directory.

```
copy "C:\Progam Files\IBM\MQ\web\mq\samp\configuration\basic_registry.xml" .\m
qwebuser.xml
```

> **Note:**
> Ensure that you include the the double quotes when entering this command. Also, respond **Yes** when asked if you want to overwrite the existing **mqwebuser.xml** file.

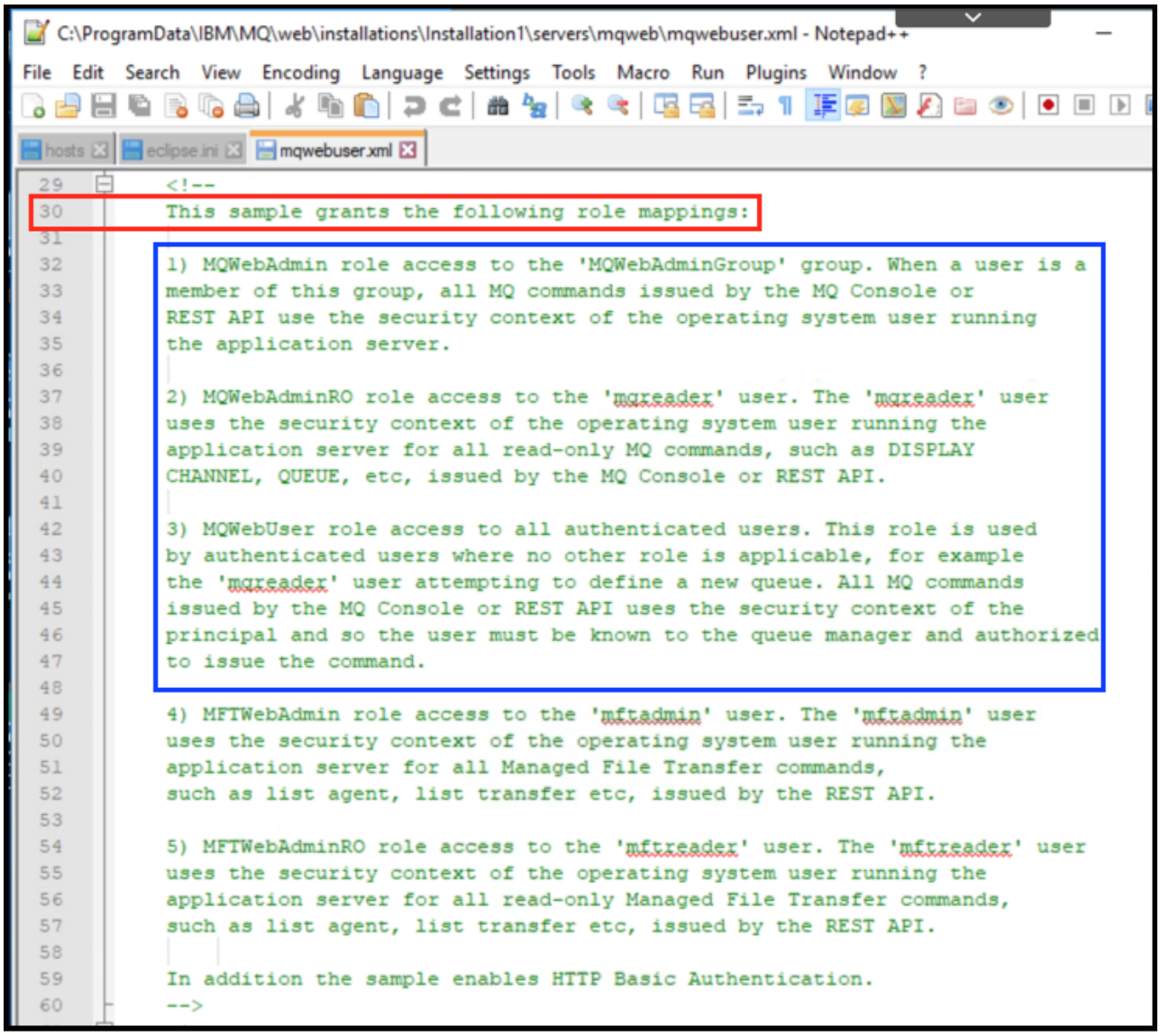


3. Use Notepad++ to open and review the **mqwebuser.xml** file.



4. Note the following:

   - On line 30 you will find a description of the sample roles that are provided with the Basic registry. Pay close attention to the differences in how authorization is determined between the **MQWebAdmin** and **MQAdminRO** roles as compared to the **MQWebUser** role.



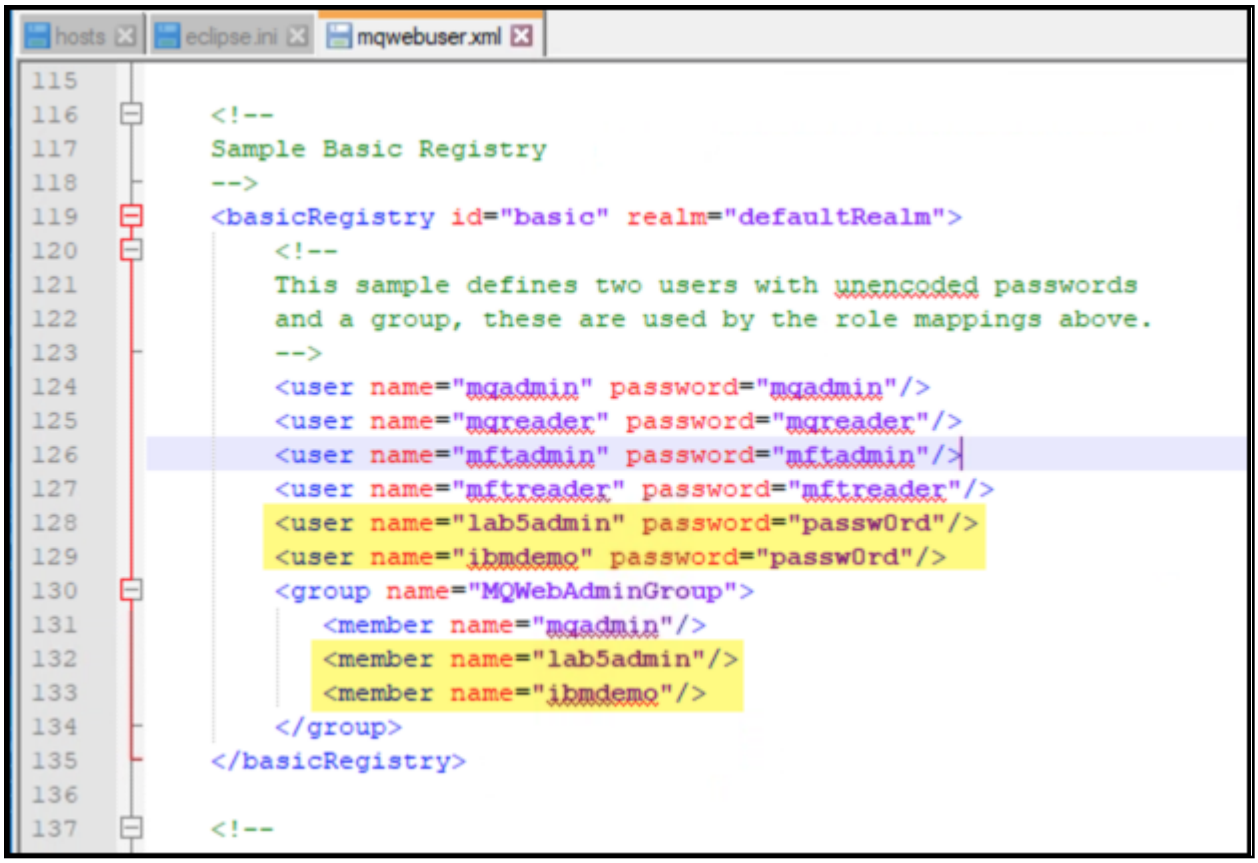
- On line 70 you will find where the default roles for users of the IBM MQ Console are defined.

> **Note:**
> By default the members of the **MQWebAdminGroup** *group* are assigned to the **MQWebAdmin** *role*.

- On line 93 you will find where the default roles for users of the Administrative APIs are defined.

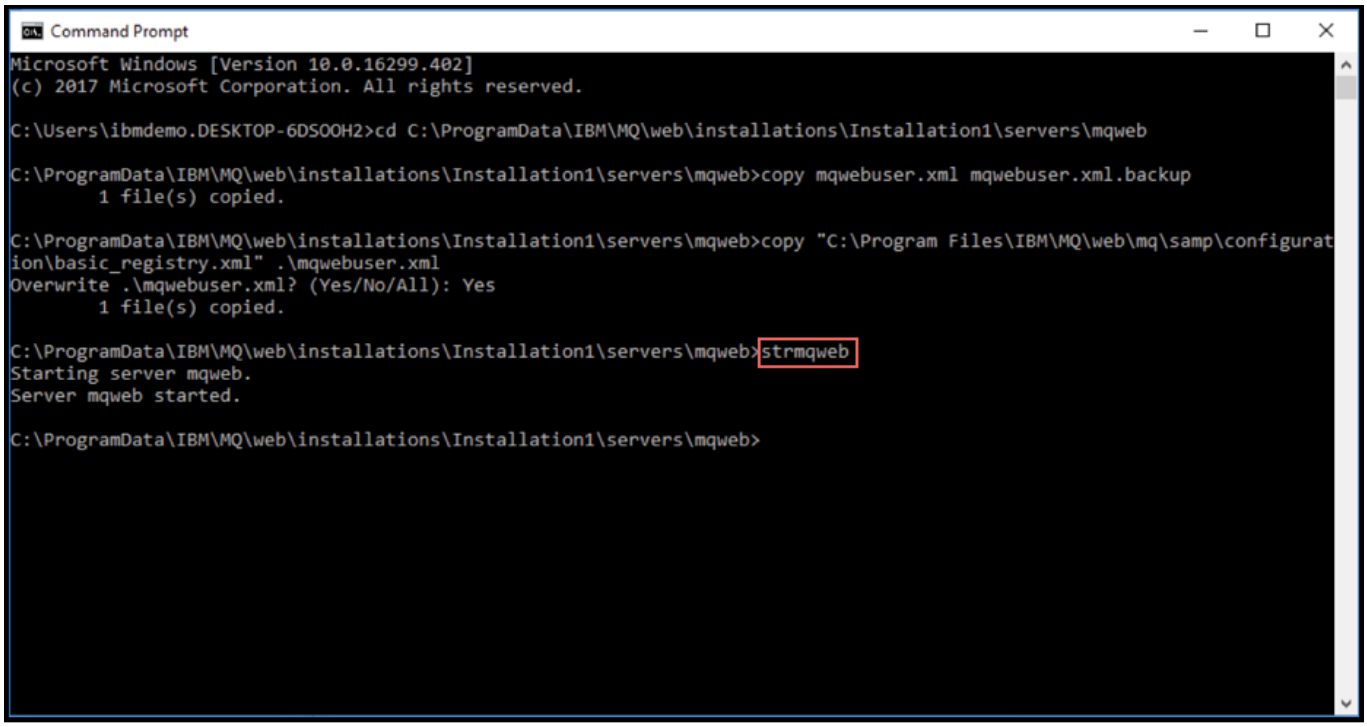
- On line 117 you will find where you can define userids, passwords and group membership for the Basic registry.

5. Create two userids named **lab5admin** and **ibmdemo** and set both of their passwords to **passw0rd**. Add both userids to the **MQWebAdminGroup** group.

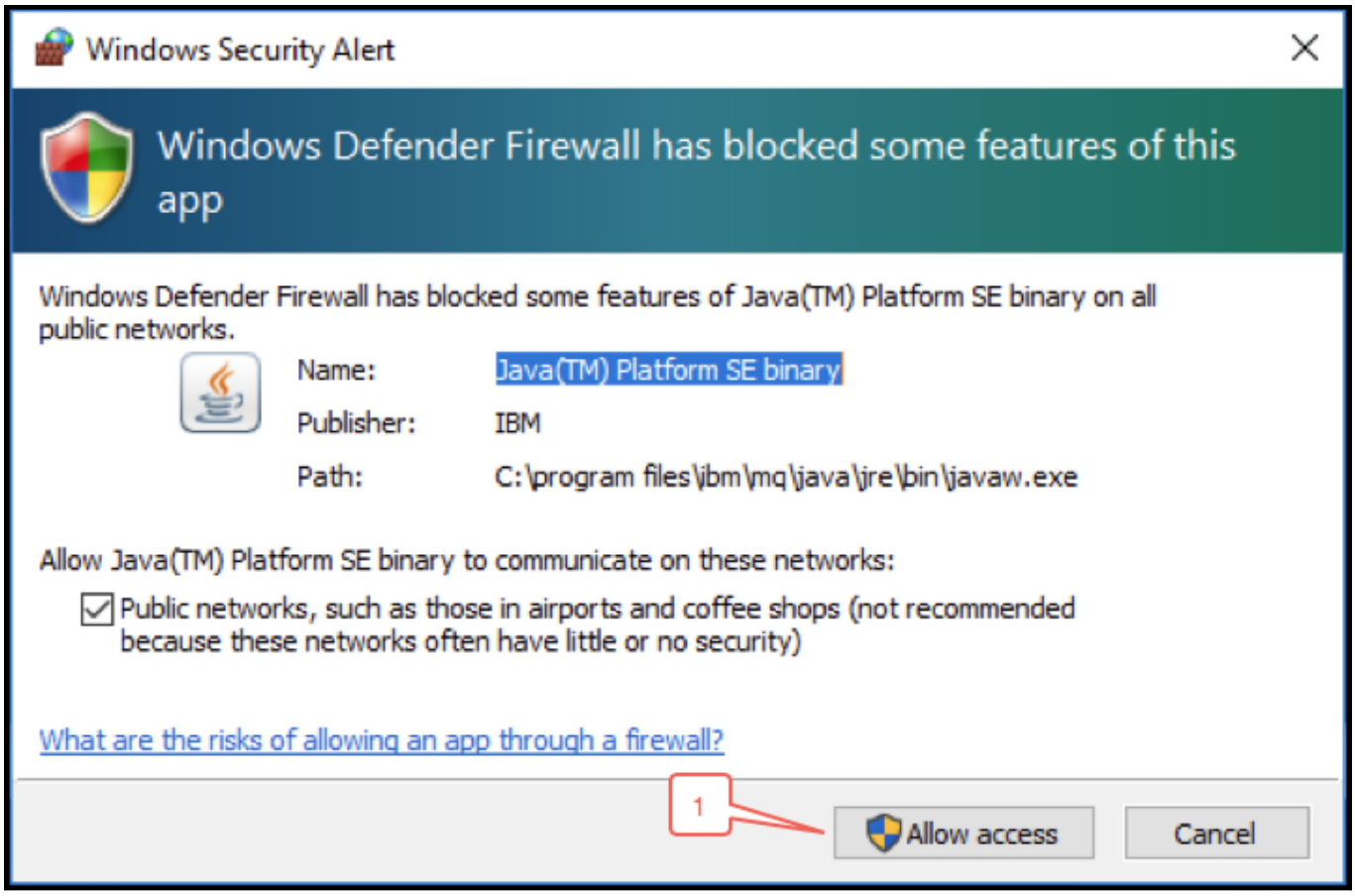


6. Save your changes to the file.
7. Review the remaining entries in the file to become familiar with the additional capabilities that you can configure with the Basic registry.
8. You have now completed the necessary steps to configure access to the MQWeb Server. Open a command prompt and enter the following command to start the server:

```
strmqweb
```



Note:
A **Windows Security Alert** popup may appear. Click on **Allow access** to continue.



Note:
By default the MQWeb Server does not start automatically when the server boots. Using the **strmqweb** from the command line will keep the MQWeb Server running for as long as you are logged in to the O/S. Also, for IBM MQ Console users that are assigned to the **MQWebAdmin** and **MQAdminRO** roles, the security context for any MQ actions will be based on the userid that invoked the **strmqweb** command from the command line. Follow normal O/S configuration steps to call the **strmqweb** batch file / shell script upon boot up in order to run it as a Windows service or as a Unix/Linux daemon.

Using the IBM MQ Console

The first MQWeb Server component that you will review is the **IBM MQ Console**. The Console is a browser-based Web UI that provides a user-friendly alternative to the IBM MQ Explorer and the **runmqsc** command shell for queue manager administration. It also enables users to administer queue managers from any workstation without the overhead of installing the Explorer.

Note:

Note that the IBM MQ Console does not currently offer all of the configuration options that the **runmqsc** or IBM MQ Explorer provide.

The IBM MQ Console was designed to allow users to create a more customized experience for monitoring and administering IBM MQ. Two main concepts to keep in mind are:

- **Dashboards** represent the presentation space that users create. These are highly customizable, and can be configured to suit a user's individual tastes.
- **Widgets** represent the object types that may be displayed on the dashboard. Each dashboard tab can hold a number of widgets, arranged in a grid.

The design of the IBM MQ Console supports the creation of multiple dashboards, enabling different views of the MQ resources being hosted by the server that the MQWeb Server is installed on. For example, dashboards can be created that offer views for different business applications that use MQ, with widgets showing the objects relevant to each application. Or, you could have a tab focused purely on monitoring, with charts showing the consumption of various resources over time (monitoring capabilities will be introduced later in this lab). Widgets representing MQ objects can be added, viewed and deleted from the dashboard as needed. In addition, some of the properties of the MQ objects represented by these widgets can be modified.

Complete the following steps to explore the various features of the IBM MQ Console.

1. Open the Chrome web browser and navigate to the IBM MQ Console home page using the following URL:

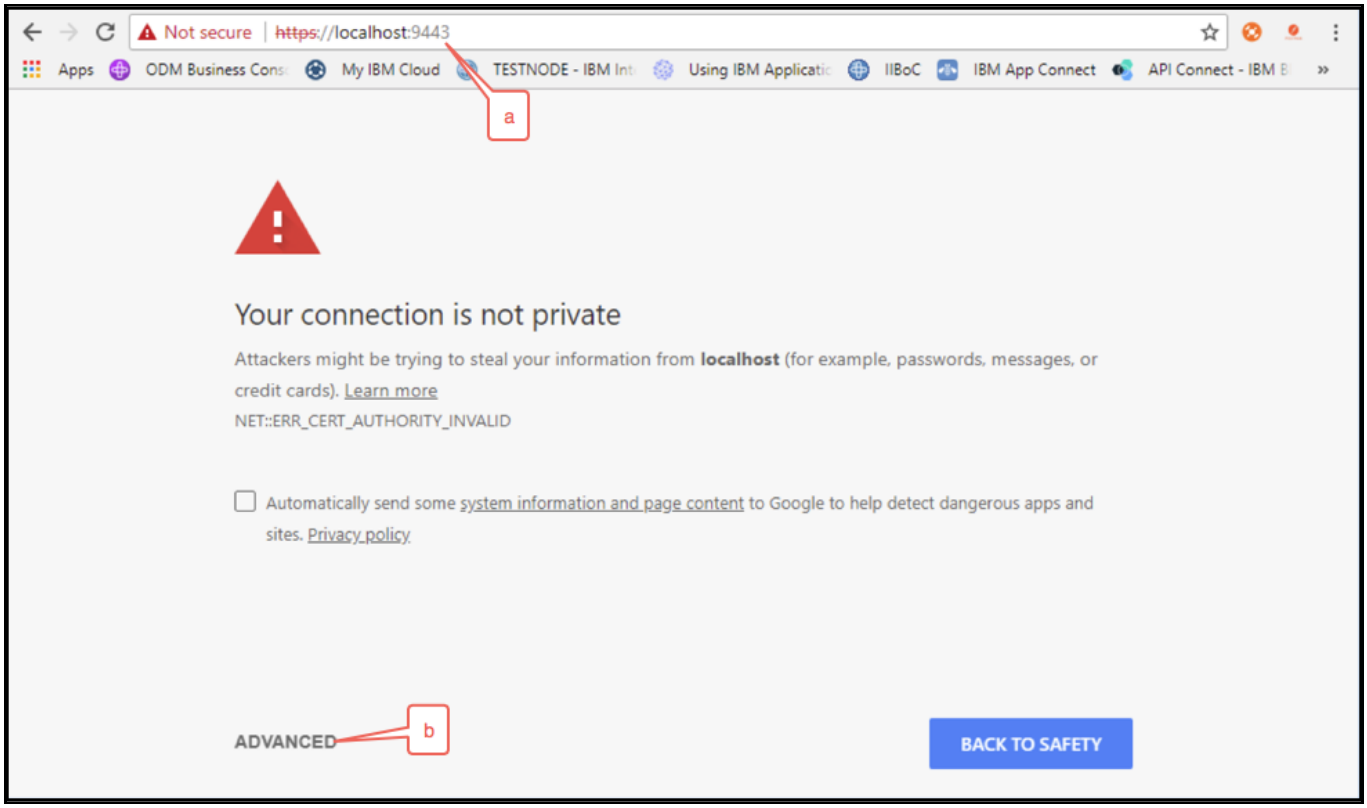
```
https://localhost:9443
```

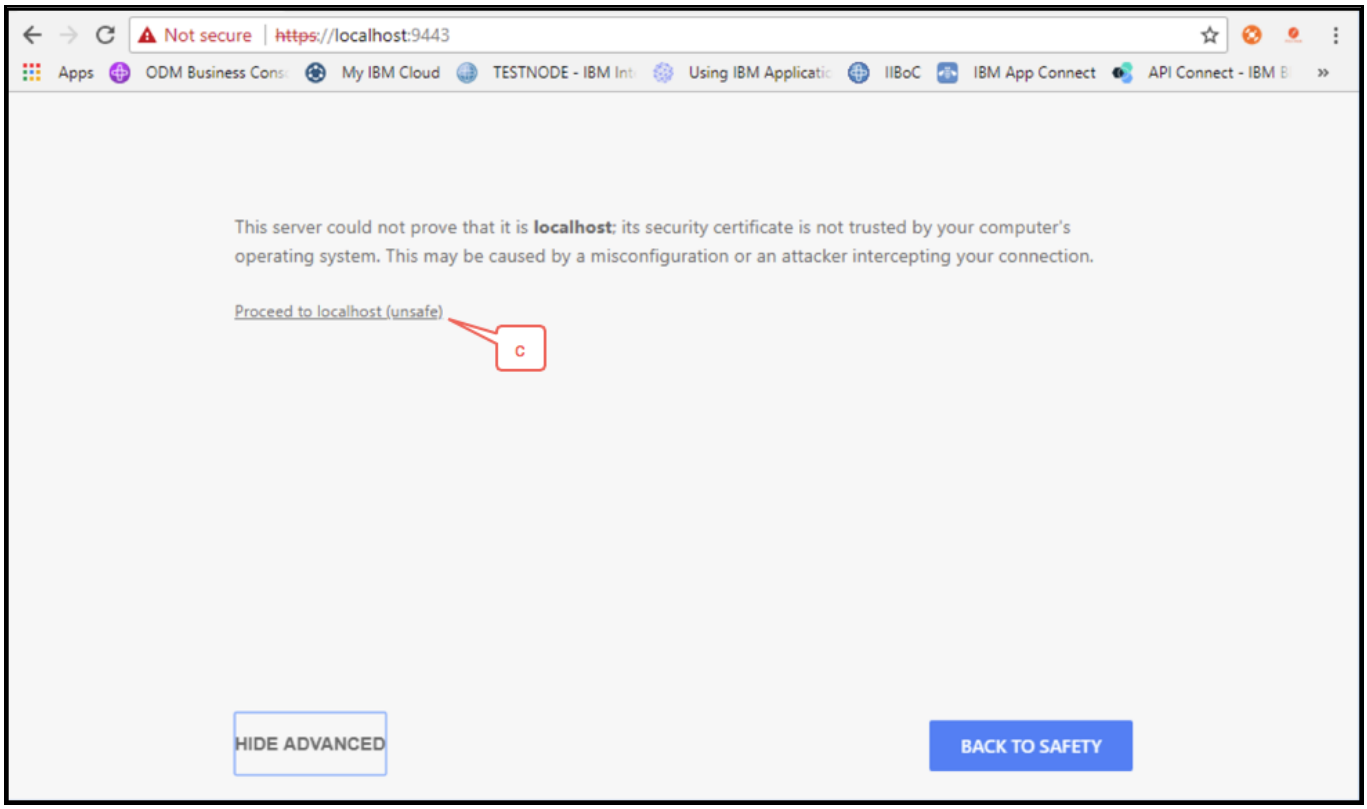
Note:
You can use any web browser to access the IBM MQ Console. Since you will need to use the Google Postman tool later in this exercise you should use the Chrome browser now in order to clear the warning regarding self signed certificates.

Tip:
If you need to recall the URL for the IBM MQ Console you can display it by opening a command prompt and entering the **dspmqweb** command.

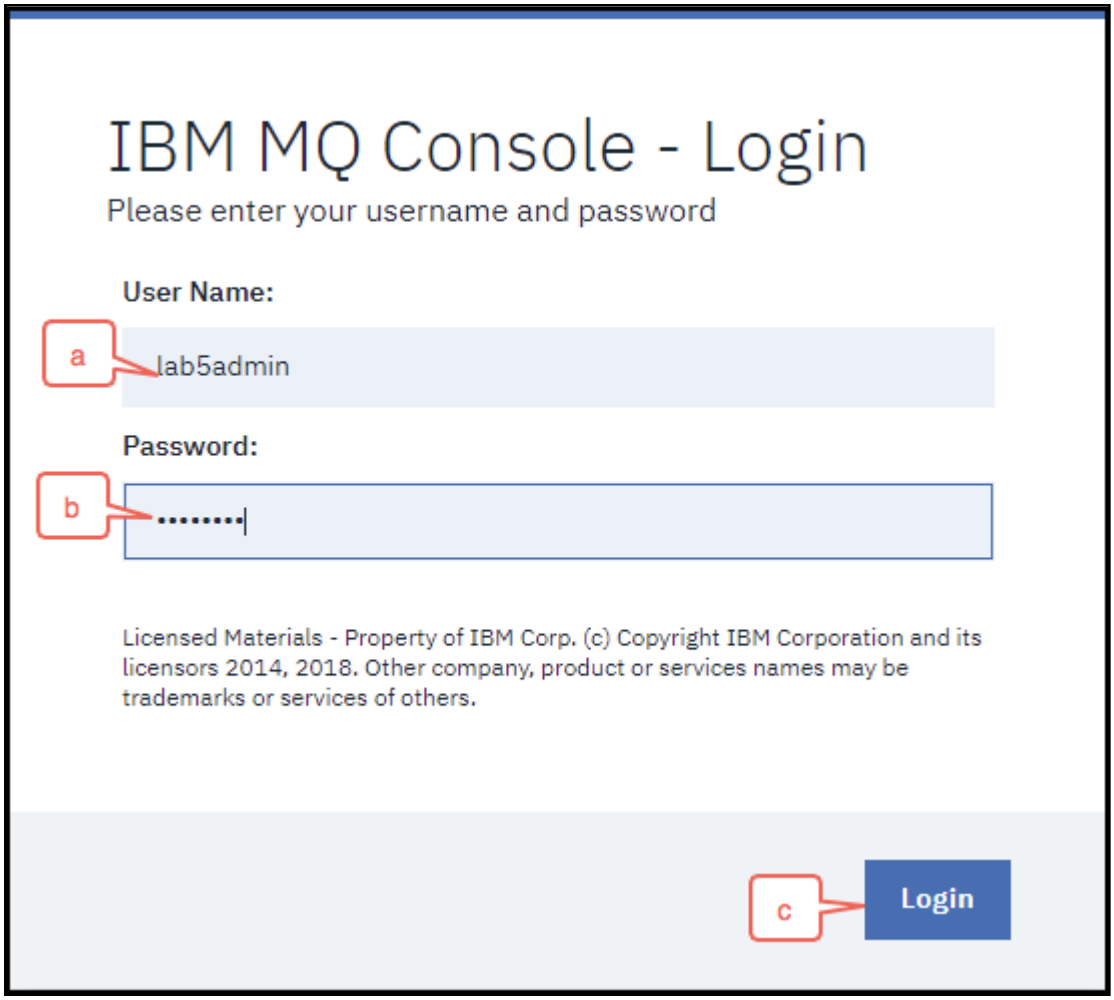
If this is the first time that you've accessed the IBM MQ Console you will received a warning (similar

to the following) regarding an untrusted certificate. Since the Console ships with a self-signed certificate this is normal behavior. Take the appropriate steps to continue to the website.





2. Once you reach the login page enter **lab5admin** as the User Name and **passw0rd** as the Password and then click on the **Login** button to continue.



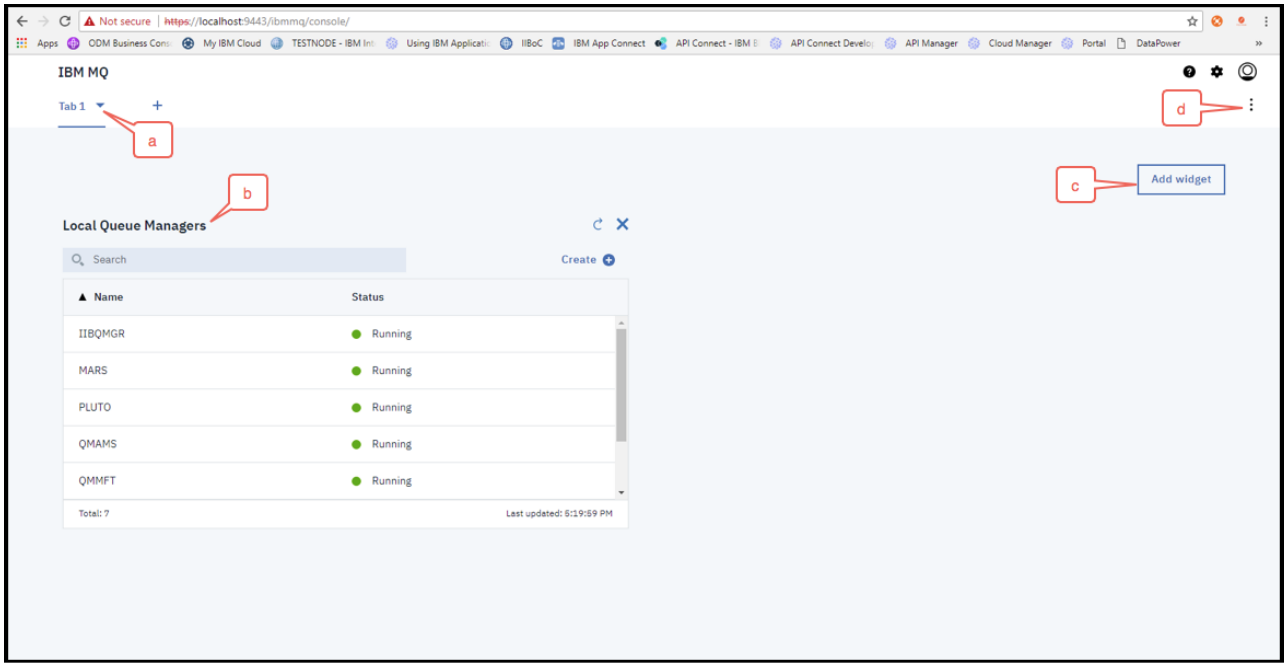
3. The initial dashboard is now displayed. Observe the following features of the dashboard:

   a. The initial dashboard contains a single tab. You may rename this tab by clicking on the drop down icon next to the tab name. You may add any number of additional tabs by clicking on the **+** character at the right of the existing tab(s).

   b. The initial dashboard contains a single widget that lists all of the local queue managers.

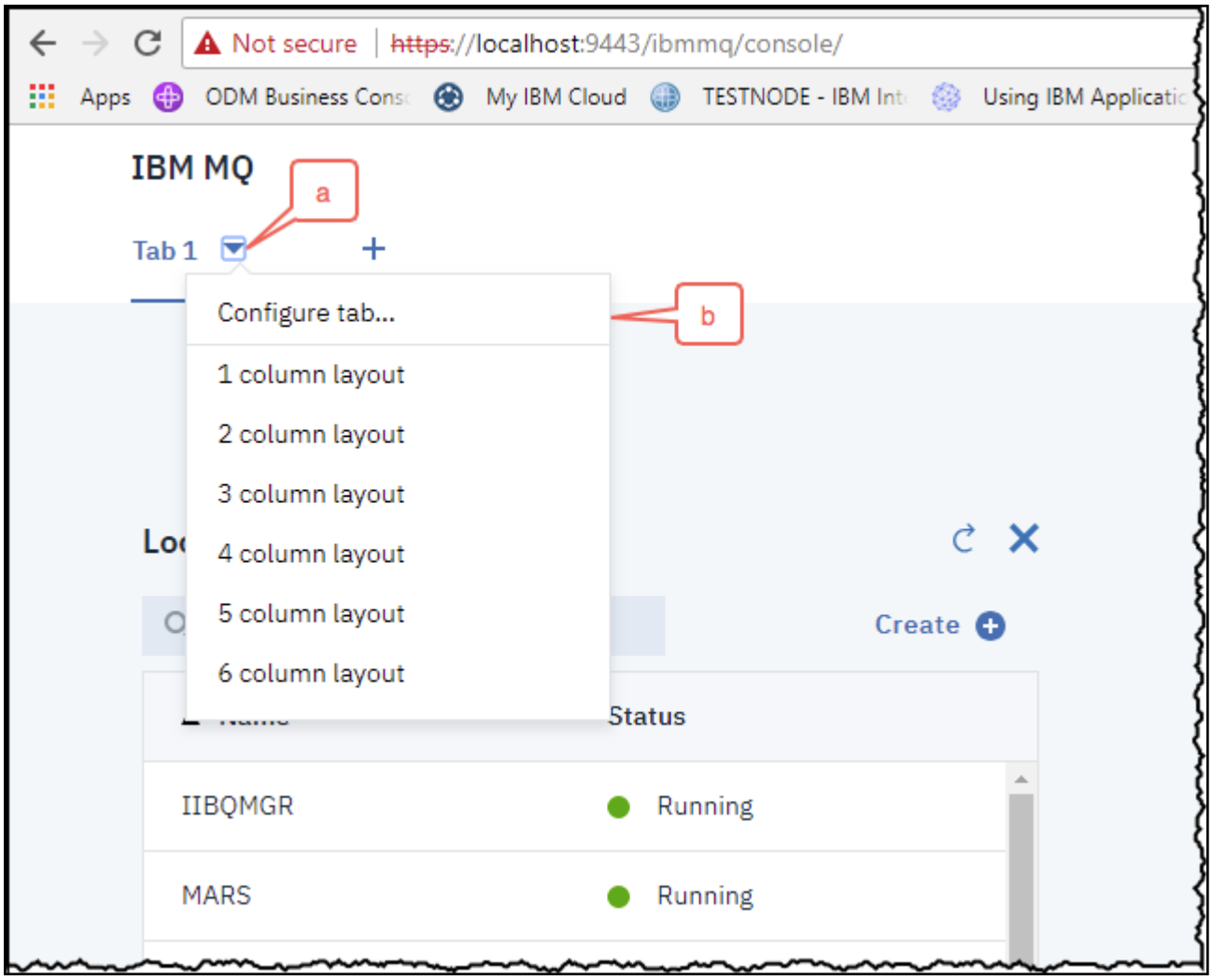
   c. You may add any number of additional widgets to a tab by clicking on the **Add Widget** button.

   d. There is a control menu (sometimes referred to as a “hamburger” menu) in the upper right corner of the dashboard. Clicking on this menu will display options to export or import your dashboard layout, configure miscellaneous settings and logout of the Console. You will explore exporting and importing your dashboard layout later in this lab.

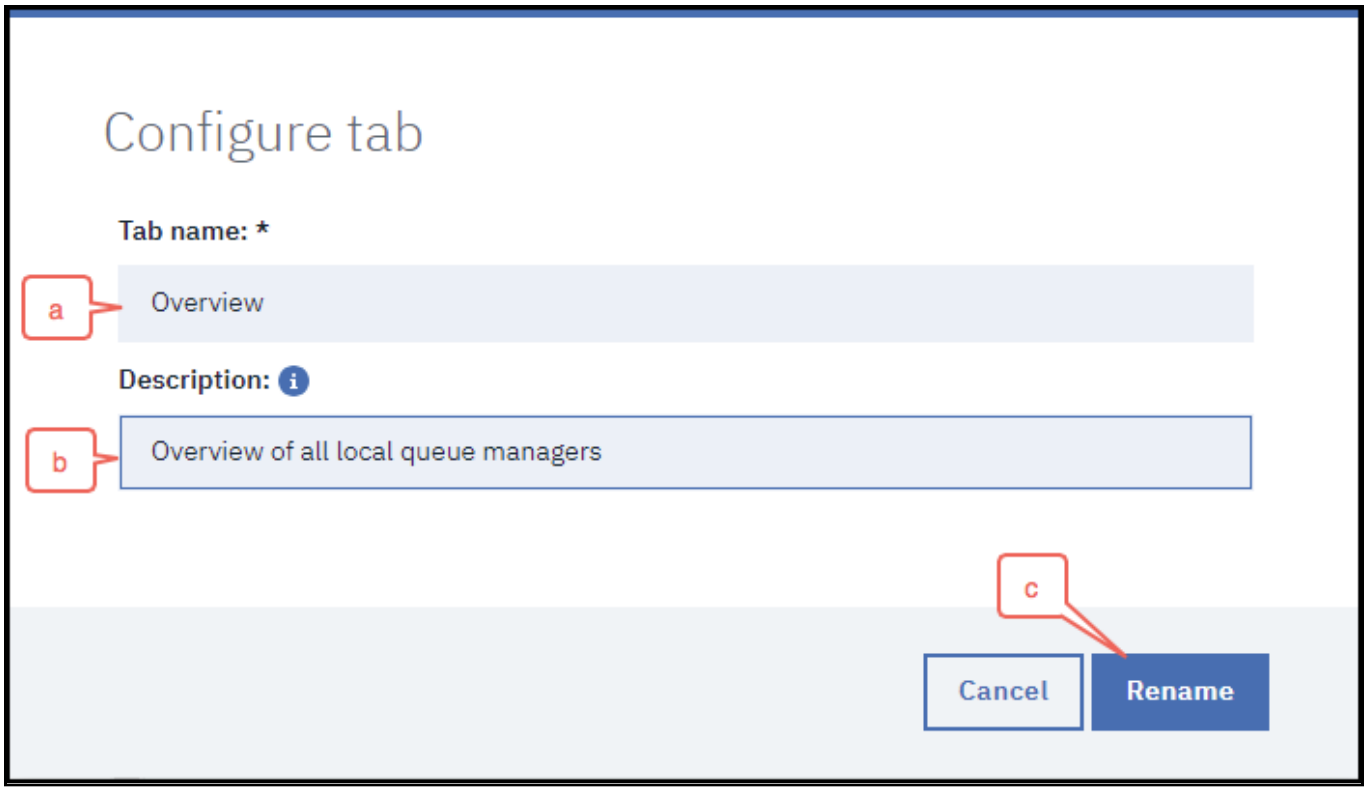


> **Note:**
> Your list of queue managers may not match the screen shot depending on which labs you may have previously completed.

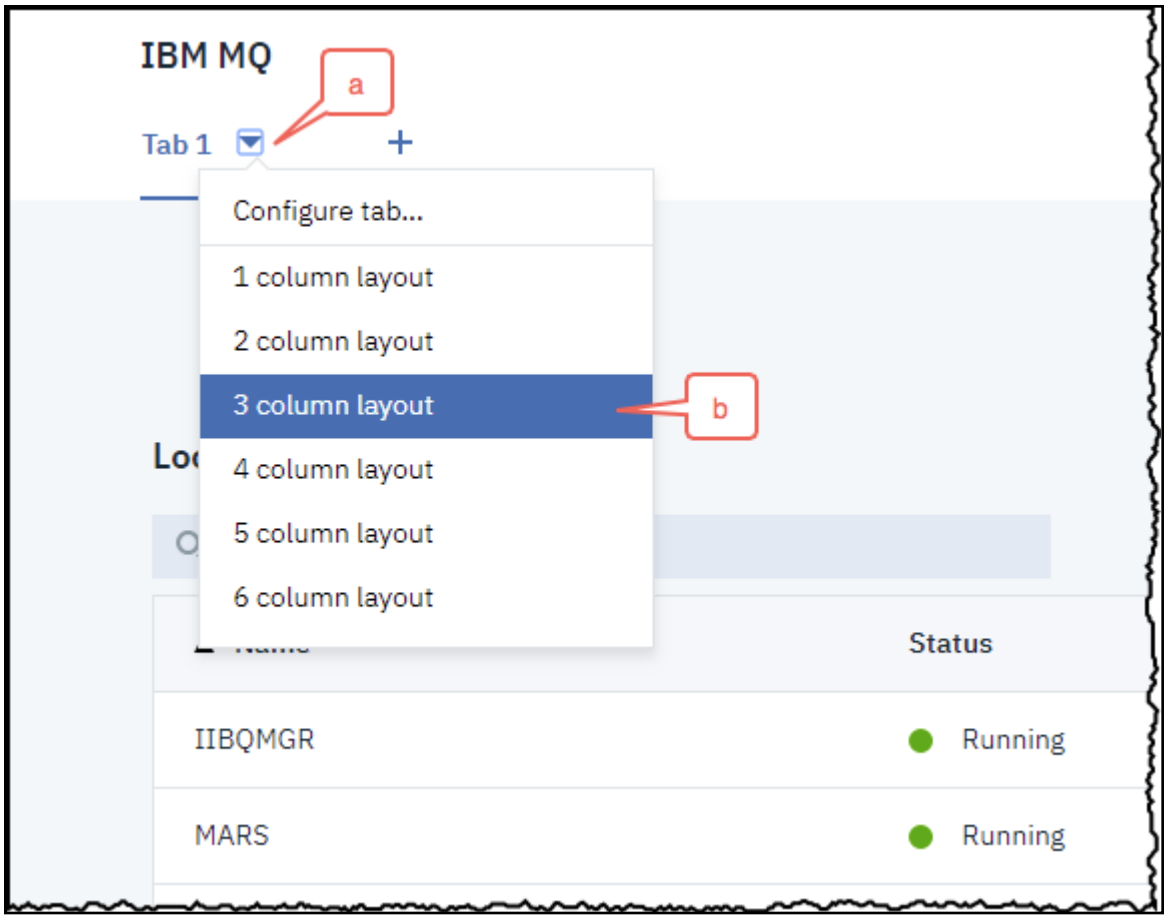
4. Start customizing the dashboard by changing the layout of Tab 1. Click on the drop down icon next to the tab name and then click on the **Configure tab…** menu item.



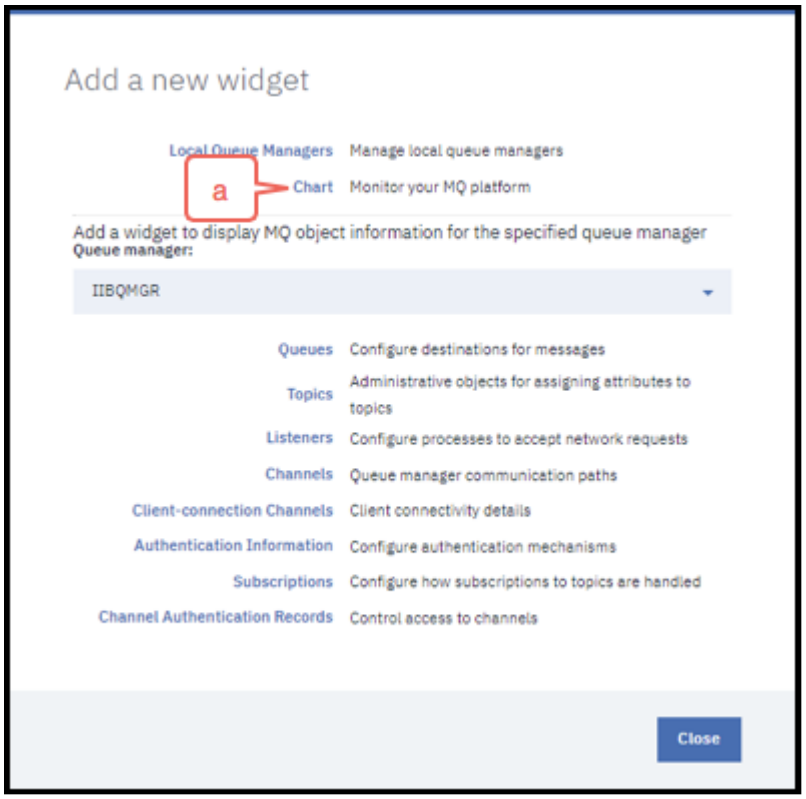
5. Change the name of the tab and then provide a brief description. Click on the **Rename** button when finished.



6. Now set the tab to display widgets in a 3 column layout. Click on the drop down icon next to the tab name again and then click on the **3 column layout** menu item.

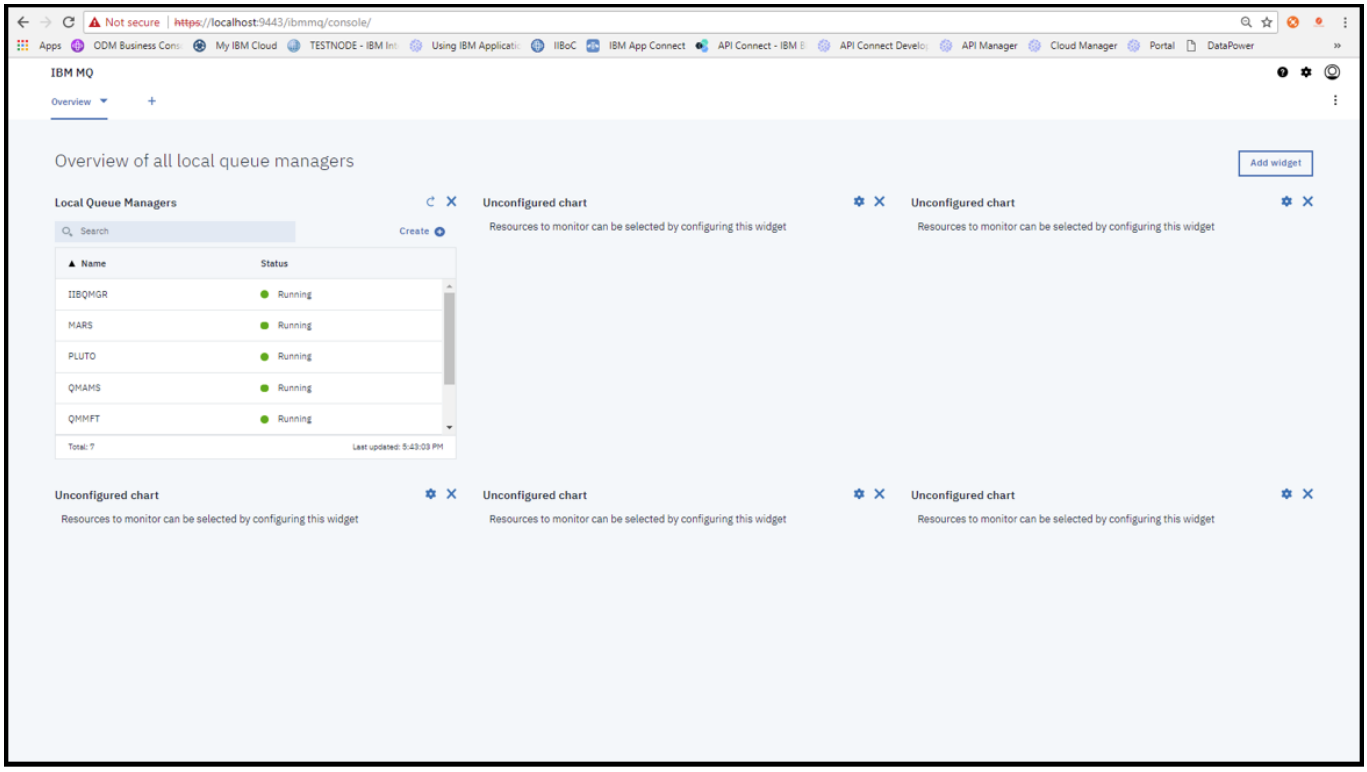


7. Now add some additional widgets to the tab. Start by clicking on the **Add Widget** button in the upper right side of the workspace. The **Add a new widget** dialog will appear.



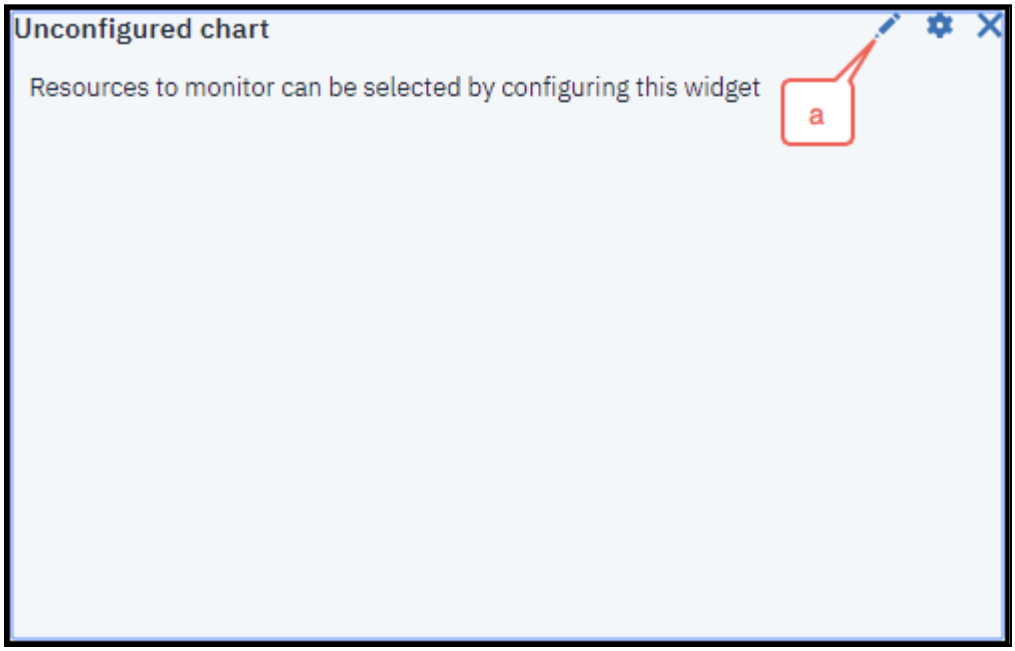
8. The upper portion of the dialog allows you to add widgets that may be used to display information for all local queue managers. The remainder of the dialog allows you to add widgets that display information for a specific queue manager. For the **Overview** tab you will include several Chart widgets that will provide a summary view of various metrics for the server you are working on.

   Click on the **Charts** link to add a Chart widget to the tab. Repeat this step to create a total of 5 Chart widgets.

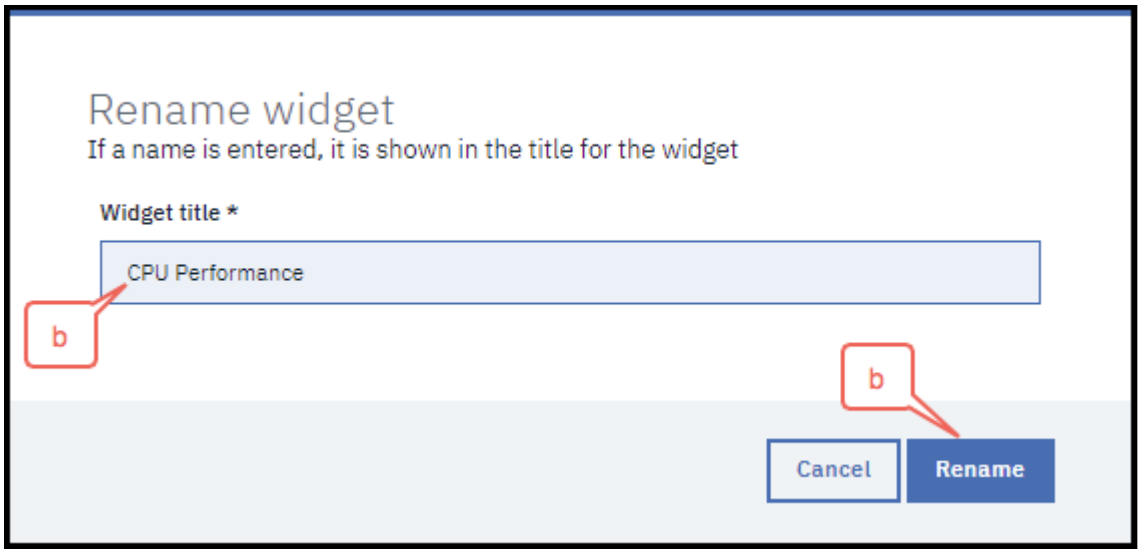


9. Tailor each Chart widget to show a different metric. Use the following screen captures as a guide to show some of the configuration options for the widget.

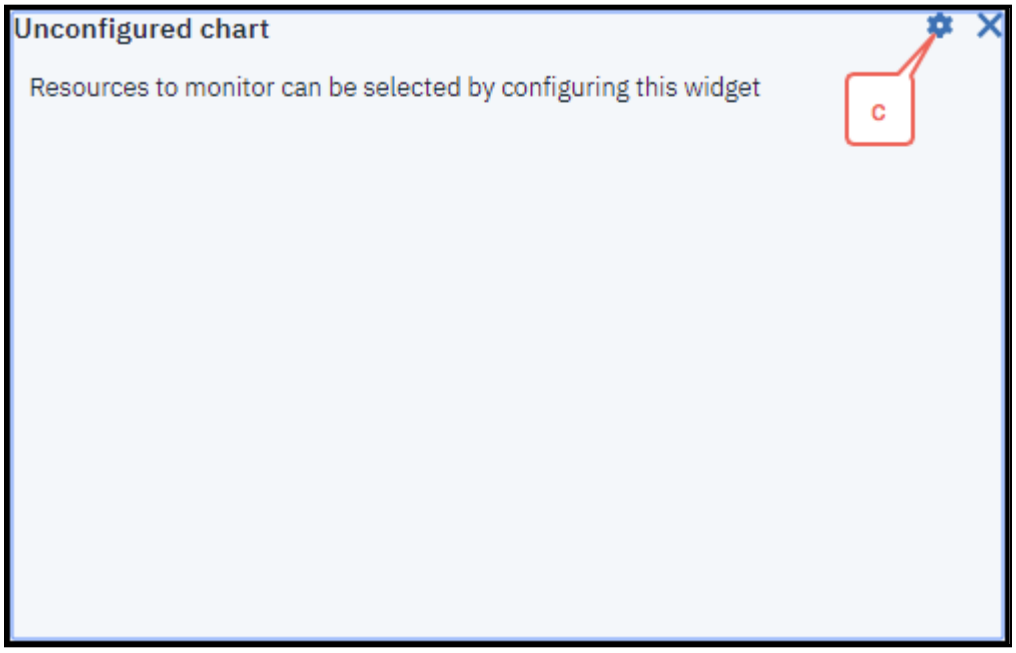
    a. Hover over and then click the title bar of the widget to edit the title.



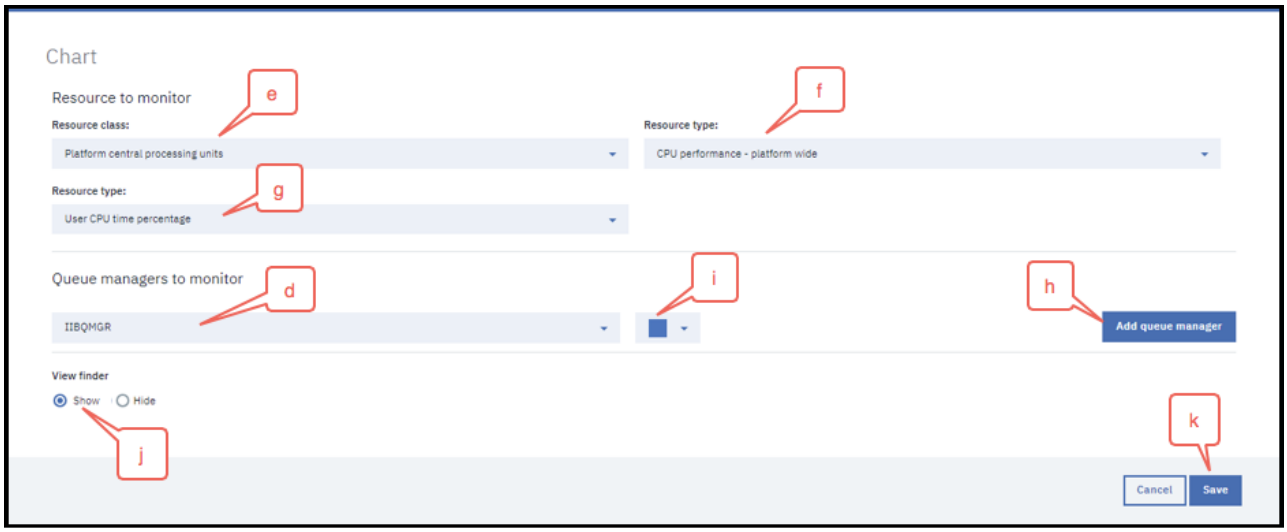
    b. Enter an appropriate title for this widget and then click on the **Rename** button.



c. Click on the gear icon to edit the settings.



A configuration properties window will be displayed.



d. Select the queue manager that you would like to display.

e. Select the desired **Resource class** that you would like to display. Use the drop down icon to see the various classes that are available. Changing the **Resource class** selection will change the options that are available in the **Resource type** entry.

f. Select the desired **Resource type** that you would like to display. Changing the **Resource type** selection will change the options that are available in the **Resource element** entry.

g. Select the desired **Resource element** (Labeled as **Resource type:** under the **Resource class:** entry) that you would like to display.

h. The Chart widget is capable of showing metrics for more than one queue manager. Click on the **Add queue manager** button if you would like to display metrics for additional queue managers.

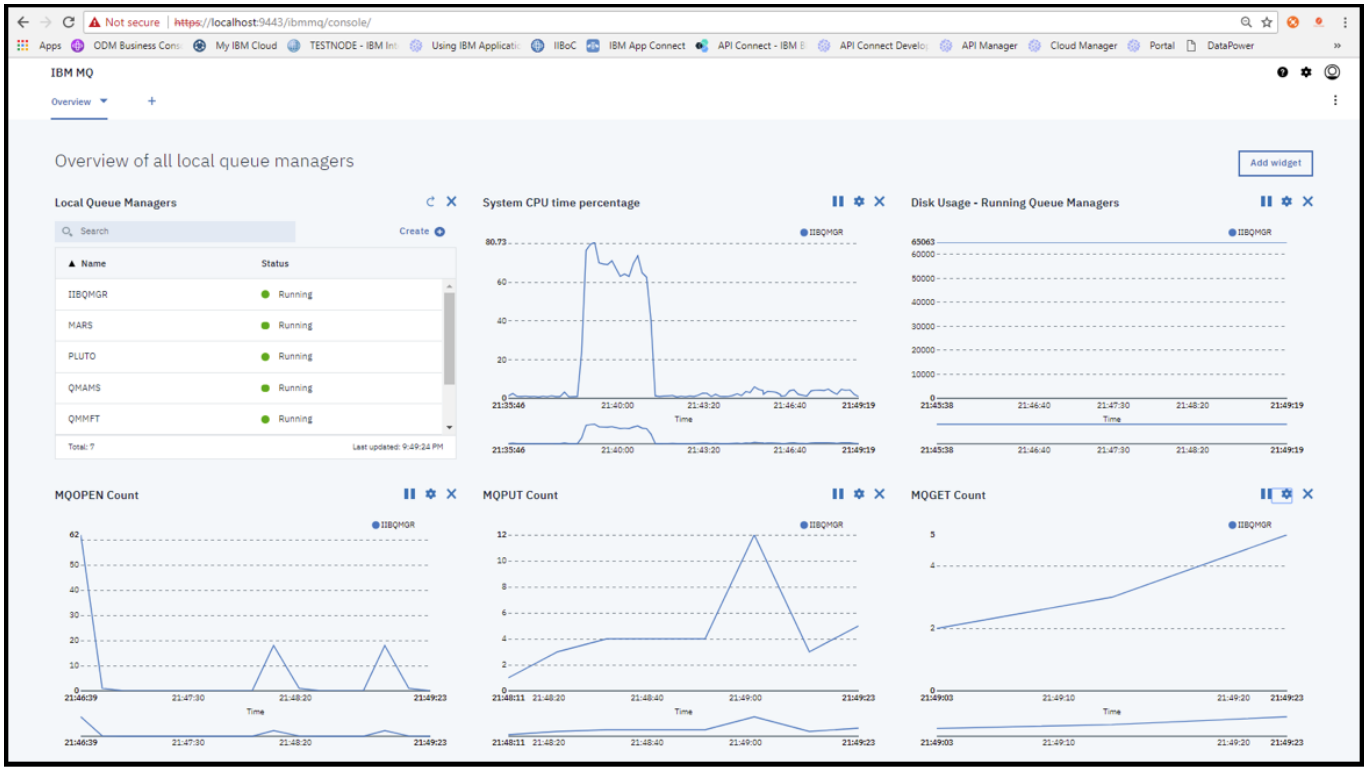
> **Note:**
> Clicking on the **Add queue manager** button will add a queue manager to the collection of queue managers. You will need to select the proper queue manager in the drop down list.

i. In order to identify metrics for a specific queue manager you may set a different color for each queue manager.

j. The **View finder** is a special feature of the widgets that allow the viewer to focus on specific time intervals in the displayed chart. Click on the **Show** radio button in order to review this feature.

k. Click on the **Save** button when you have configured the widget.

10. Repeat the above process to configure the remaining Chart widgets to display various metrics that may be of interest to you. Metrics are collected on an interval basis, so it may take up to a minute to start displaying any data. (Of course, without much activity running against the queue managers you won't see much detail on the charts.)



11. Note that at the bottom of the Chart widgets there is a smaller graph diagram displayed. You can use this smaller graph to zoom into a time interval of interest. For example, if you notice a sudden change in activity in a graph you can click and drag your mouse in the smaller graph to focus on that time interval in the larger graph.

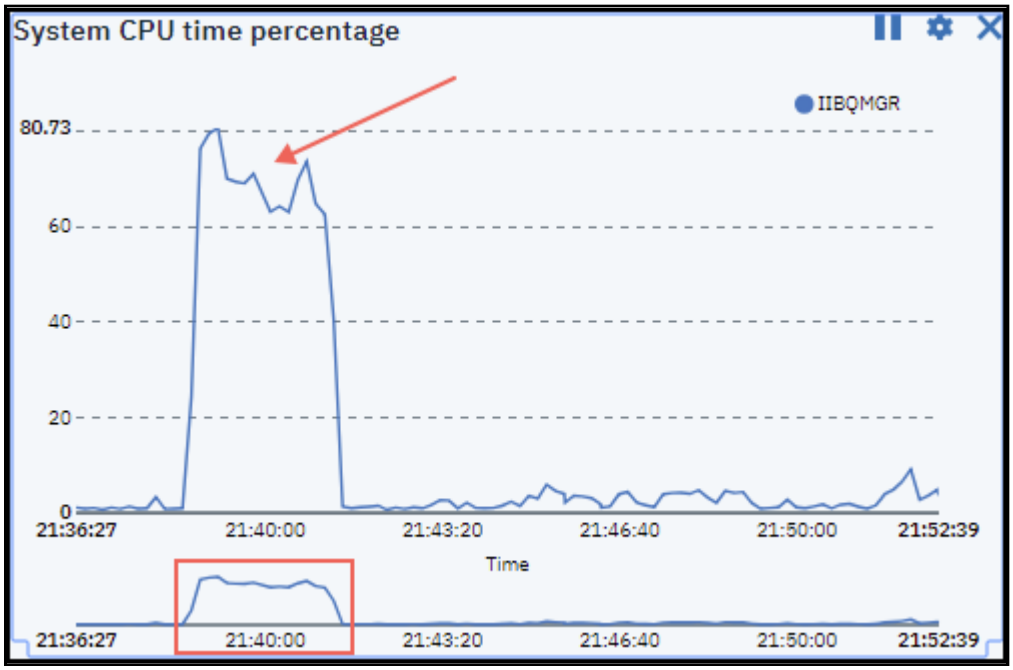
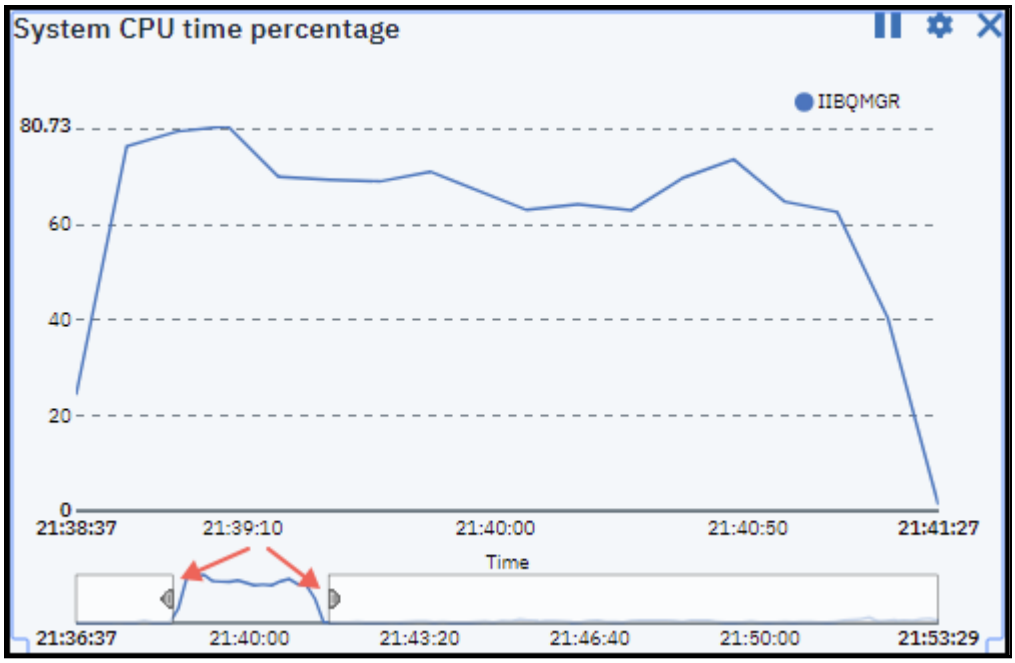
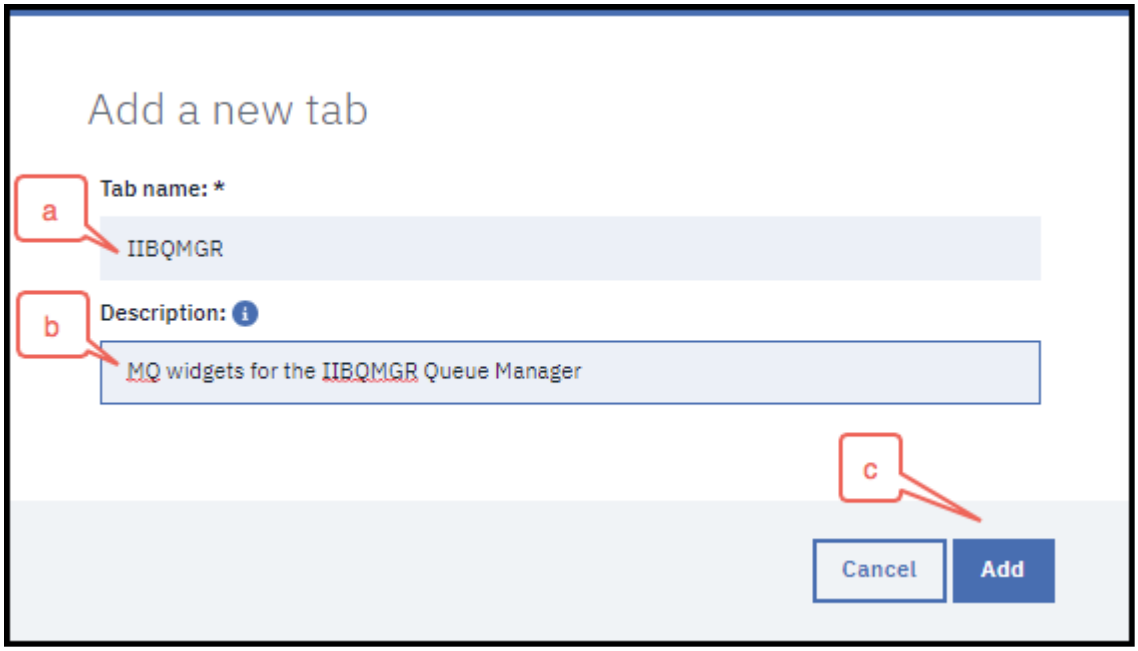


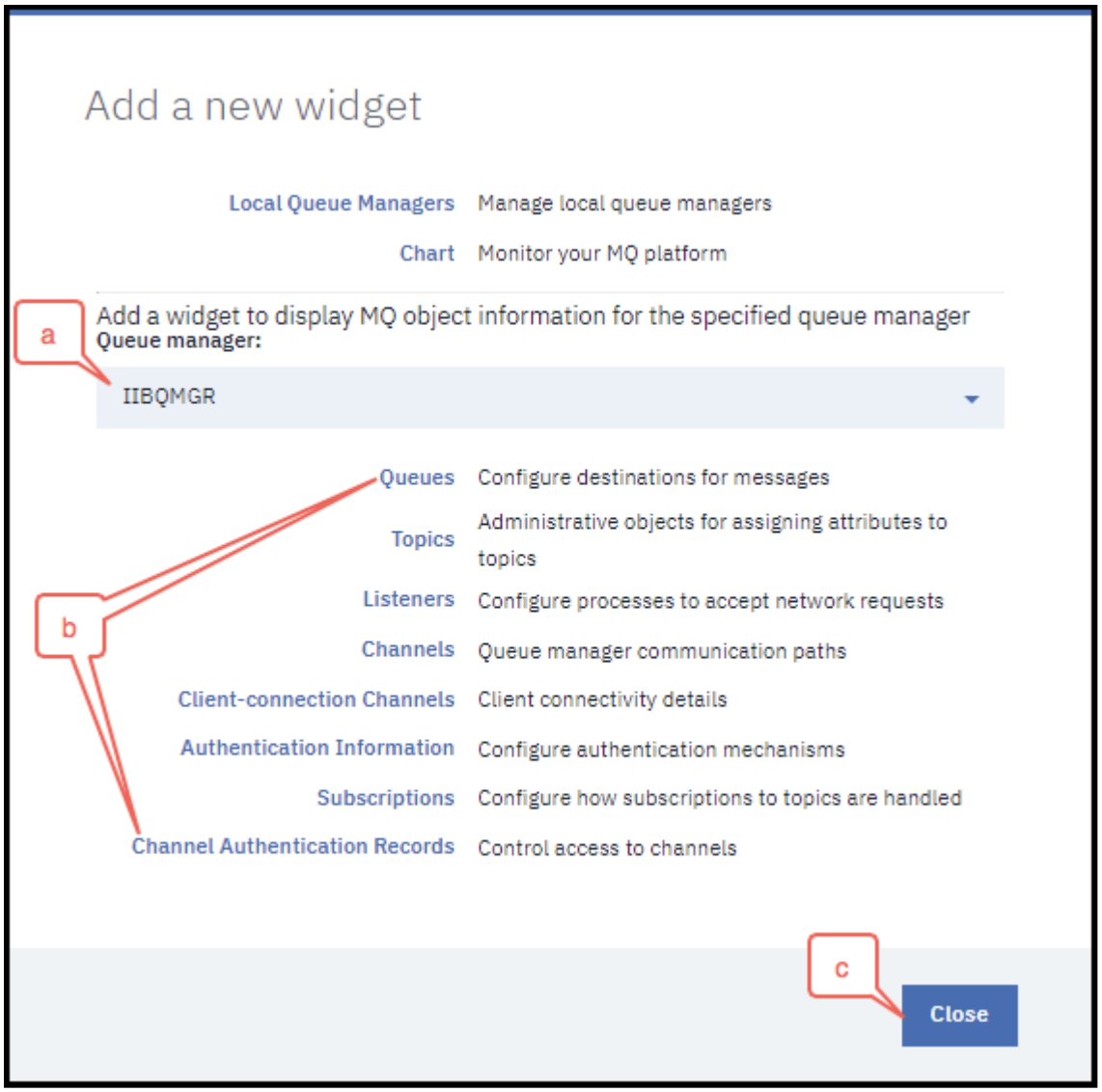
Chart widget after clicking and dragging the smaller chart.



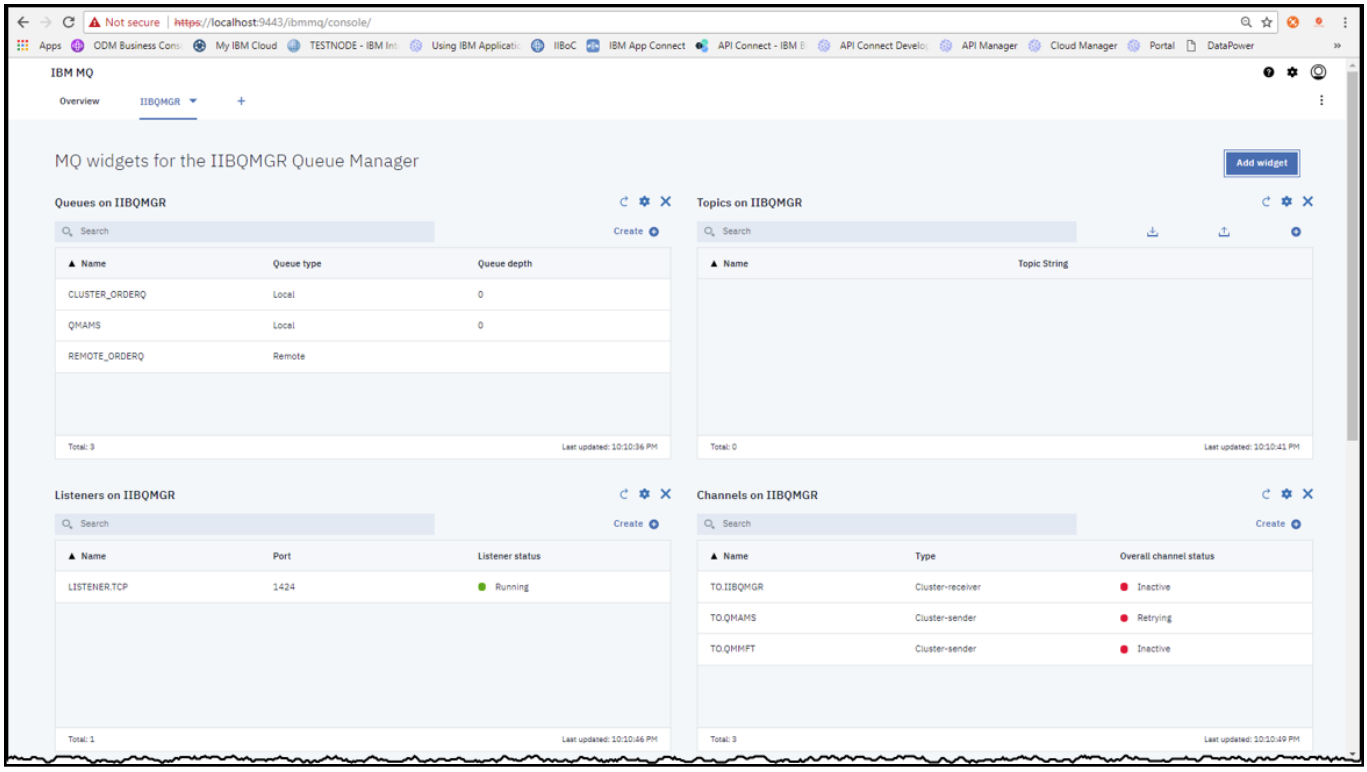
12. Next create an additional tab that will contain several widgets that you may use to administer MQ objects for the **IIBQMGR** queue manager. Click on the **+** icon at right of the existing tab to create an additional tab. Name the tab **IIBQMGR** and provide an appropriate description. Click on the **Add** button when you have finished.



13. Use the **Add widget** button in the upper right side of the workspace to add one of each of the queue manager specific widgets that are available. Ensure you select the **IIBQMGR** queue manager before you add each widget.



14. Note that since you didn't change the column layout of the **IIBQMGR** tab the widgets were laid out in the default, two column format.



15. Each widget includes a common set of controls located in the right side of the title bar. These are:

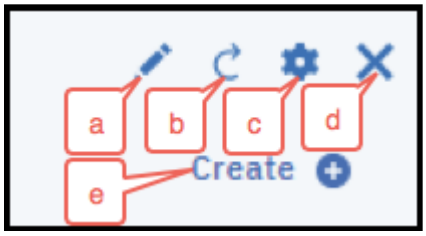
    a. The **Edit description** icon. (Visible only when you hover over the title bar.)

    b. The **Refresh widget** icon.

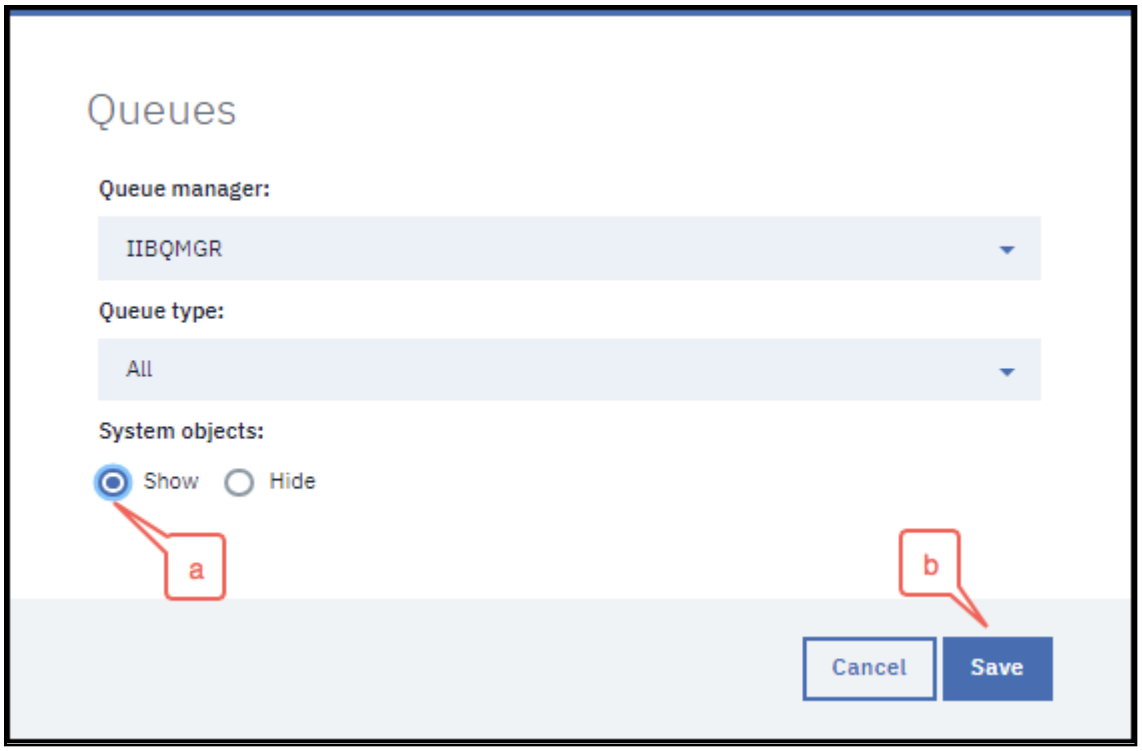
    c. The **Configure widget** icon.

    d. The **Remove widget** icon.

    e. The **Create** icon, which is used to create new MQ objects via the widget.

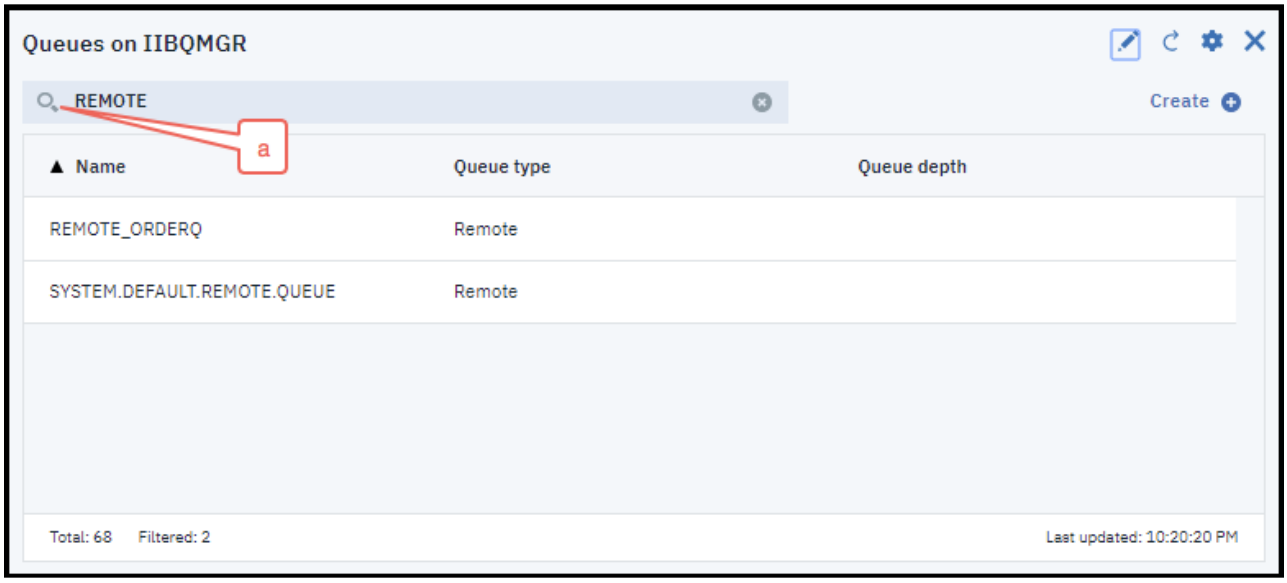


16. Click on the **Configure widget** icon for the **Queues on IIBQMGR** widget to display the properties that may be configured. Click on the **Show** button to enable the widget to display SYSTEM queues and then click on the **Save** button.



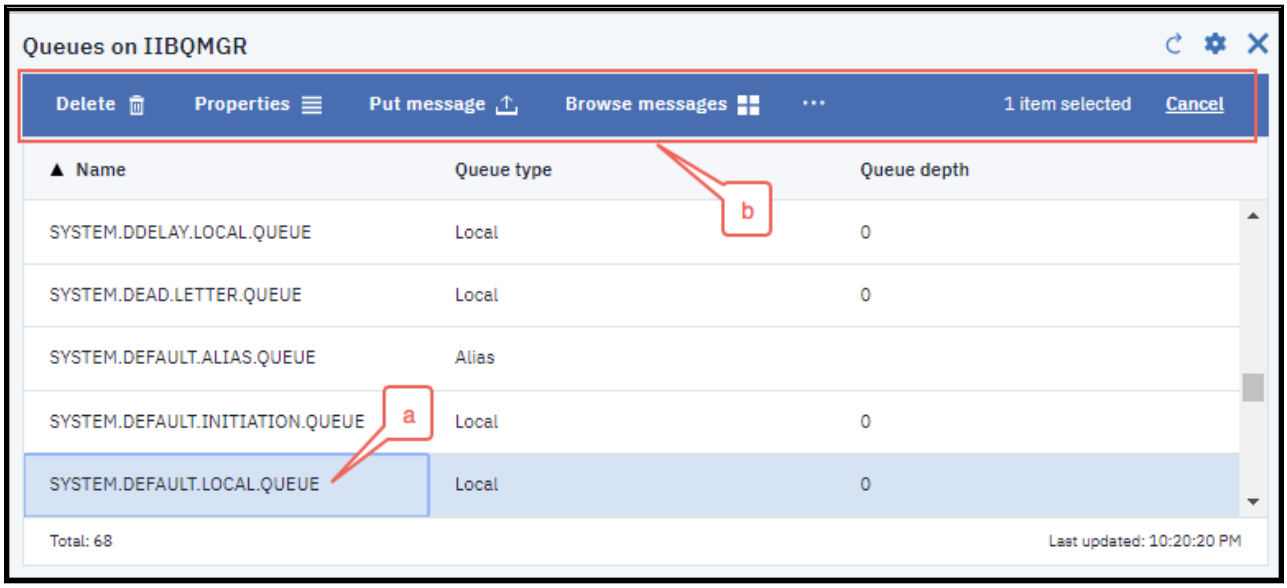
17. Note the following for the **Queues on IIBQMGR** widget:

    a. There is a filter function that enables users to quickly locate object(s) that they would like to work with. TO demonstrate, enter **REMOTE** to only display queues with **REMOTE** in the queue name.



    b. Each widget has a set of action icons that will vary depending on the widget that is in use. For the **Queues on IIBQMGR** widget there are icons that allow authorized users to create new queues, PUT messages on a queue, change properties of a queue, etc. Hover your cursor over each icon to see what function it performs.

c. You will need to select a queue from the list in order to use most of the action icons. Click on the **SYSTEM.DEFAULT.LOCAL.QUEUE** queue to select that queue and then explore the functions of the various icons that are available for a local queue.



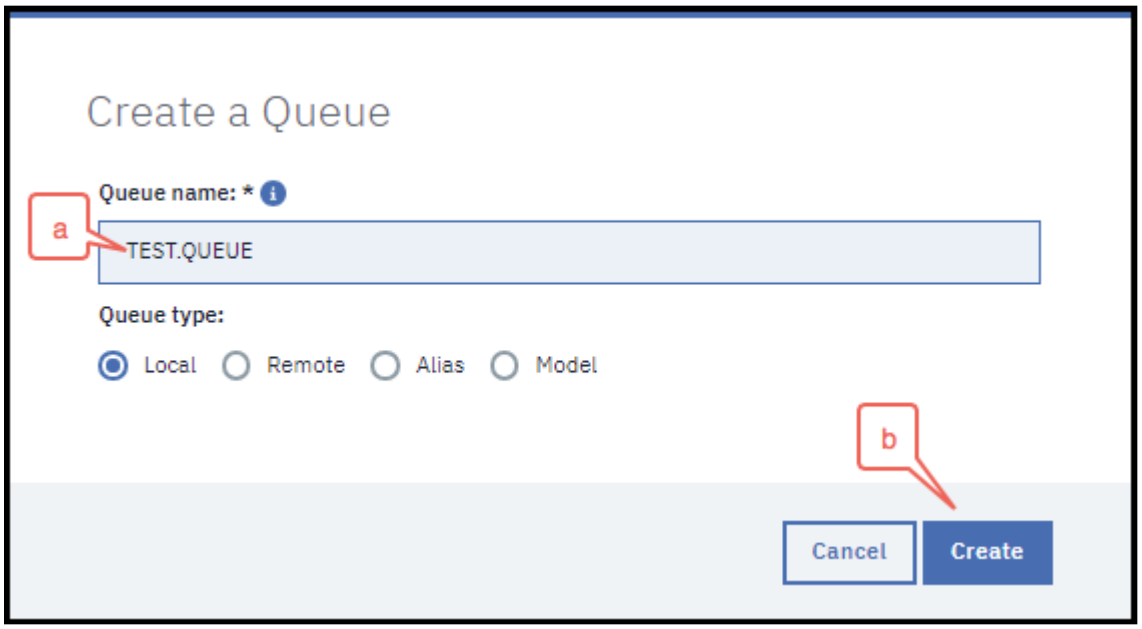
d. Now click on the **Properties** action icon to display a list of properties for that queue. Review the various properties that may be set using this dialog. Click on **Save** or **Close** to return to the dashboard.

18. Next review the remaining widgets on the dashboard to become familiar with the properties and actions that you may access.

> **⚠ Important:**
> You should note that when using the Basic registry, any IBM Web Console users that have been assigned to either the **MQWebAdmin** or **MQWebAdminRO** roles may access MQ objects based on the access that is granted to the userid that the MQWeb Server is running under. IBM MQ Console users that resolve to the **MQWebUser** role will have their access based on what has been defined in the queue manage for the operating system userid that they have logged on

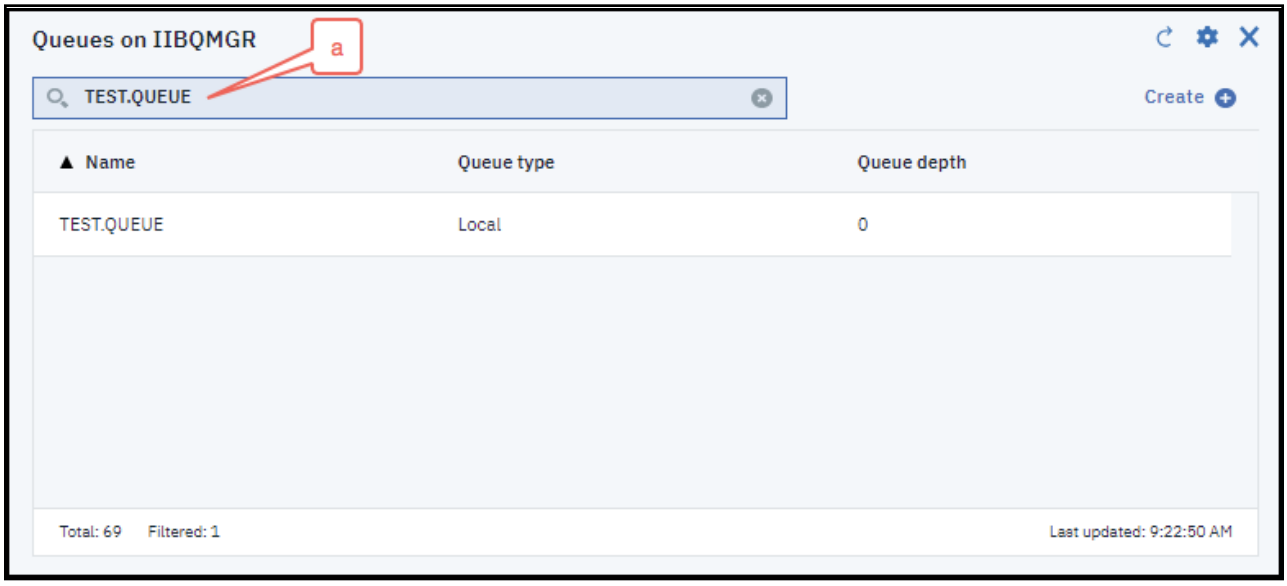
with.

19. To see how security works with the Basic registry use the **Create +** action icon in the **Queues on IIBQMGR** widget to create a new local queue. Name the queue **TEST.QUEUE** and then click on the **Create** button.



20. Now view the authority records for the **TEST.QUEUE** by completing the following steps:

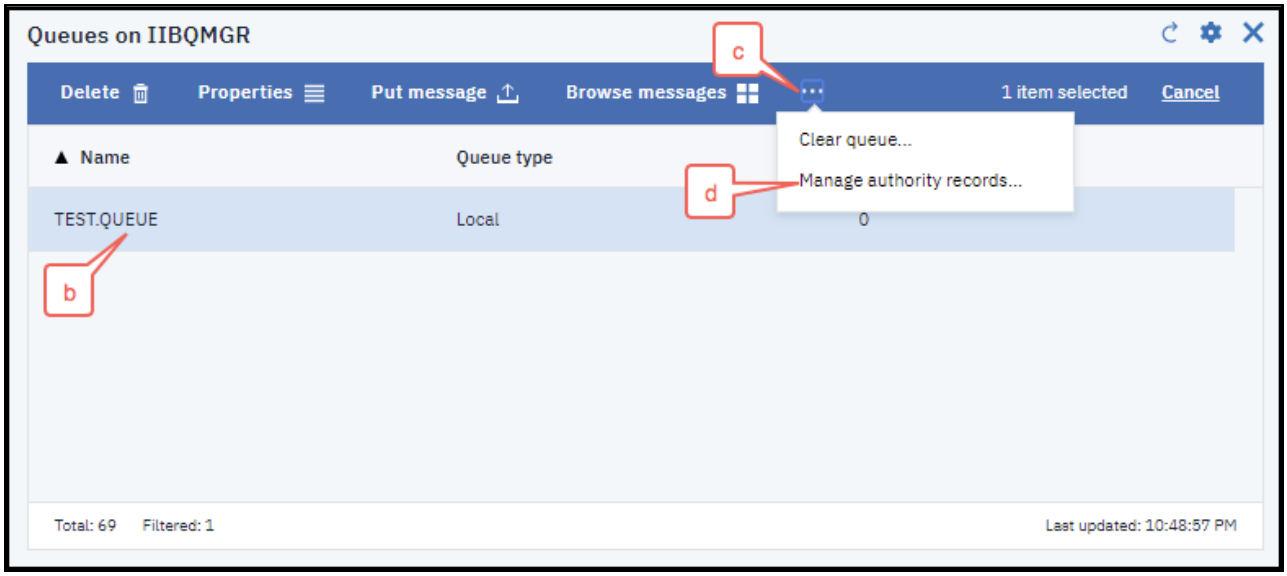
    a. Set your filter to **TEST**.



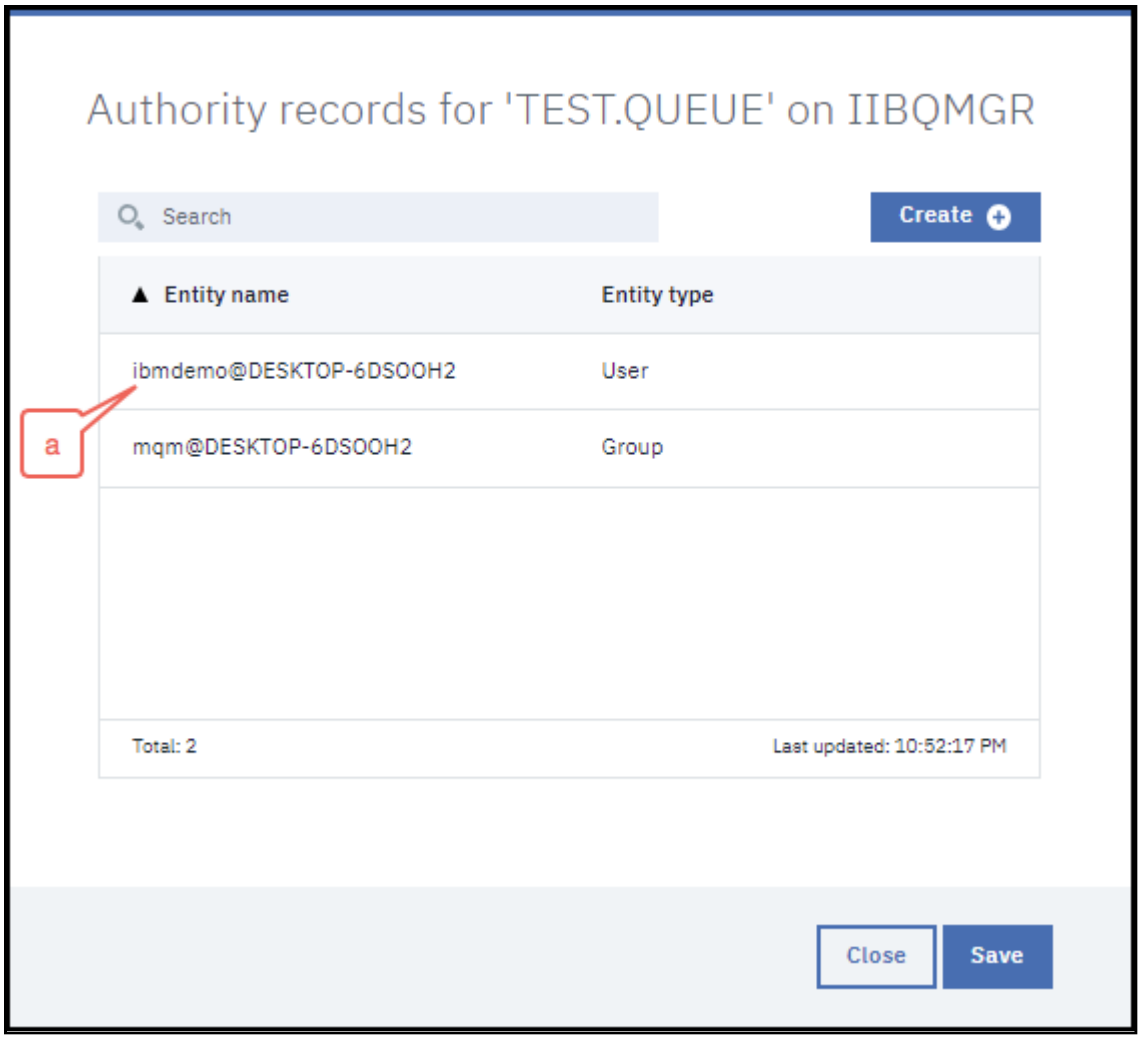
    b. Click on the **TEST.QUEUE** queue to select it.

    c. Click on the **...** menu item.

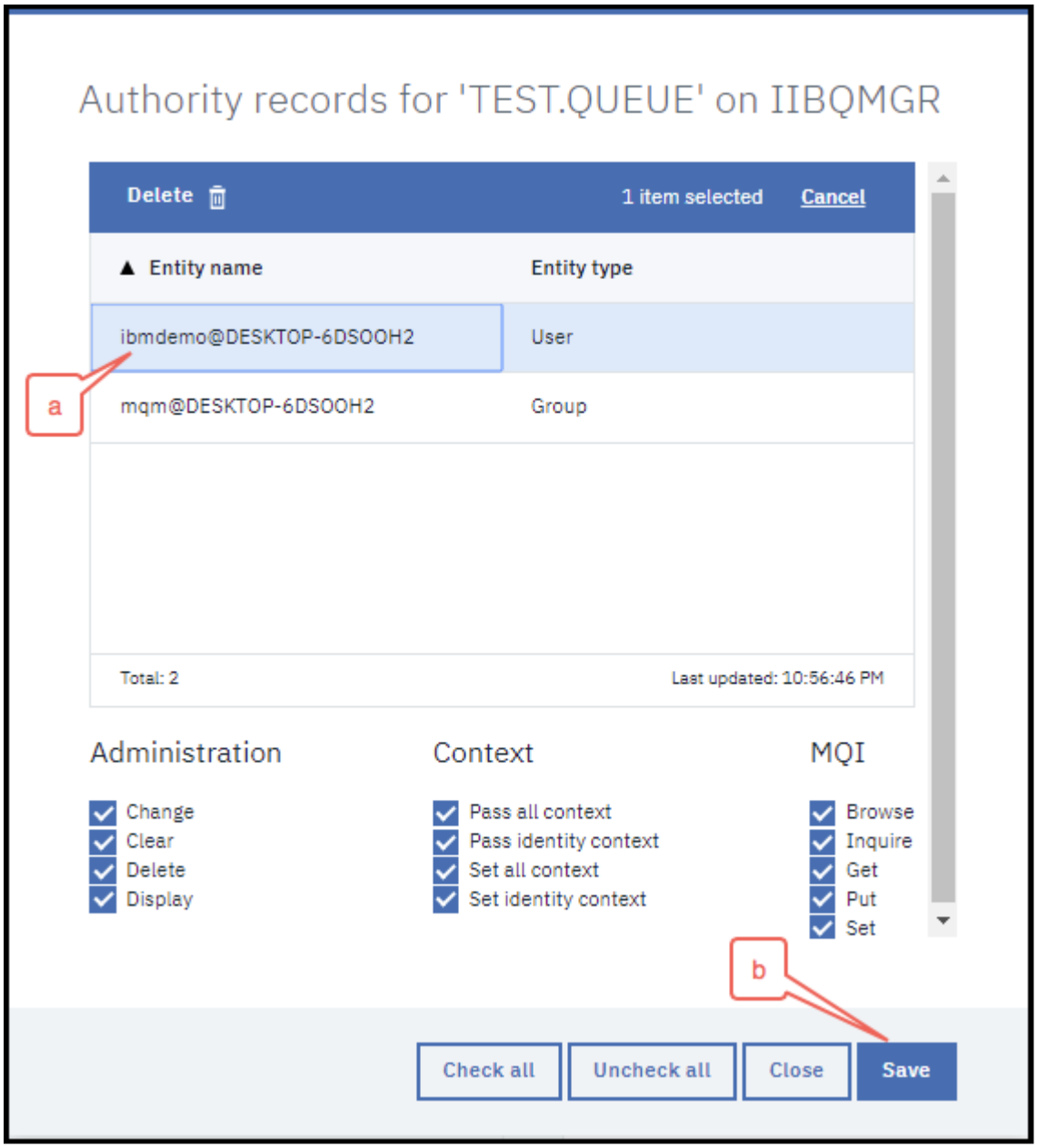
d. Click on the **Manage authority records…** to display the **Authority records for 'TEST.QUEUE' on IIBQMGR** dialog.



21. Pay close attention to the userids that are associated with the authority records for the **TEST.QUEUE** queue. As you may recall you logged into the Windows image using the **ibmdemo** userid and then also started the MQWeb Server while logged in as **ibmdemo**. Later, you created a userid in the Basic registry named **lab5admin** and added it as a member of the **MQWebAdminGroup** *group*. You then logged into the IBM MQ Console with the **lab5admin** userid. Note that the **Entity name** associated with the authority record is the **ibmdemo** userid.



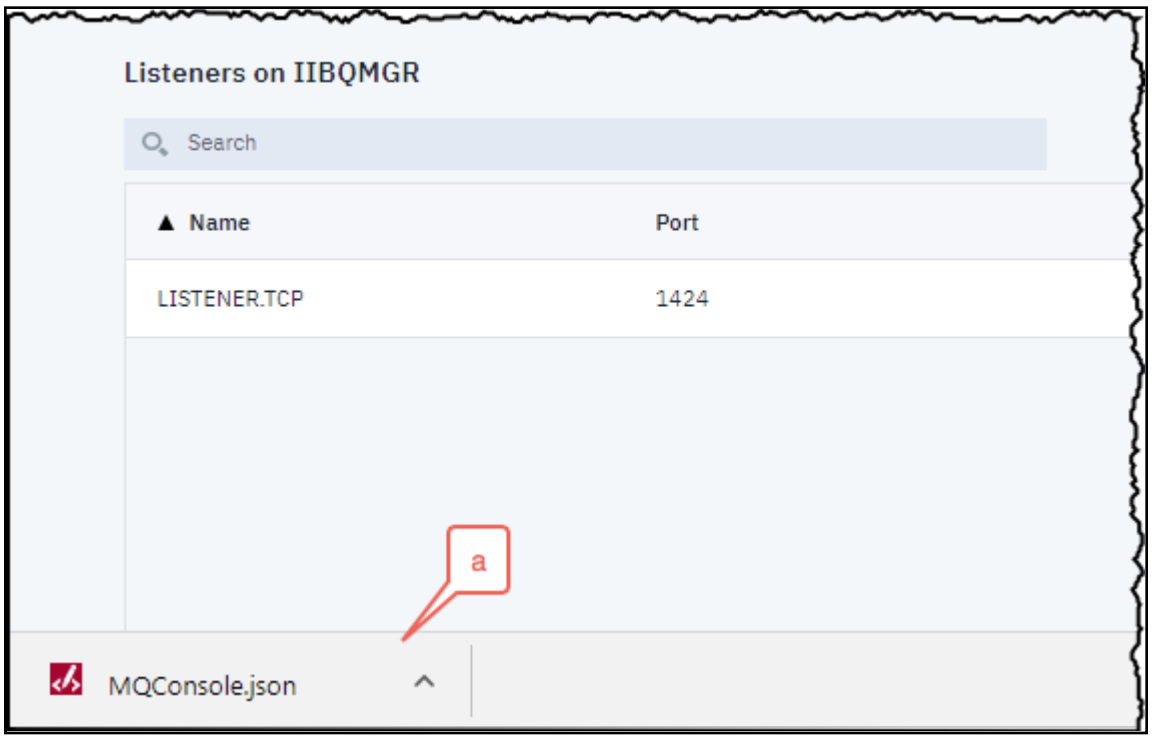
22. Click on the **ibmdemo** userid to see the permissions that are set for that userid. Click on the **Save** or **Close** button when you are finished.



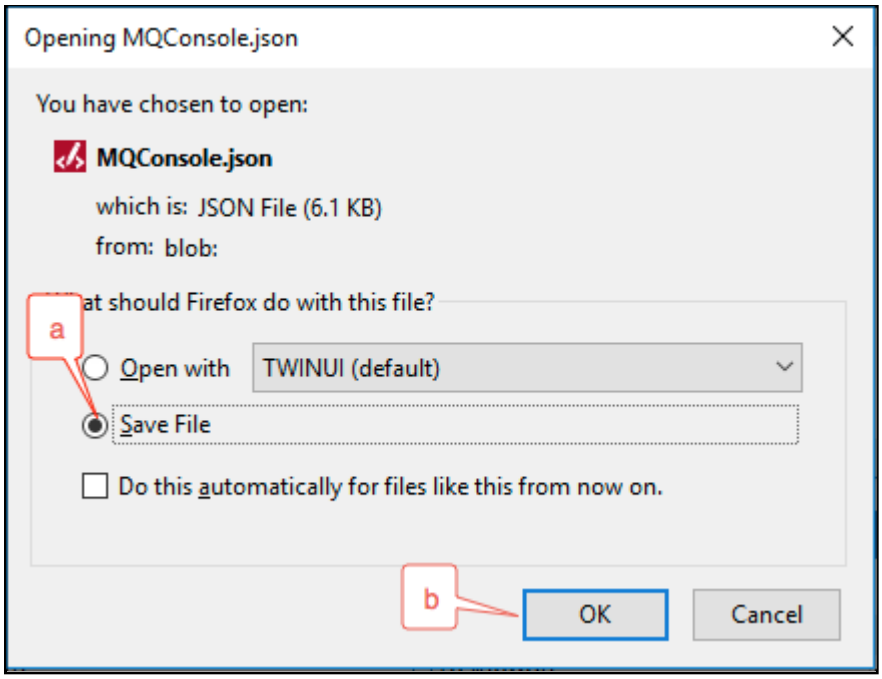
23. In order to better understand how security with the Basic registry works you will re-login to the IBM MQ Console using a different userid and test to see what functions the new userid will have access to. First, you will want to save your dashboard layout so you may use it with the other userid. Click on the control menu at the top right corner of the Dashboard and then click on the **Export dashboard** menu item.

24. The dashboard will be saved as a JSON file in the browser's download directory.

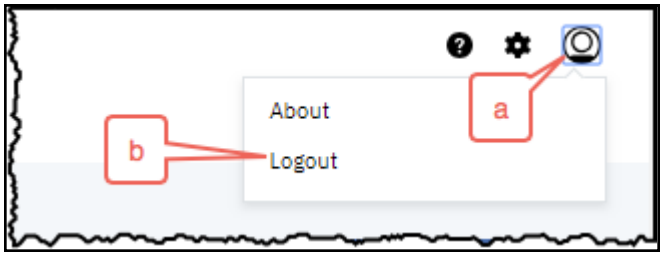
    a. When you are using the Chrome browser the downloaded file will appear in the lower left corner of the browser:



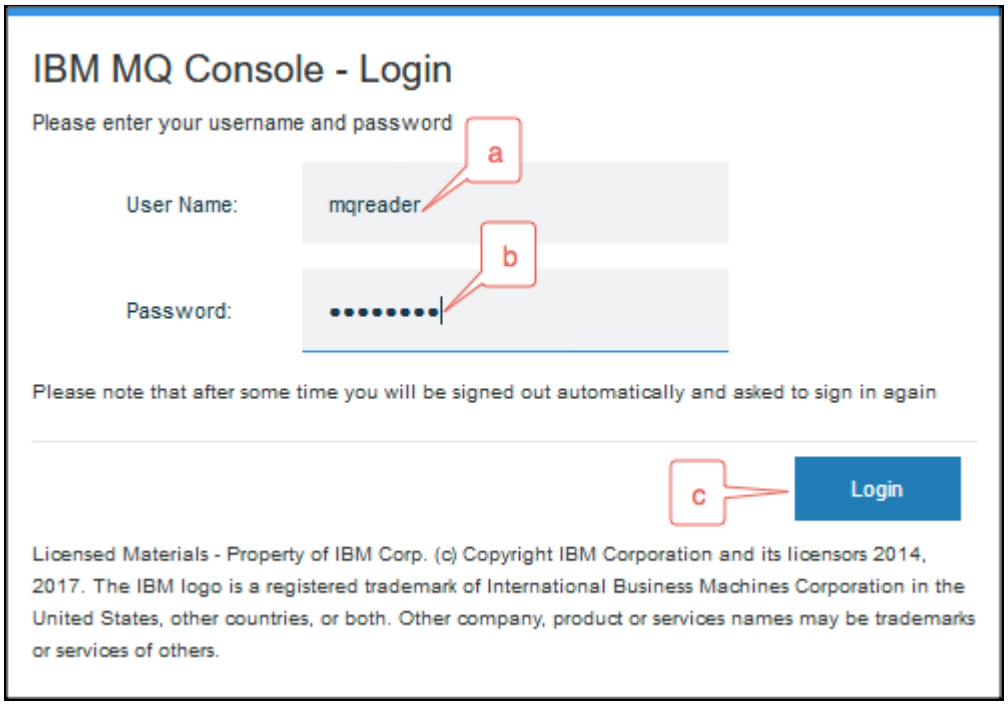
    b. When you are using Firefox a dialog box will appear. Select the **Save File** option and then click on the **OK** button:



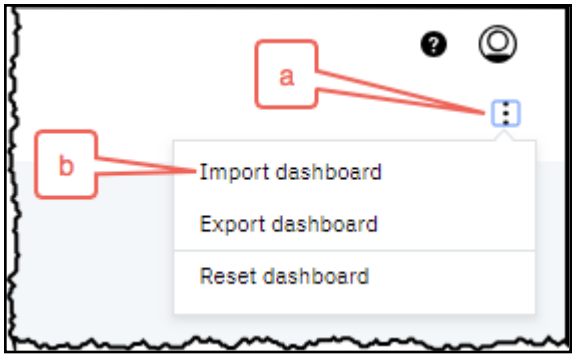
25. Next click profile icon just above the control menu icon and then click on the **Logout** menu item to return to the IBM MQ Console login page.



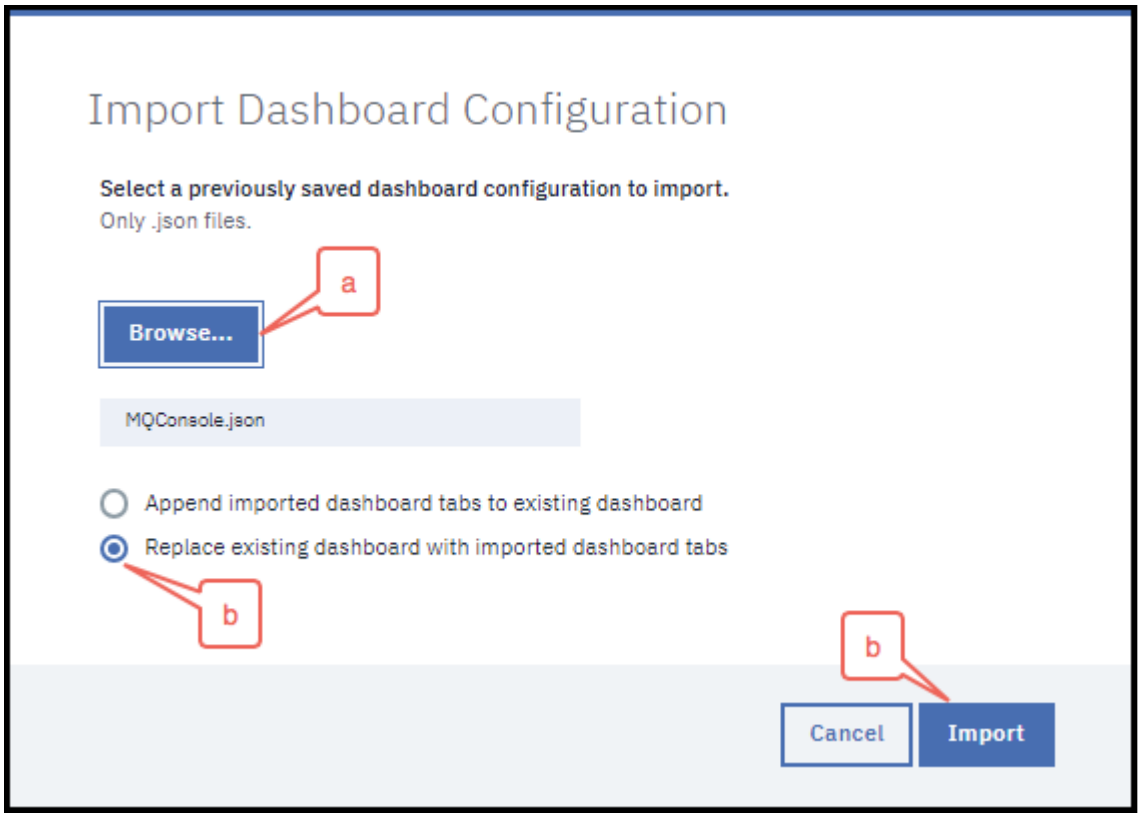
26. Login to the MQ Console using a userid of **mqreader** and a password of **mqreader**.



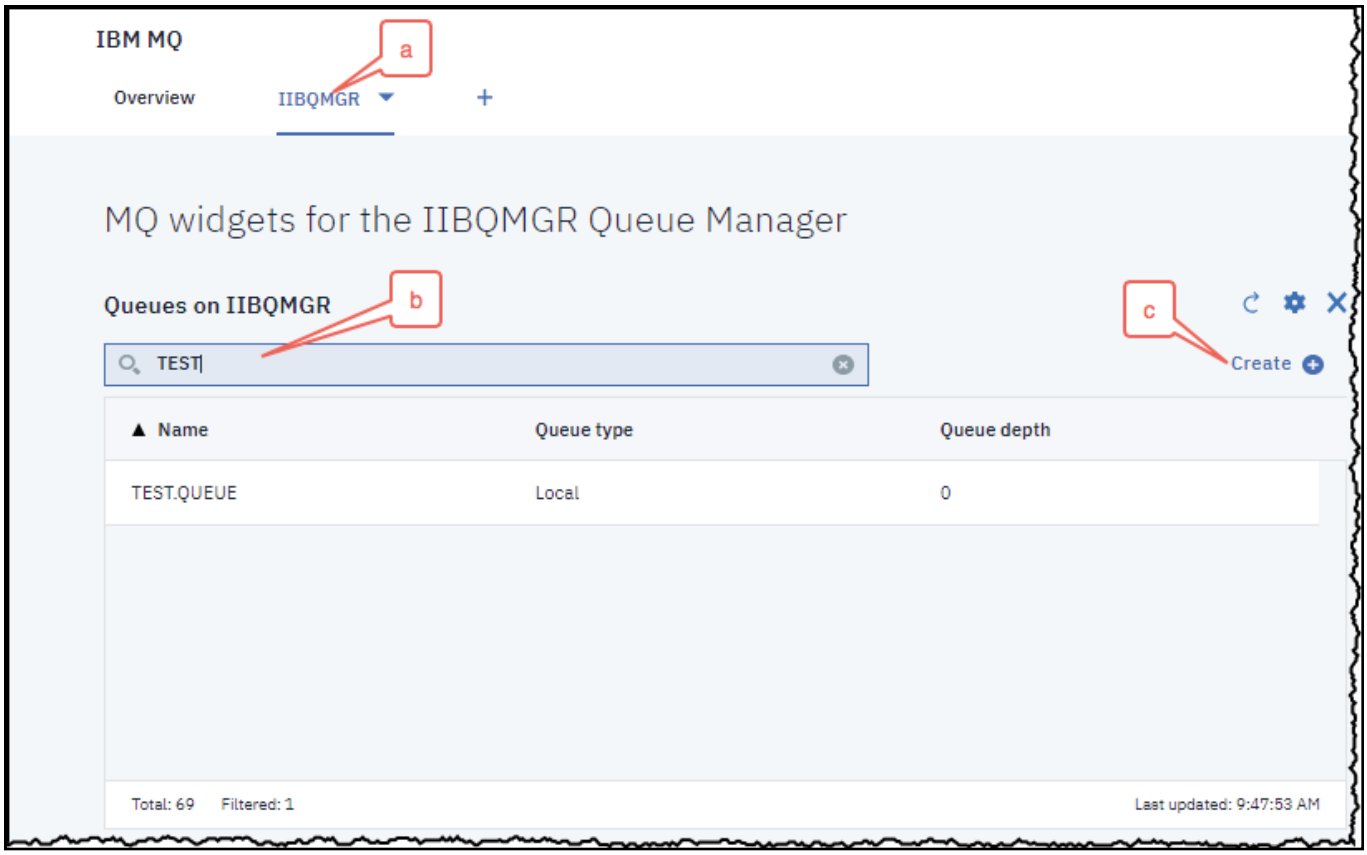
27. Since the **mqreader** user has not customized the IBM MQ Console you will see it has the default layout. Import the dashboard that you had saved earlier by clicking on the control menu icon and then selecting the **Import dashboard** menu item.



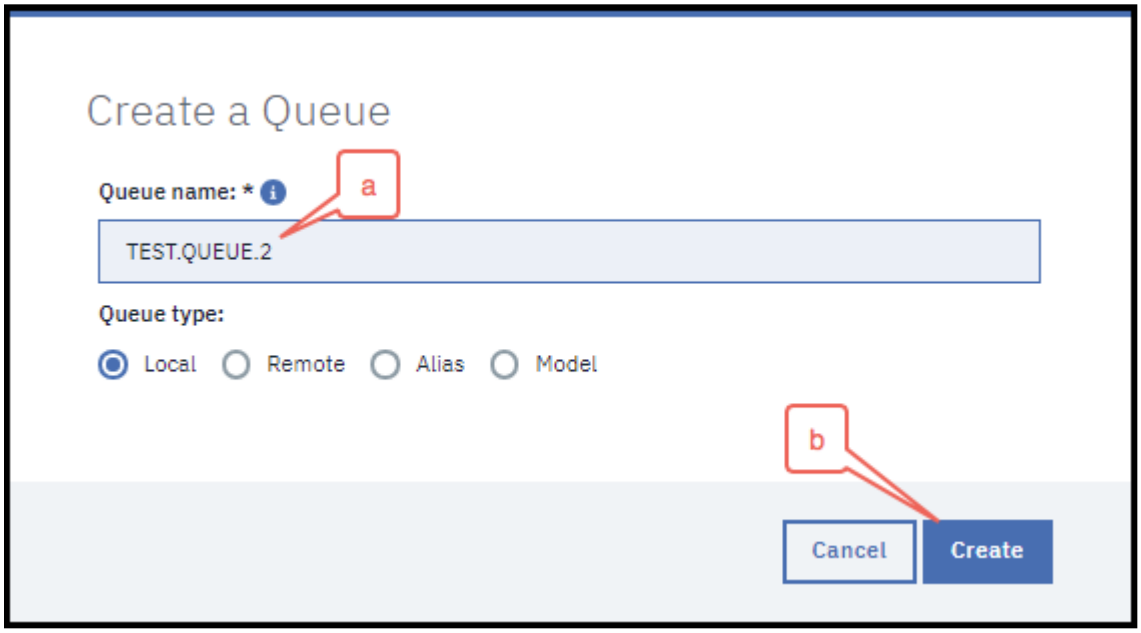
28. Use the **Browse…** button to select the exported dashboard, select the **Replace existing dashboard…** option and then click on the **Import** button.



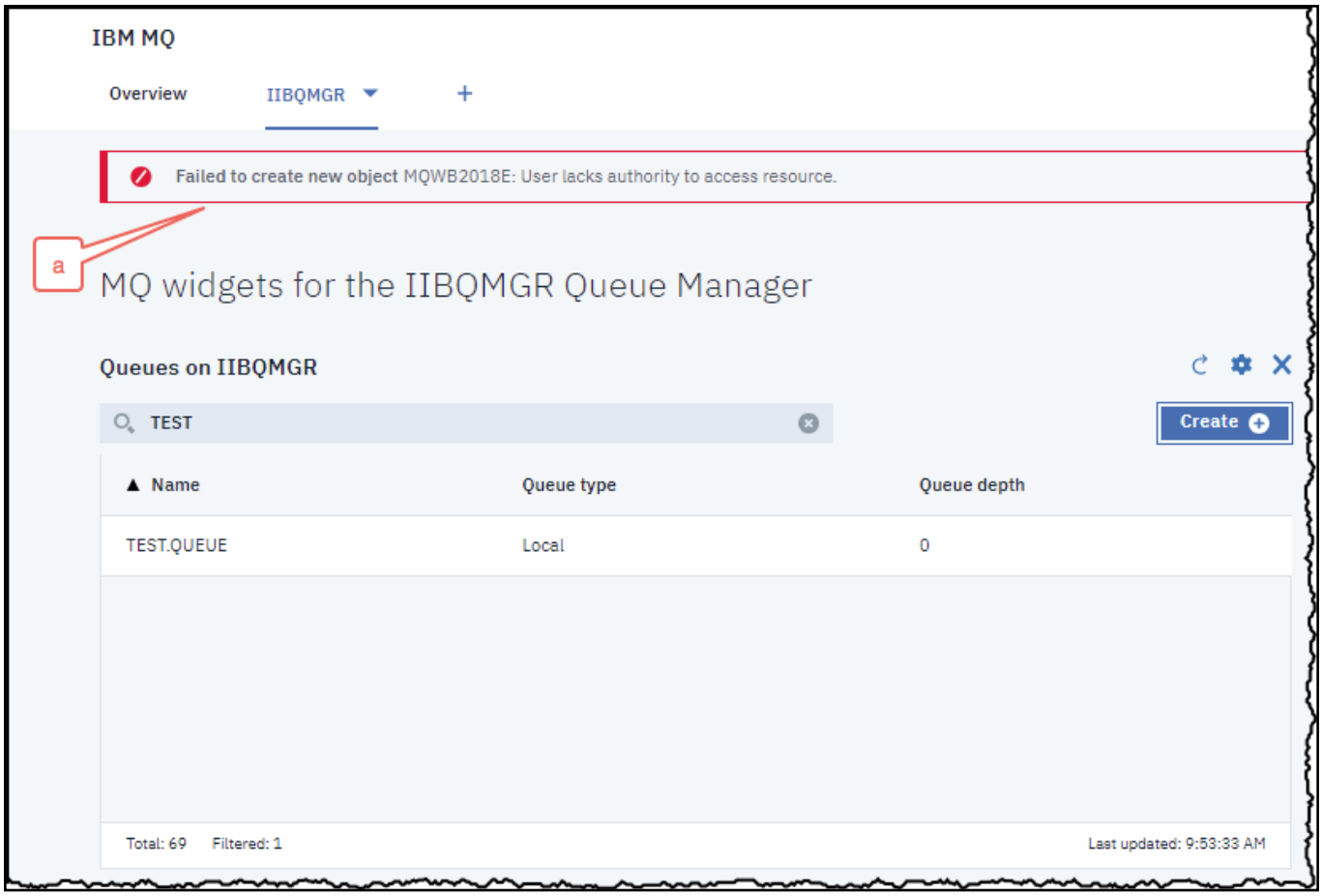
29. Click on the **IIBQMGR** tab and set the filter to **TEST**. Next click on the **Create +** icon in the action toolbar to attempt create a new queue as the **mqreader** user.



30. Name the new queue **TEST.QUEUE.2** and then click on the **Create** button.



31. Note the error message that is displayed at the top of the dashboard. Since the **mqreader** userid does not have authority to work with MQ objects **mqreader** can only display them.



32. Explore additional actions such as attempting to PUT a message to the **TEST.QUEUE** queue using either the **mqreader** or the **lab5admin** userids.
33. Logout of the IBM MQ Console when you have finished.

Summary of using the IBM MQ Console

You have now explored a number of features of the IBM MQ Console. Some of these features include:

- Configuring the MQWeb Server and starting it.
- Creating and customizing a dashboard.
- Exporting and importing a dashboard.
- Security options when using the Basic registry.

Next you will review the new functions that may be accessed via the IBM MQ REST API interfaces.

Using the IBM MQ REST API Interfaces

IBM MQ Version 9 introduced new capabilities in the form of REST API interfaces to enable administration

of queue managers as well as to support messaging based applications. The REST API interfaces are being enhanced on an ongoing basis, so you should refer to the IBM MQ Version 9 Knowledge Center for the most up to date information on using the REST APIs.

In this section of the lab you will discover how to use *administrative* APIs to query information about a queue manager and create and subsequently delete a local queue. Next, you will use *messaging* APIs to PUT and GET messages from the **TEST.QUEUE** queue that you had created previously.

As mentioned earlier, security for the IBM MQ REST APIs is configured in the **mqwebuser.xml** file. For this lab you will use the Basic registry and use HTTP Basic Authentication. When using the REST APIs you must first submit a “Login” request using Basic Authentication.

> **Note:**
> MQ version 9.0.4 and earlier required that the user extract a **Cross-Site Request Forgery** (CSRF) token from a cookie that is returned from the Login request for use with subsequent REST API calls. MQ version 9.0.5 has changed the method of protecting from CSRF attacks. Thus, extracting a CSRF token from the cookie returned from the Login request is no longer needed.

In order to enable CSRF protection with the MQ REST API calls you will need to include an HTTP header named **ibm-mq-rest-csrf-token** for each HTTP POST, PATCH and DELETE request. The content of the header is not significant. It is the presence of the header that is required.

> **Note:**
> Additional options for securing the REST APIs include using LTPA tokens or client certificates. Refer to the IBM MQ Version 9 Knowledge Center for further information.

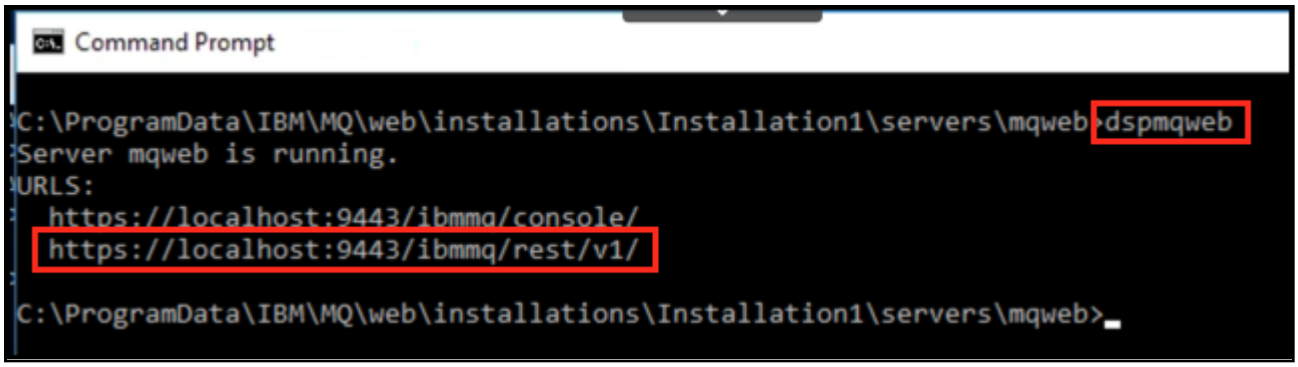
Fundamentals of Using IBM MQ REST API Interfaces

- CORS

  By default, you cannot access the REST API from web resources that are not hosted on the same domain as the REST API. That is, cross-origin requests are not enabled. You can configure the **mqwebuser.xml** file to support Cross Origin Resource Sharing (CORS), which will then allow cross-origin requests from specified URLs.

- Endpoint URL construction

  > **Note:**
  > If the host or port is changed from the default, or if HTTP is enabled, you can determine the URL by opening a command prompt and entering the **dspmqweb** command.



The default URL prefix to access administrative REST APIs is:

https://localhost:9443/ibmmq/rest/v1/admin

The default URL prefix to access messaging REST APIs is:

https://localhost:9443/ibmmq/rest/v1/messaging

- Formatting REST requests

  In order to use the REST APIs to perform an action on an object, you need to construct a URL to represent that object. Each URL starts with one of the prefixes listed above, and the rest of the URL describes the object, or set of objects, known as a resource. Resources may include:

  - installation
  - mqsc
  - qmgr
  - channel
  - queue
  - subscription

  For example, to interact with queue managers add **/qmgr** to the prefix URL:

  https://localhost:9443/ibmmq/rest/v1/admin/qmgr

  The action that is to be performed on the resource defines whether the URL needs query parameters or not. It also defines the HTTP method that is used, and whether additional information is sent to the URL, or returned from it, in JSON form. The additional information might form part of the HTTP request, or be returned as part of the HTTP response.

  After you construct the URL, and create an optional JSON payload for sending in the HTTP request, you send the HTTP request to the MQWeb Server. During development you can test your requests by using tools such as **cURL** or **Postman**. When you're ready you can use an appropriate programming language that supports the HTTP/HTTPS protocols.

- Timestamps

When date and time information is returned by the administrative REST API, it is returned in Coordinated Universal Time (UTC), and in a set format. The date and time is returned in the following time stamp format:

YYYY-MM-DDTHH:mm:ss:sssZ

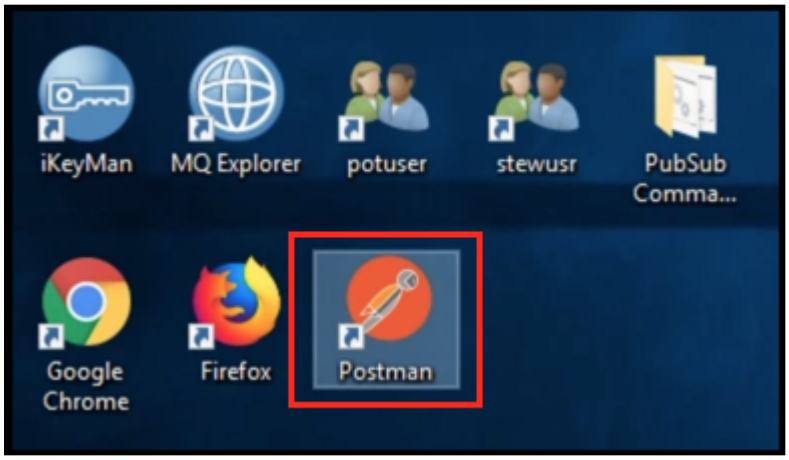
- Error handling

  The REST API reports errors by returning an appropriate HTTP response code, for example 404 (Not Found), and a JSON response. Any HTTP response code that *is not* in the range 200 - 299 is considered an error. Note that the error response may contain nested JSON objects.

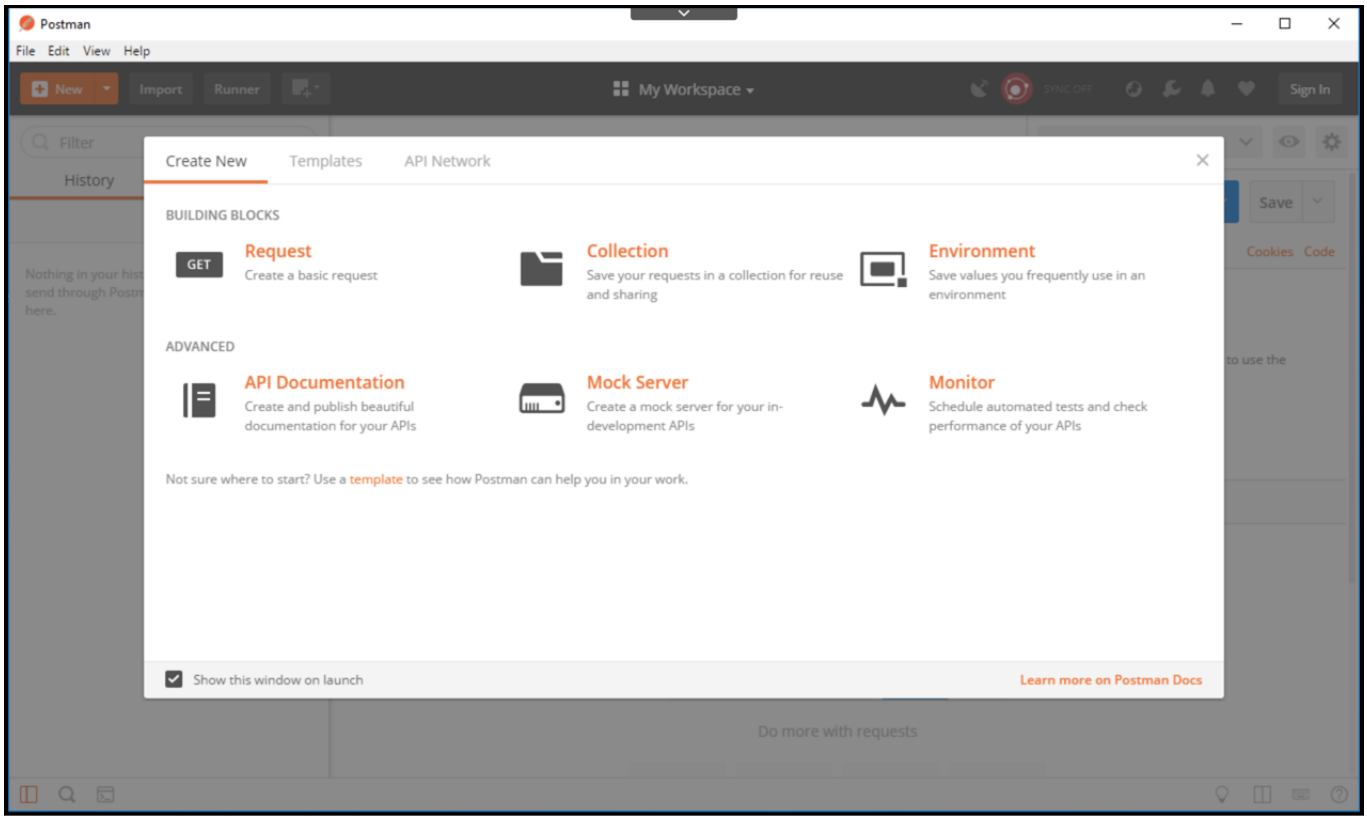
Lab Preparation

As mentioned above, it is necessary to use an HTTP testing tool in order to test out API development. For this lab it will be easiest to use the **Postman** tool from Google. Complete the following steps in order to launch Postman and prepare it for use:

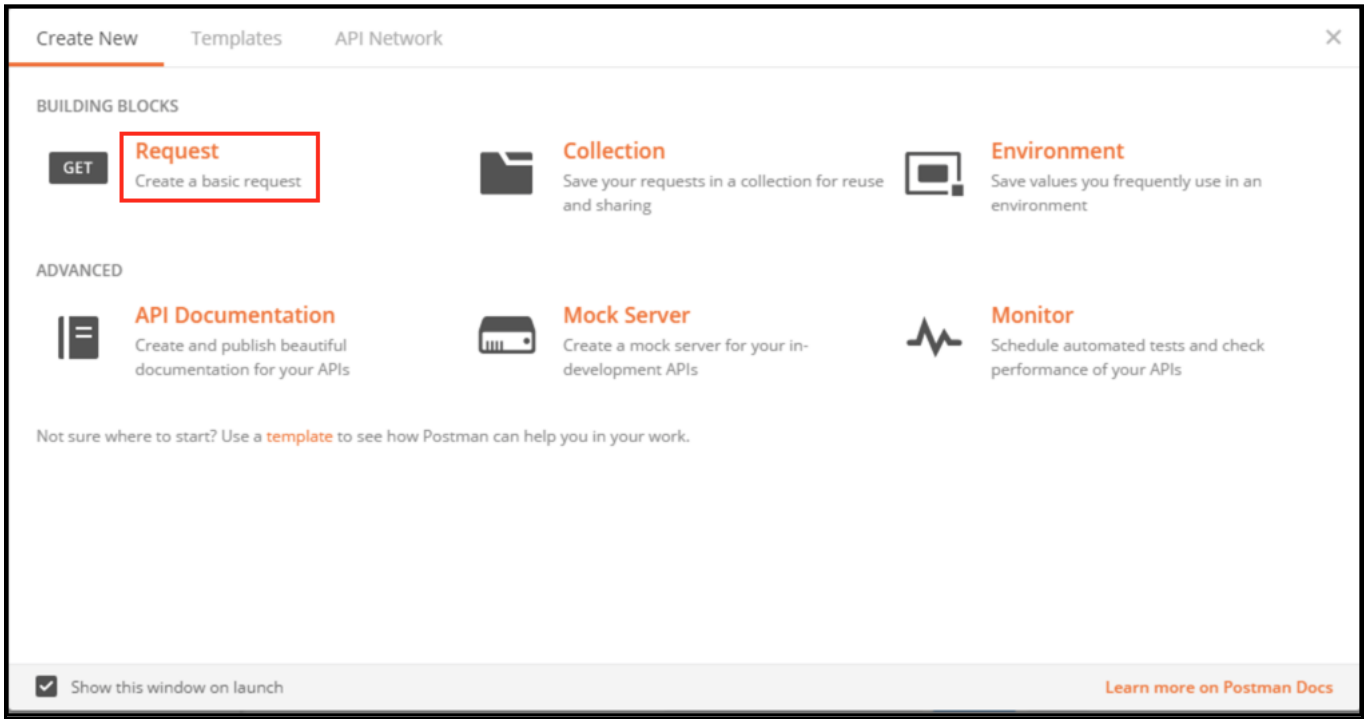
1. Double-click the **Postman** icon on the desktop.



2. Click on the **Postman** icon. The Postman tool will launch and appear in a separate window.

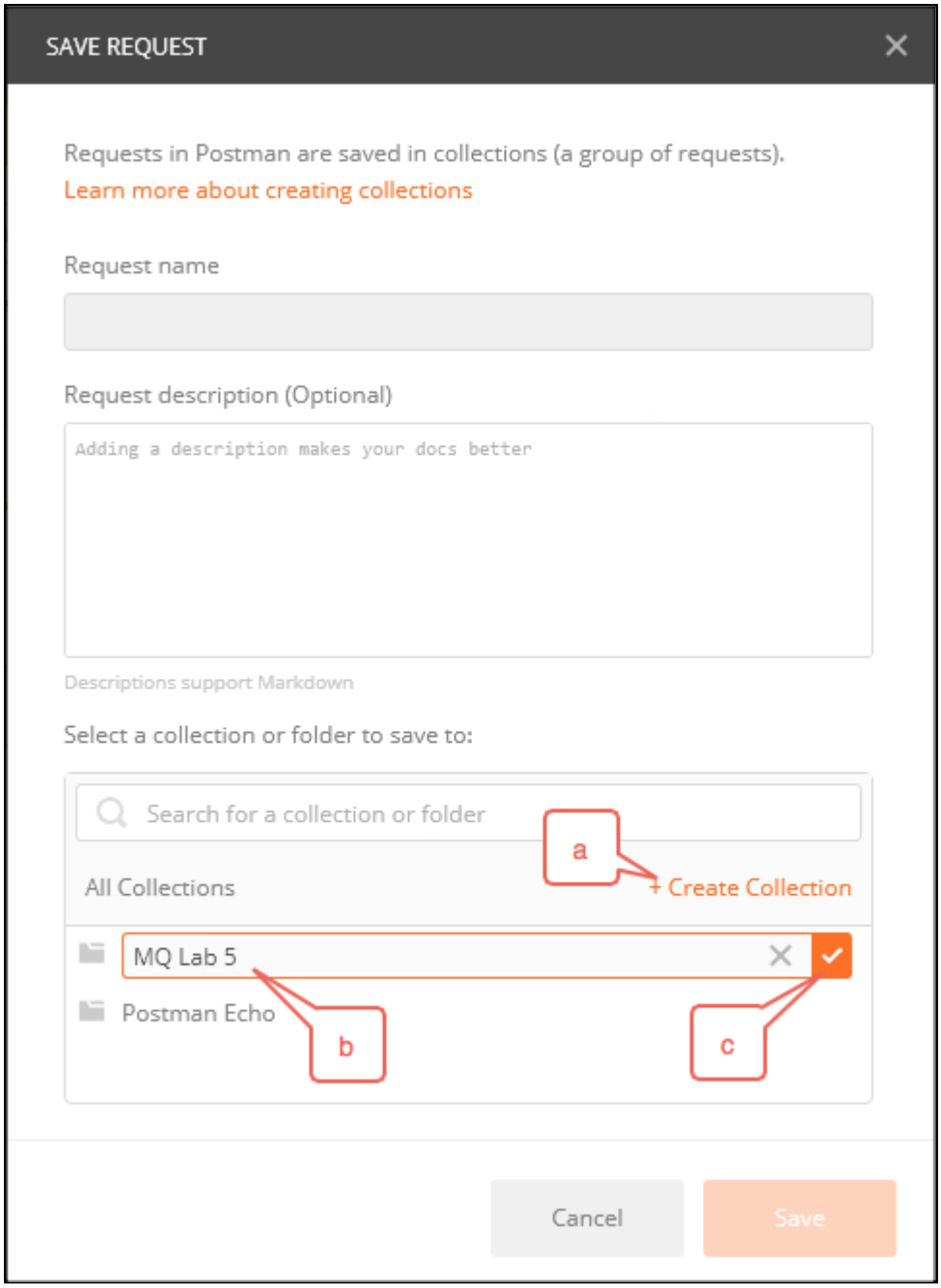


3. You will be presented with a “Getting Started” wizard. Click on the **Request** link to proceed.



4. The **SAVE REQUEST** dialog will appear. Postman allows you to save HTTP requests into a “Collection”. This feature enables you to define and save multiple requests and execute them

repeatedly for testing purposes. Click on the **+ Create Collection** link, enter an appropriate name and then click on the orange checkmark to create your collection.



5. The MQWeb server uses self-signed certificates by default. Complete the following steps to configure Postman to accept self-signed certificates:

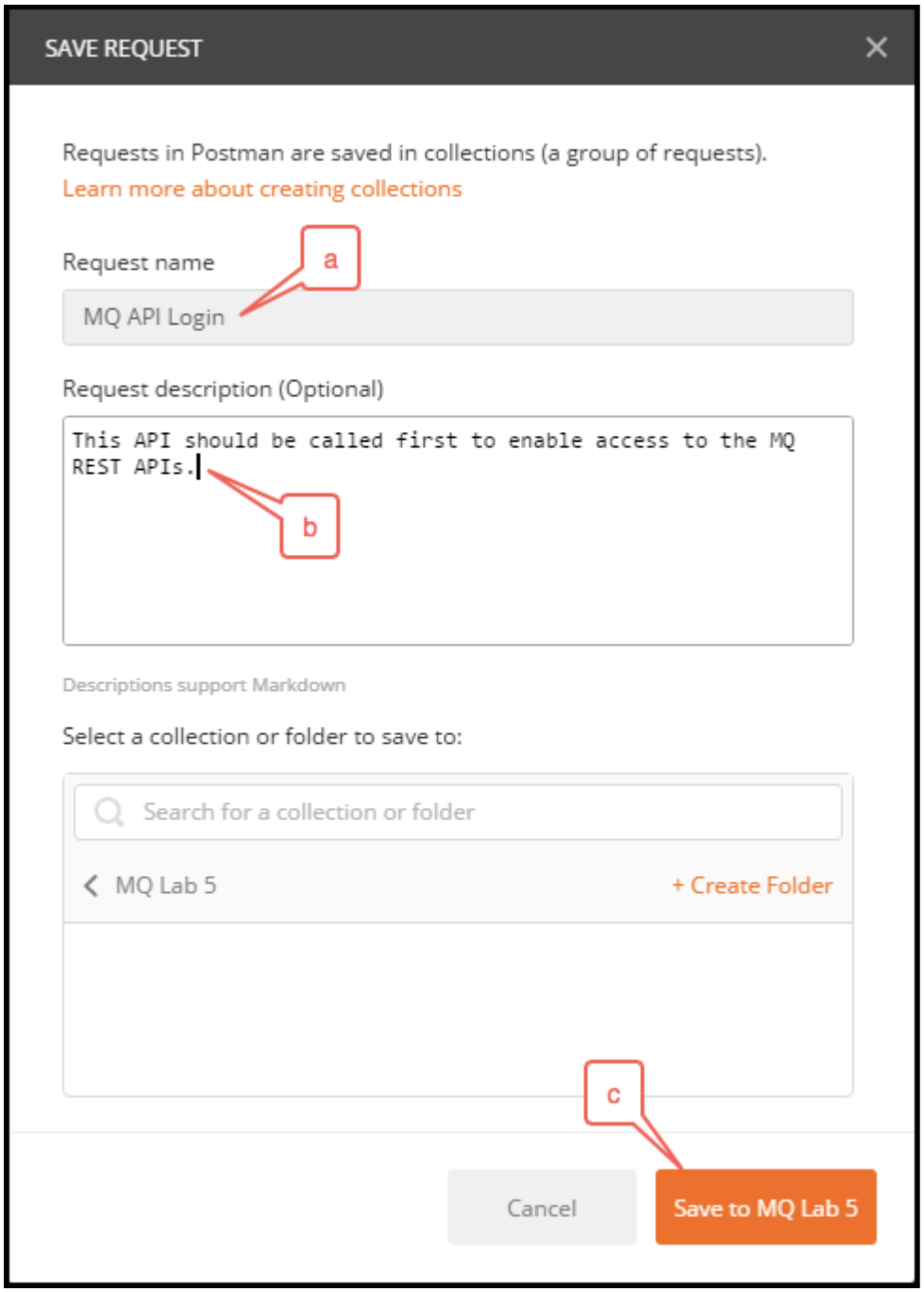
    a. Click on the **File** and **Settings** menu items to open the **Settings** page.

```
![](./images/pots/mq/lab5/mqweb_PostmanFileSettings.png)
```

a. Switch off the **SSL certificate verification** option and then close the window.

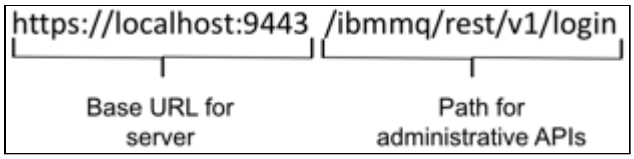
```
![](./images/pots/mq/lab5/mqweb_PostmanSslSetting.png)
```

6. Since you will be testing several different APIs our first action will be to create a login request. Enter a name and description for this login request and ensure that you select your collection. Click on the **SAVE** button to continue.



7. You will need to determine the appropriate URL to login to the MQWeb Server APIs. The URL will be composed as follows:

> ☑ **Tip:**
> Remember that you can use the **dspmqweb** command to display the base URL for the MQWeb Server.




```
https://localhost:9443/ibmmq/rest/v1/login
```

8. Perform the following steps to submit your Login request:

    a. Select the **GET** method.

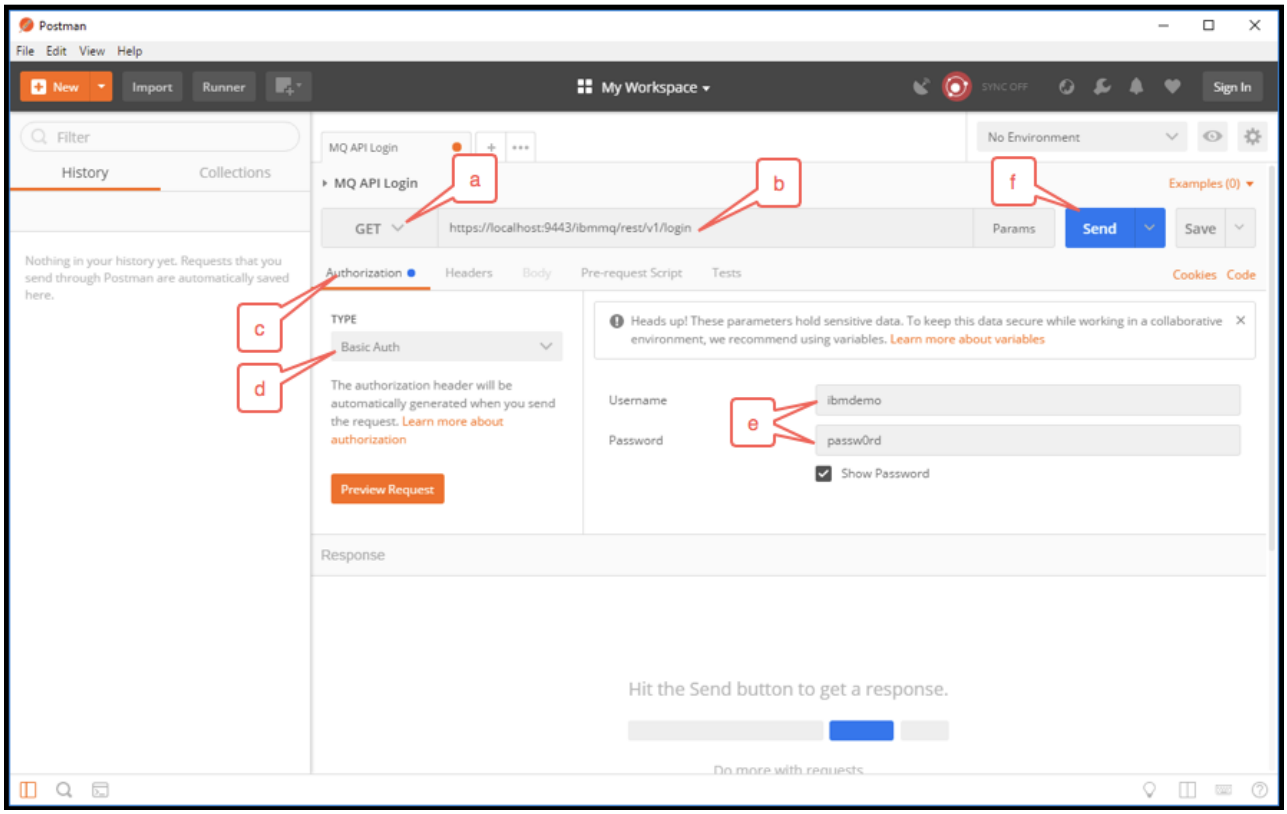
    b. Enter the Login URL from above.

    c. Select the **Authorization** tab.

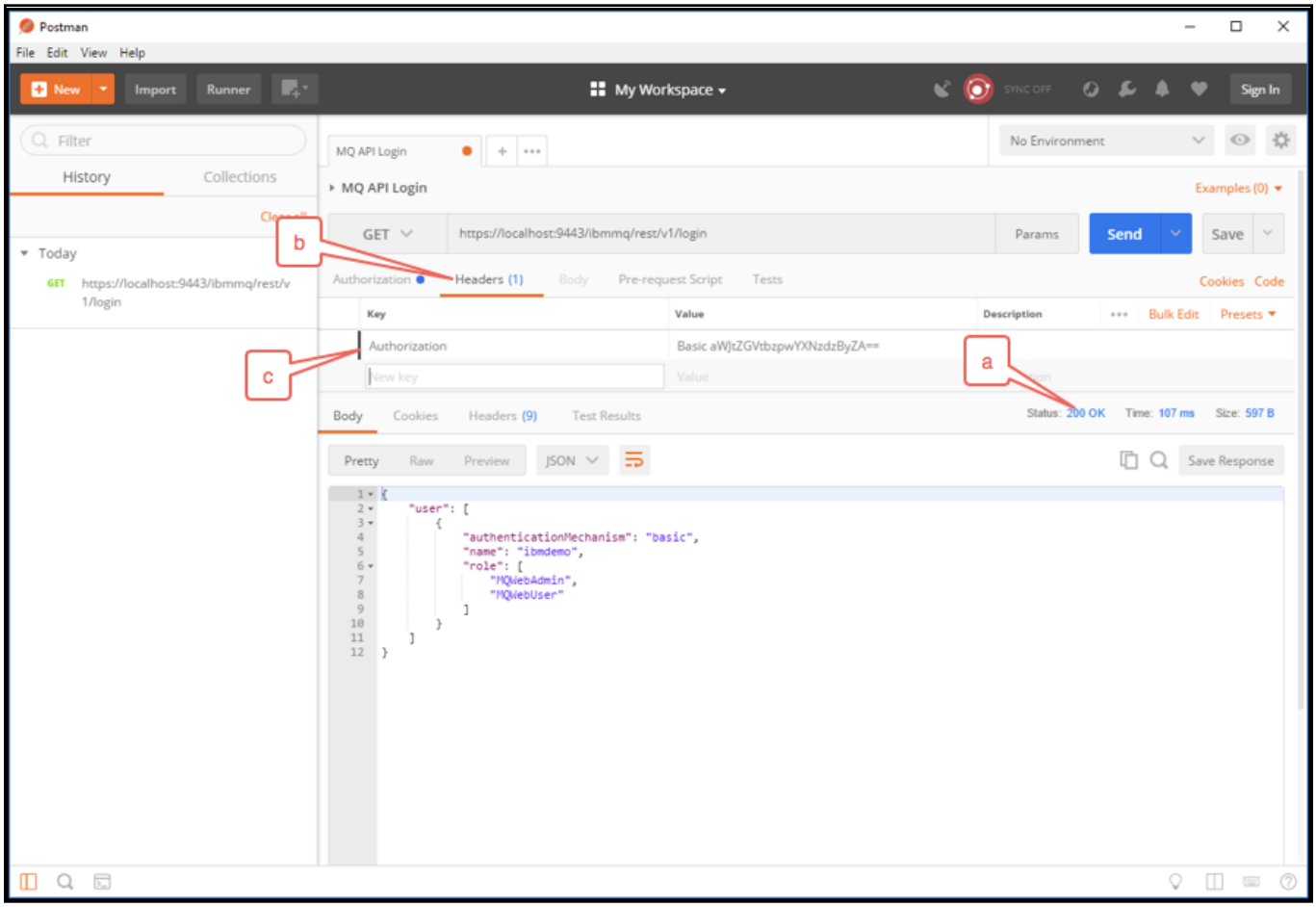
    d. Select **Basic Auth** for the Type.

    e. Enter **ibmdemo** and **passw0rd** as the Username and Password.

    f. Click on the **Send** button to submit your Login request.



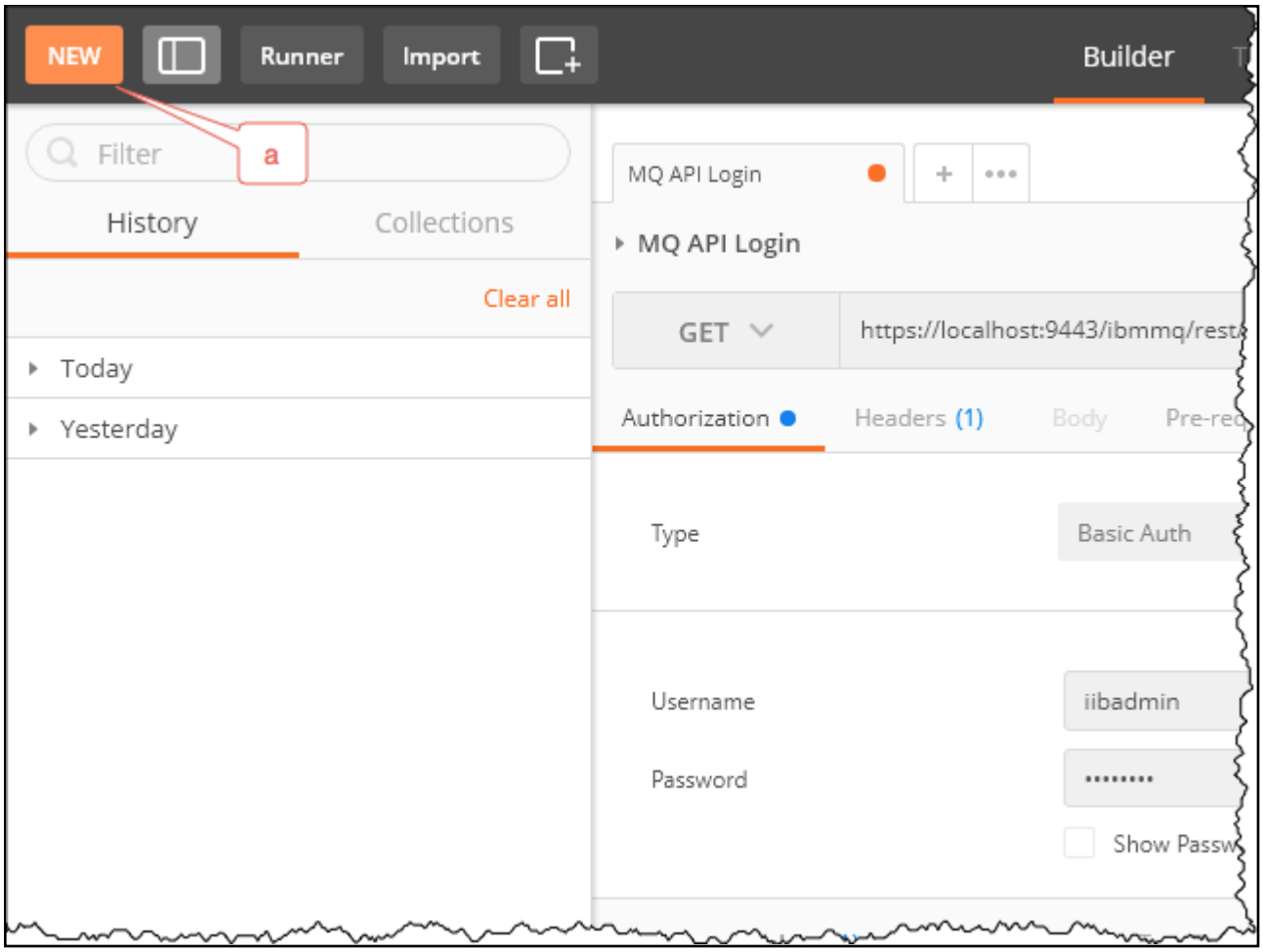
9. The MQWeb Server will respond with an HTTP Response code of **200 OK**. The response will also include an **Authorization** header. The value from this header may be used in subsequent API calls to avoid having to include the userid and password with each subsequent REST API call. Select the **Headers** tab and then copy the value from the **Authorization** header for later use.



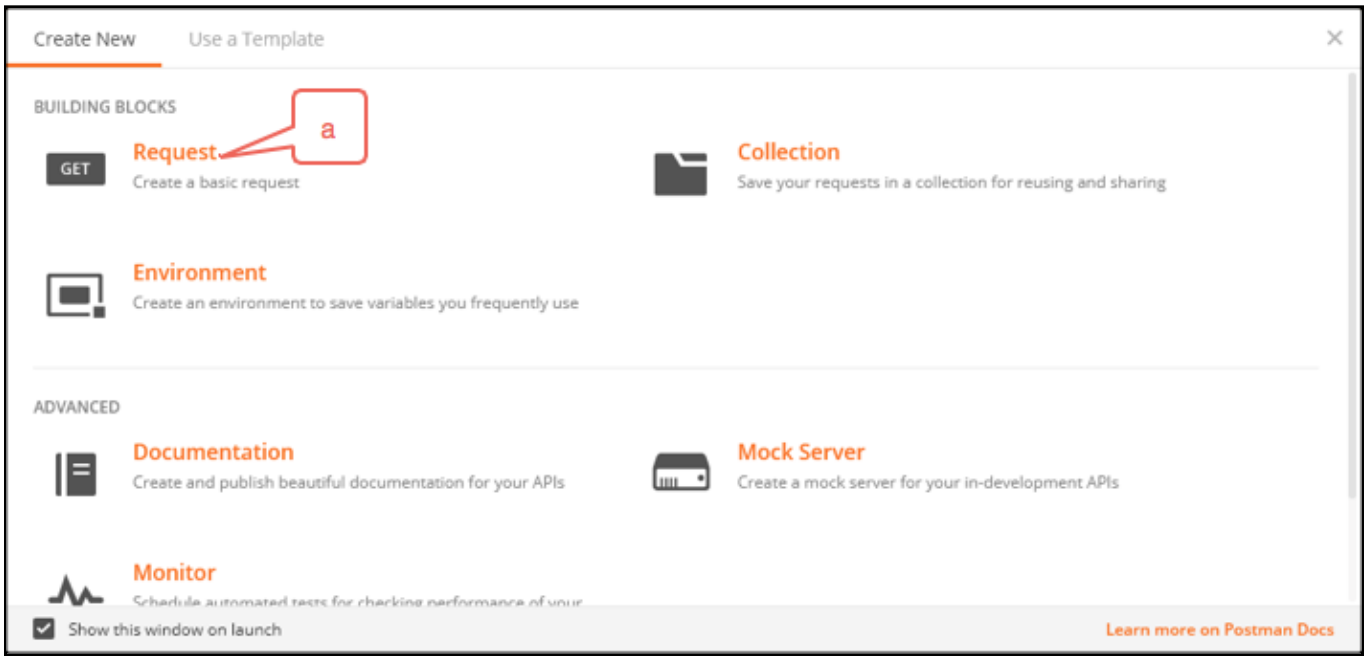
> **Note:**
> Login requests have a limited lifespan. You may need to re-run the Login request if your login session expires between API calls.

Now that you have created a collection and called the Login API you are ready to begin working with the IBM MQ REST APIs. Perform the following steps for every new API request that you create.

10. Click on the **NEW** button at the top left corner of the Postman tool.



11. The “Getting Started” wizard will appear again. Click on the **Request** icon to proceed.



You are now ready to begin testing APIs.

Using Administrative APIs

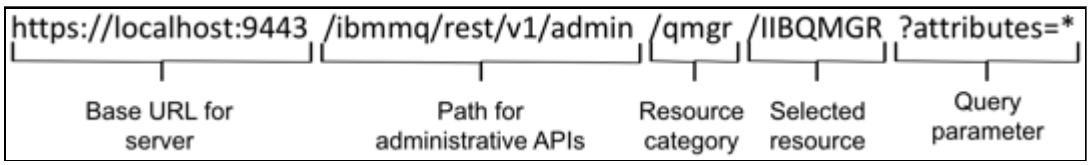
You can perform a number of administrative tasks using the IBM MQ REST APIs. For this lab you will try out just a few of the APIs that are available. You should refer to the IBM MQ Version 9 Knowledge Center for information on how to perform the full range of administrative tasks that are available using the APIs. Perform the following steps to start exploring some of the functions that are available:

- Query a queue manager's attributes
- Execute an MQSC command to get a full list of queue manager attributes
- Create a queue
- Update the description for the queue
- Delete the queue

1. Your first exercise is to use a REST API to retrieve attributes of the **IIBQMGR** queue manager. Requesting information is performed with an HTTP **GET** request. Create a new request in Postman and ensure that you save it to your Collection. Next, complete the following steps:

    a. Select the **GET** method.

    b. Enter an appropriate URL to retrieve the attributes for the **IIBQMGR** queue manager. The URL will be composed as follows:

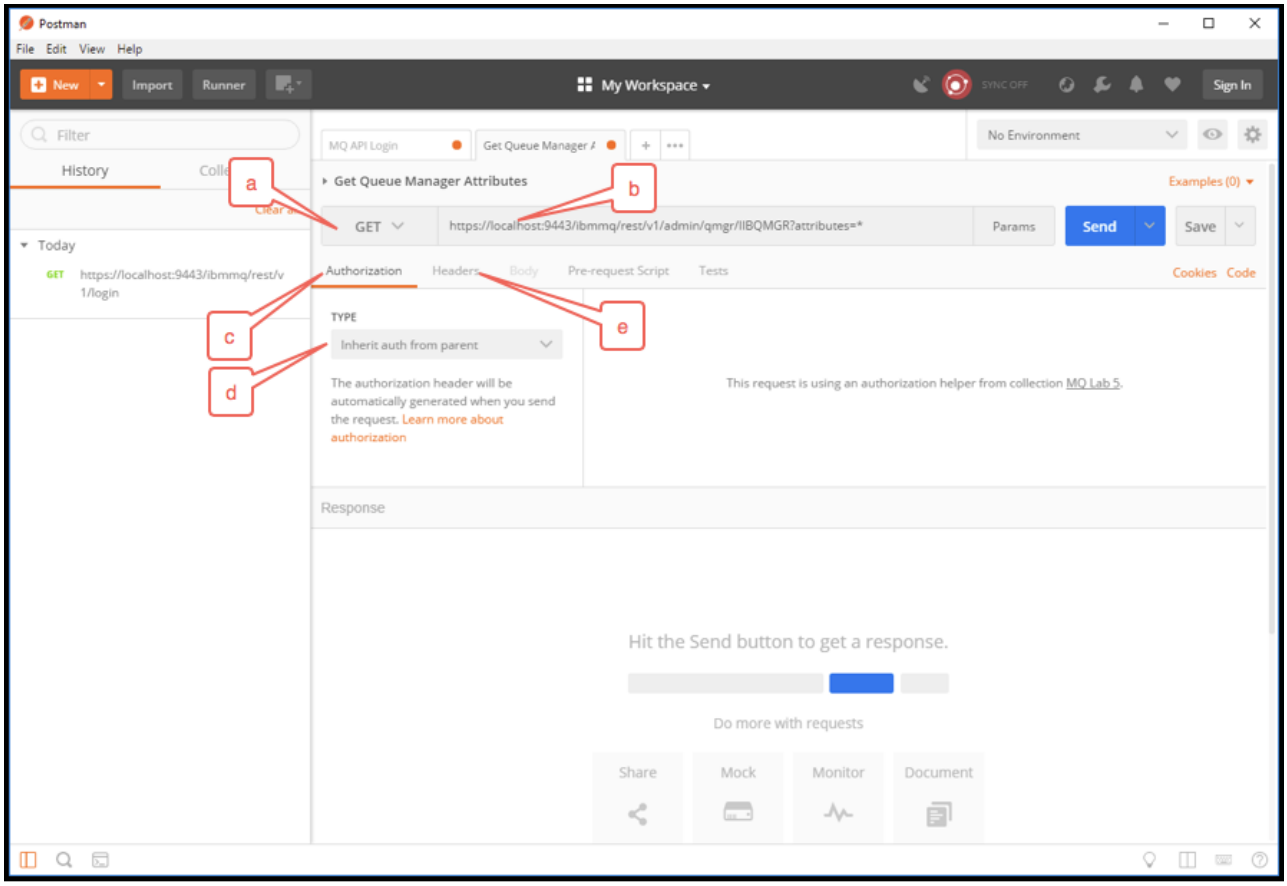



```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/IIBQMGR?attributes=*
```

    c. Select the **Authorization** tab.

    d. Select **Inherit auth from parent** for the Type.

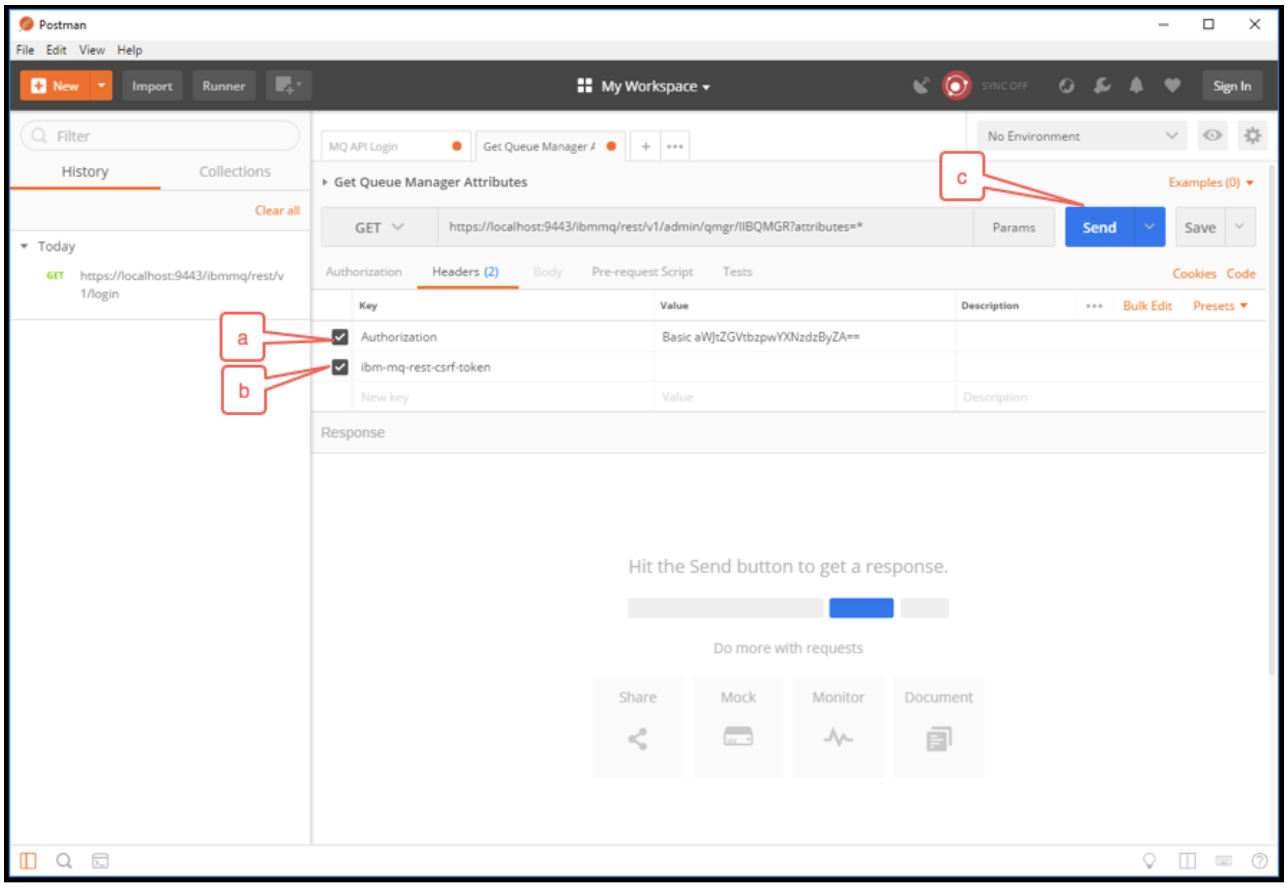
    e. Click on the **Headers** tab.



f. Add the following headers:

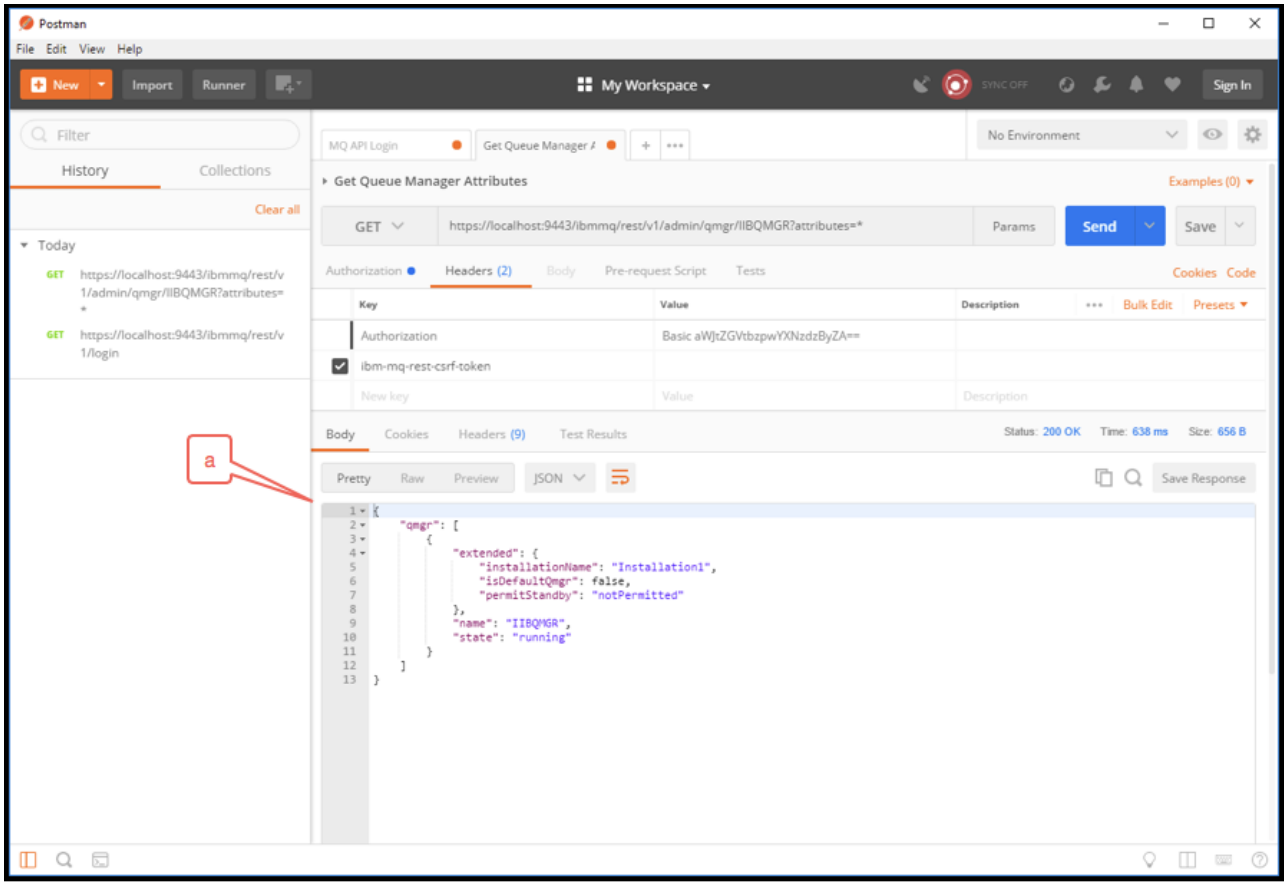
- The **Auhorization** header with the value returned from the Login request.
- The **ibm-mq-rest-csrf-token** header to enable CRSF protection for the MQWeb REST API. Note that it is not necessary to set a value for this header.

g. Click on the **Send** button to submit your GET request.



h. Review the results. Notice that for release 9.0.5 of IBM MQ the REST API does not return all of the queue manager properties that the equivalent **DIS QMGR** MQSC command provides. You may find similar differences with other REST APIs.

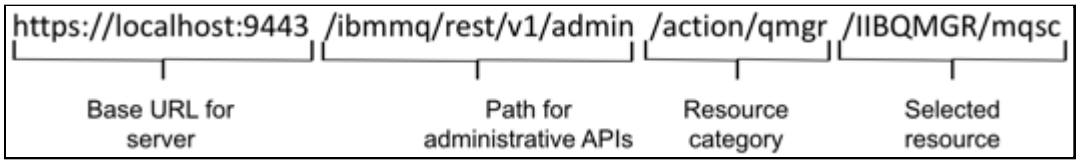
> **Note:**
> With the new Continuous Delivery support for IBM MQ you can expect to see enhancements to the REST APIs delivered on a frequent basis.



2. There is still a way to retrieve full queue manager information. The IBM MQ REST API supports sending MQSC commands similar to how you would execute commands in the **runmqsc** command shell. Perform the following steps to send an MQSC command to the **IIBQMGR** queue manager using the **mqsc** API resource.

    a. Since an MQSC command can do more than simply read data, your request must be sent as an HTTP POST. Create a new HTTP POST request in Postman and ensure that you save it to your Collection.

    b. Compose an appropriate URL to invoke an MQSC command and then enter it in the URL field:

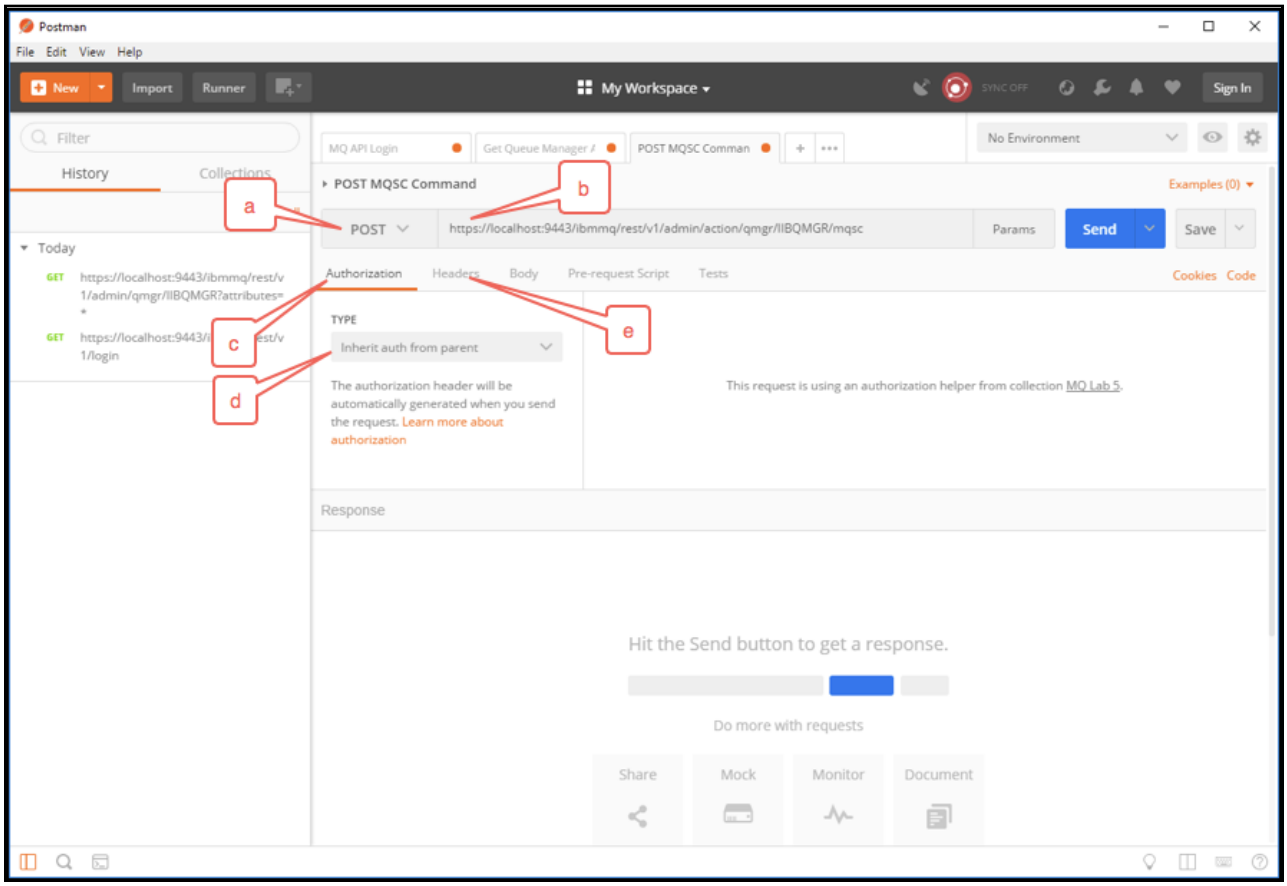



```
https://localhost:9443/ibmmq/rest/v1/admin/action/qmgr/IIBQMGR/mqsc
```

c. Select the **Authorization** tab.

d. Select **Inherit auth from parent** for the Type.

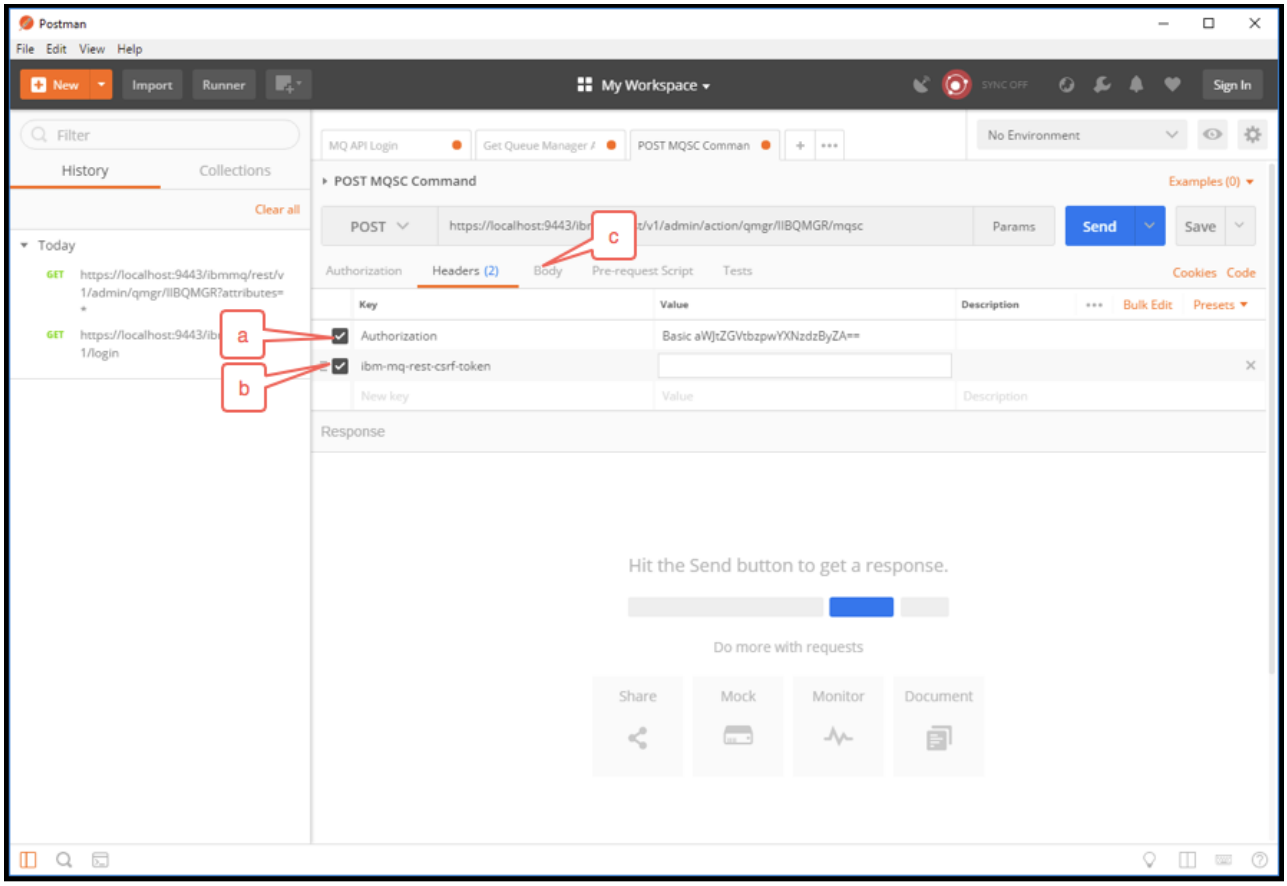
e. Click on the **Headers** tab.



f. Add the following headers:

- The **Auhorization** header with the value returned from the Login request.
- The **ibm-mq-rest-csrf-token** header to enable CRSF protection for the MQWeb REST API. Note that it is not necessary to set a value for this header.

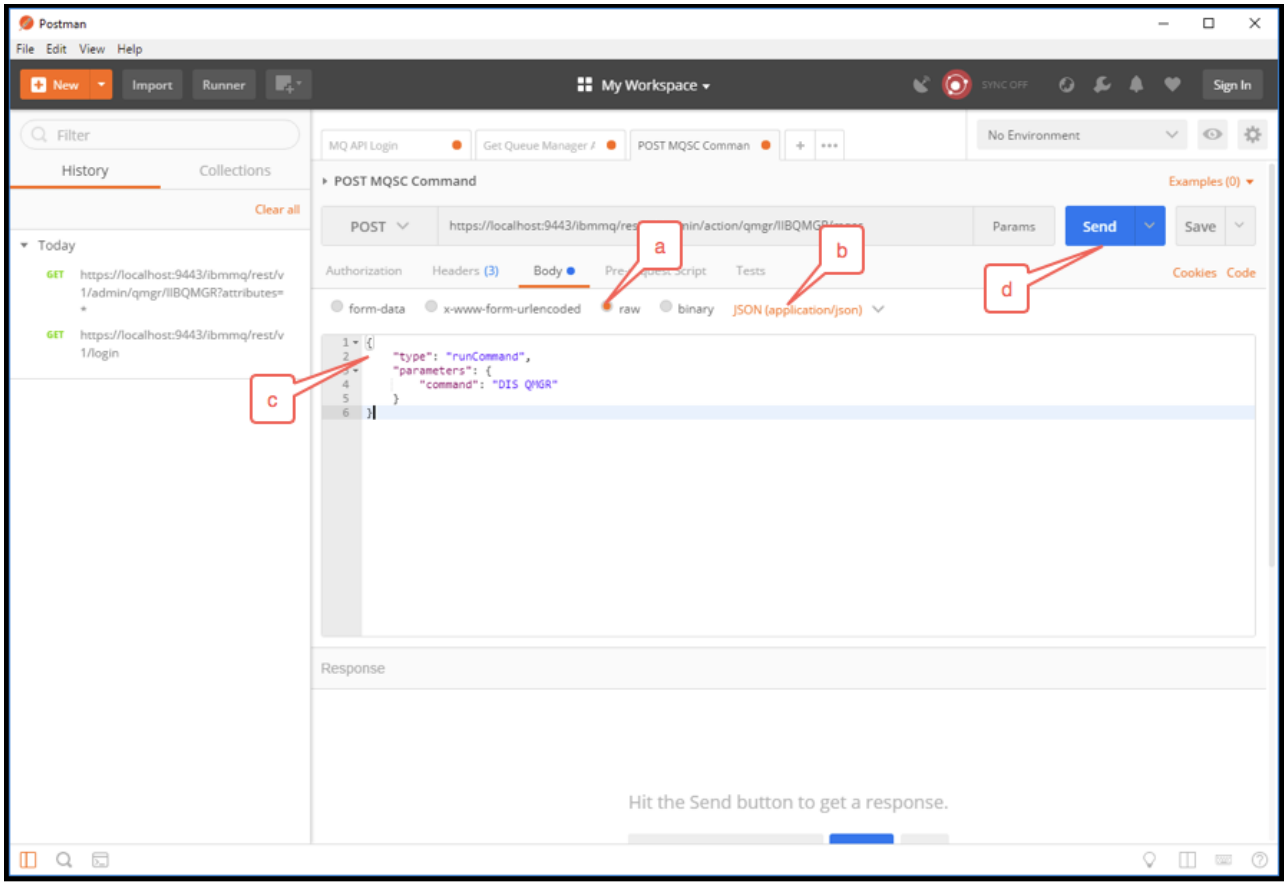
g. Click on the **Body** tab.



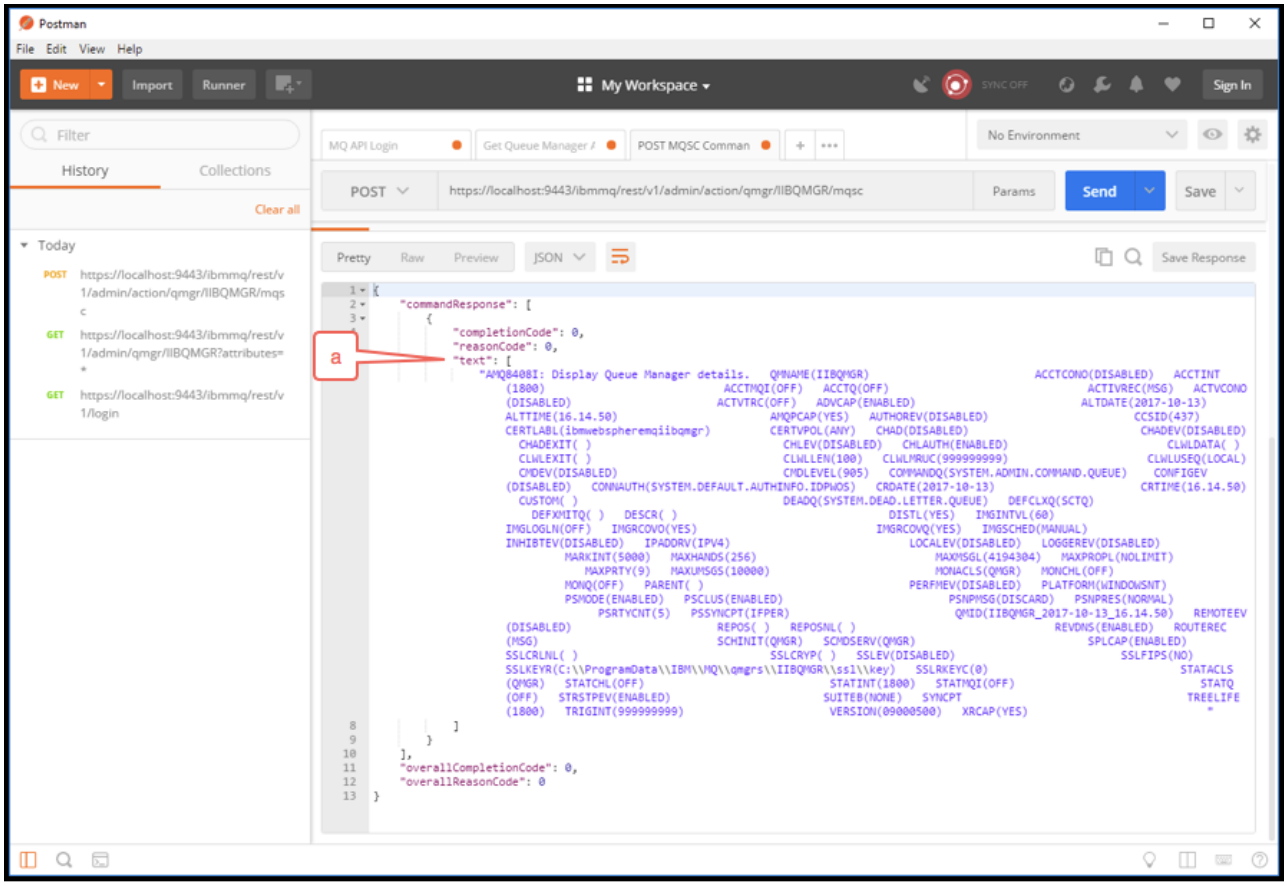
h. You will send your MQSC command in the body of a JSON formatted message. In order to send JSON you must first specify **raw** as your message body type and **JSON (application/json)** as the **Content-Type**. Next you need to add your JSON payload that will contain the MQSC command that you would like to execute. Enter the following JSON text in the Body section:

```
{
    "type": "runCommand",
    "parameters": {
        "command": "DIS QMGR"
    }
}
```

i. Click on the **Send** button.



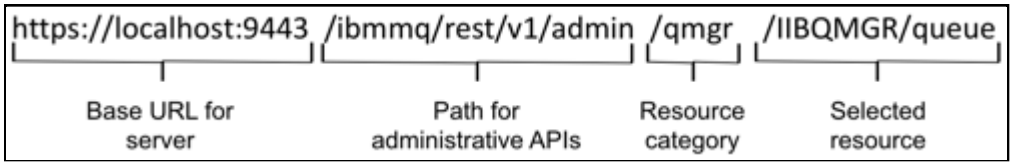
j. As you can see, the response from this REST API is sent in JSON format, with the response from the MQSC command returned in a single text field. This is an area for enhancement in later releases of IBM MQ.



3. The next REST API to review is using an HTTP POST request on the **queue** API resource to create a new queue. The steps to follow are similar to the previous lab step where you sent an MQSC command to the **mqsc** API resource.

    a. Create a new HTTP POST request in Postman and ensure that you save it to your Collection.

    b. Compose an appropriate URL to invoke an MQSC command and then enter it in the URL field:

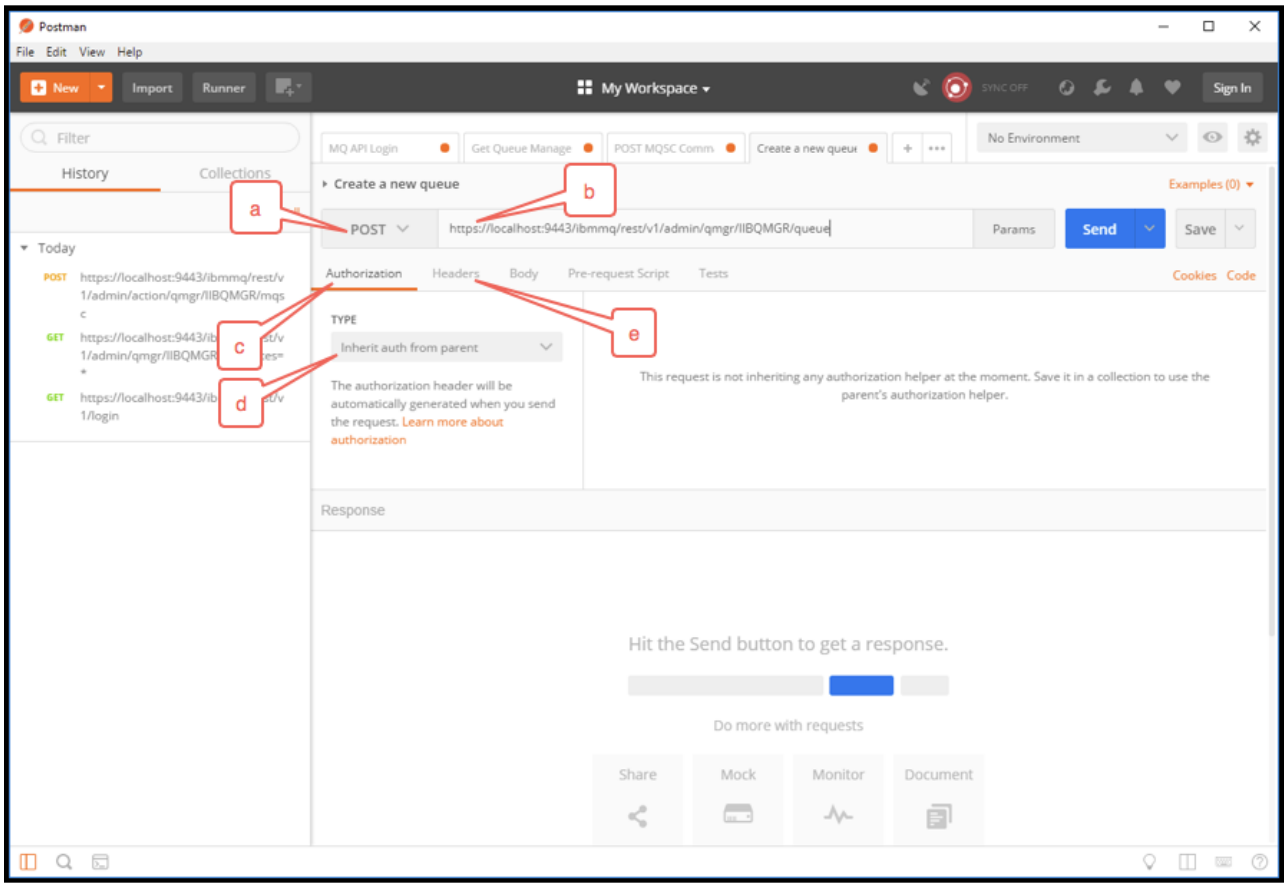



```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/IIBQMGR/queue
```

    c. Select the **Authorization** tab.

    d. Select **Inherit auth from parent** for the Type.

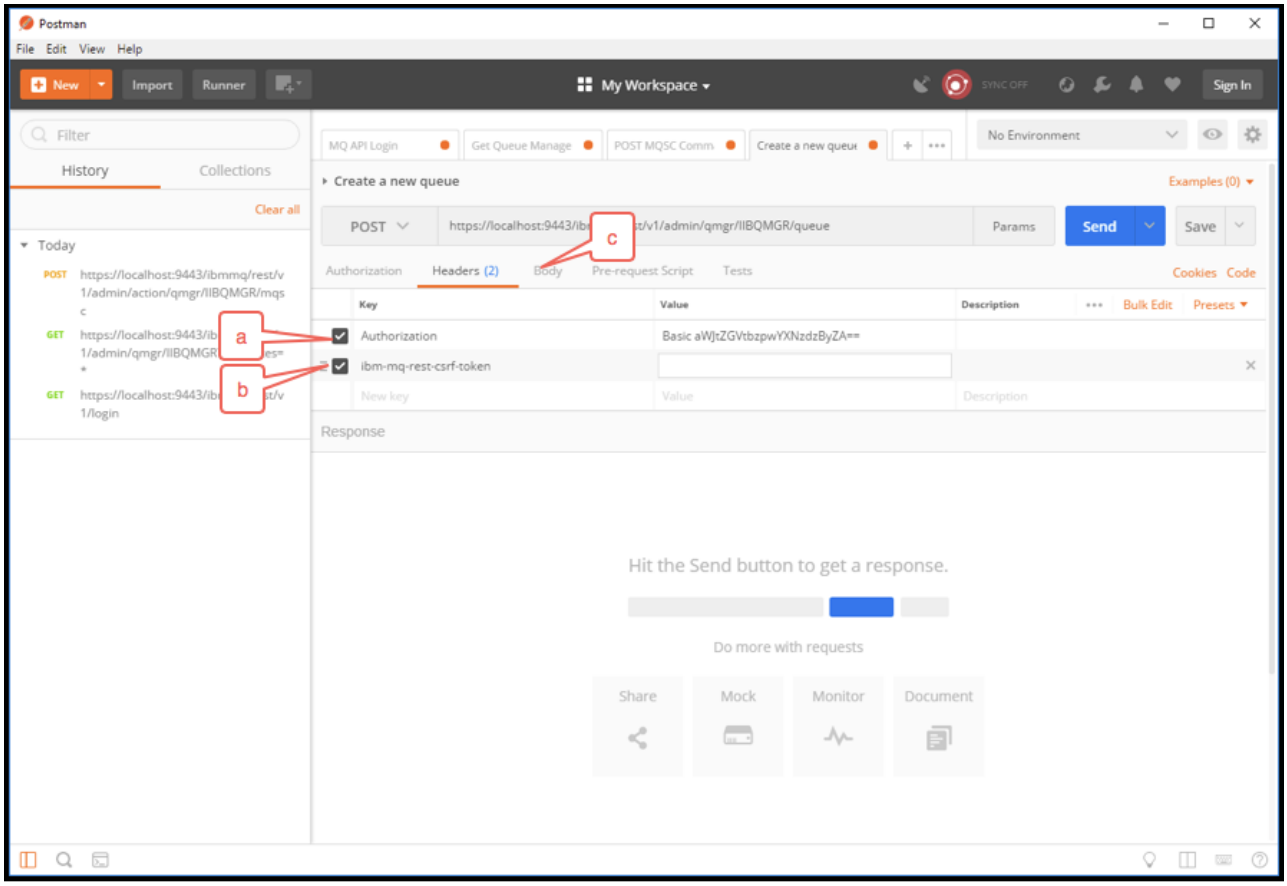
e. Click on the **Headers** tab.



f. Add the following headers:

- The **Auhorization** header with the value returned from the Login request.
- The **ibm-mq-rest-csrf-token** header to enable CRSF protection for the MQWeb REST API. Note that it is not necessary to set a value for this header.

g. Click on the **Body** tab.



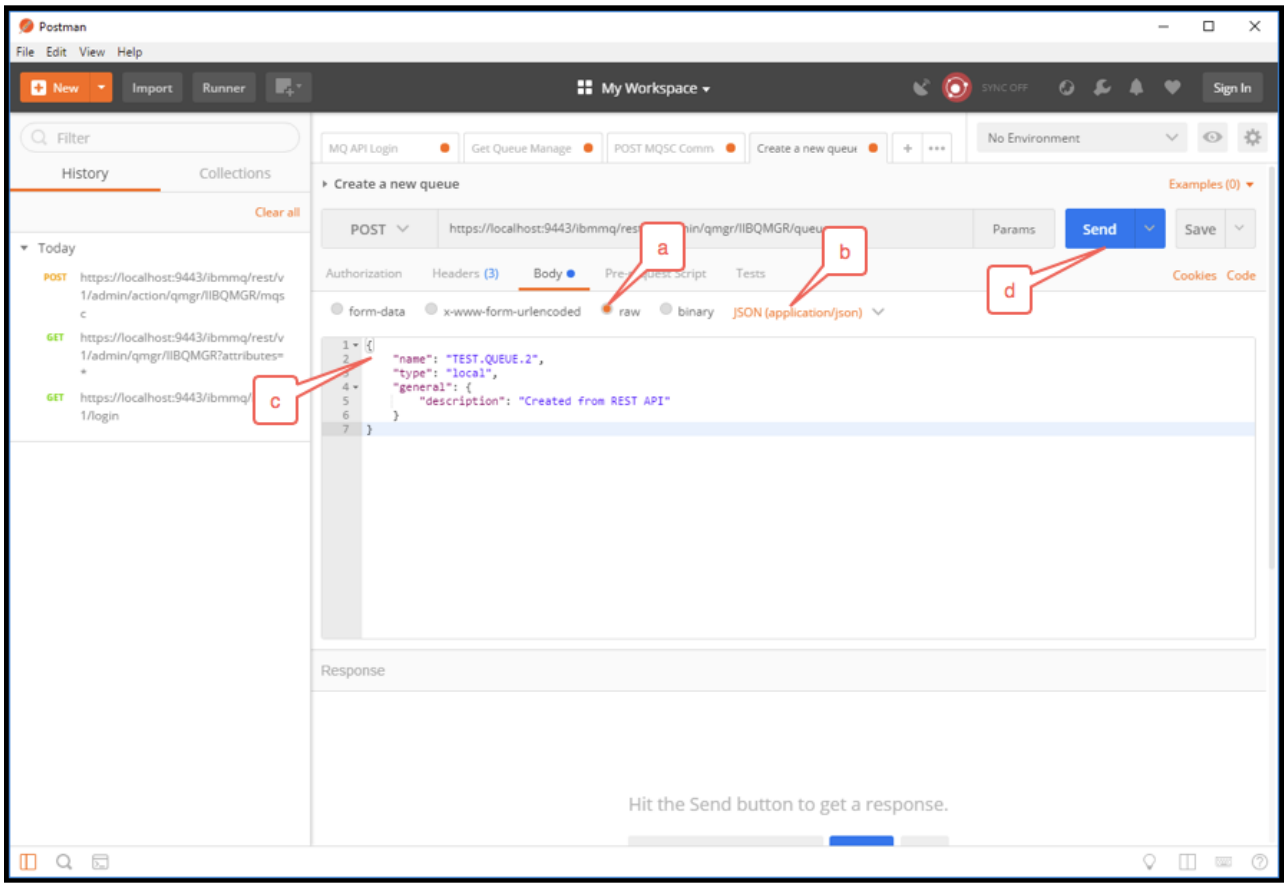
h. Click on the **Body** section.

i. You must specify **raw** as your message body type and **JSON (application/json)** as the **Content-Type**.

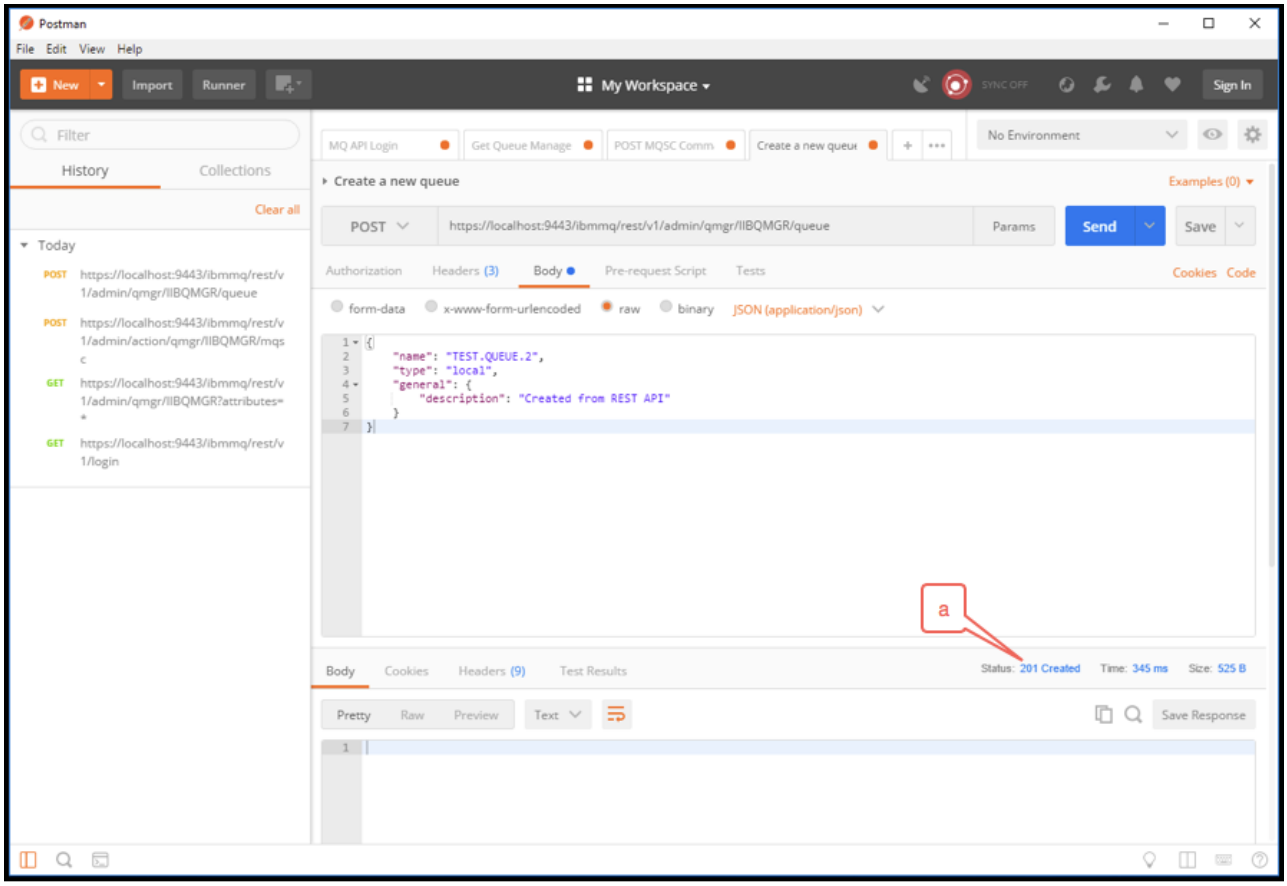
j. You will need to send a JSON formatted message with the parameters to use for creating the queue. In order to send JSON you must first specify **raw** as your message body type and **JSON (application/json)** as the **Content-Type**. Next you need to add your JSON message. Enter the following JSON text in the Body section:

```
{
    "name": "TEST.QUEUE.2",
    "type": "local",
    "general": {
        "description": "Created from REST API"
    }
}
```

k. Click on the **Send** button.



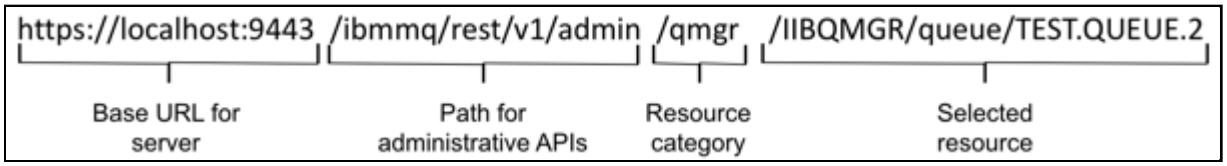
l. You should see an HTTP Response code of **201: Created** indicating that your API call completed successfully.



4. The next REST API to review is using an HTTP PATCH request on the **queue** API resource to allow you to change the description of your new queue. The steps to follow are similar to the previous lab step where you sent an HTTP POST request to create the IBM MQ queue.

    a. Create a new HTTP PATCH request in Postman and ensure that you save it to your Collection.

    b. Compose an appropriate URL to change an IBM MQ queue and then enter it in the URL field:

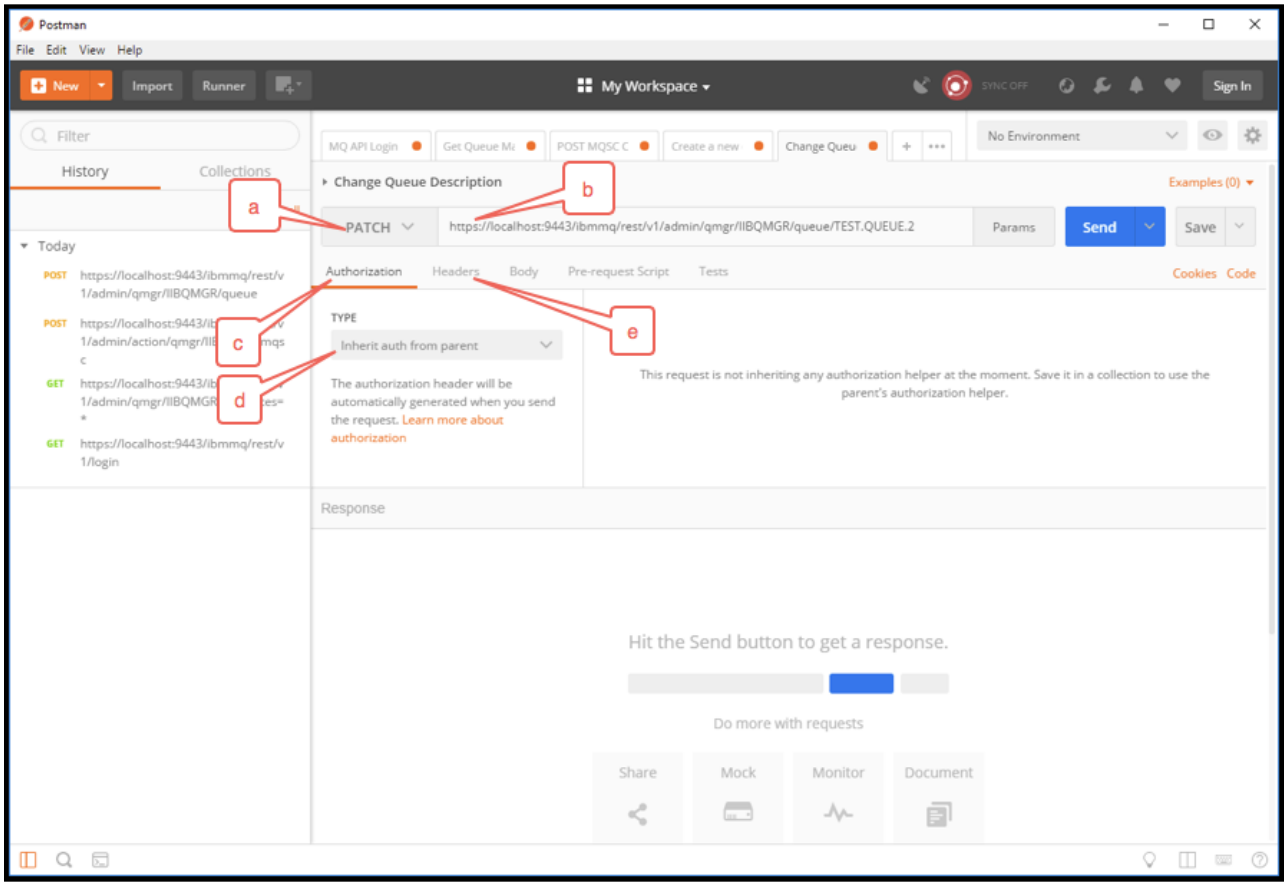



```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/IIBQMGR/queue/TEST.QUEUE.2
```

    c. Select the **Authorization** tab.

d. Select **Inherit auth from parent** for the Type.

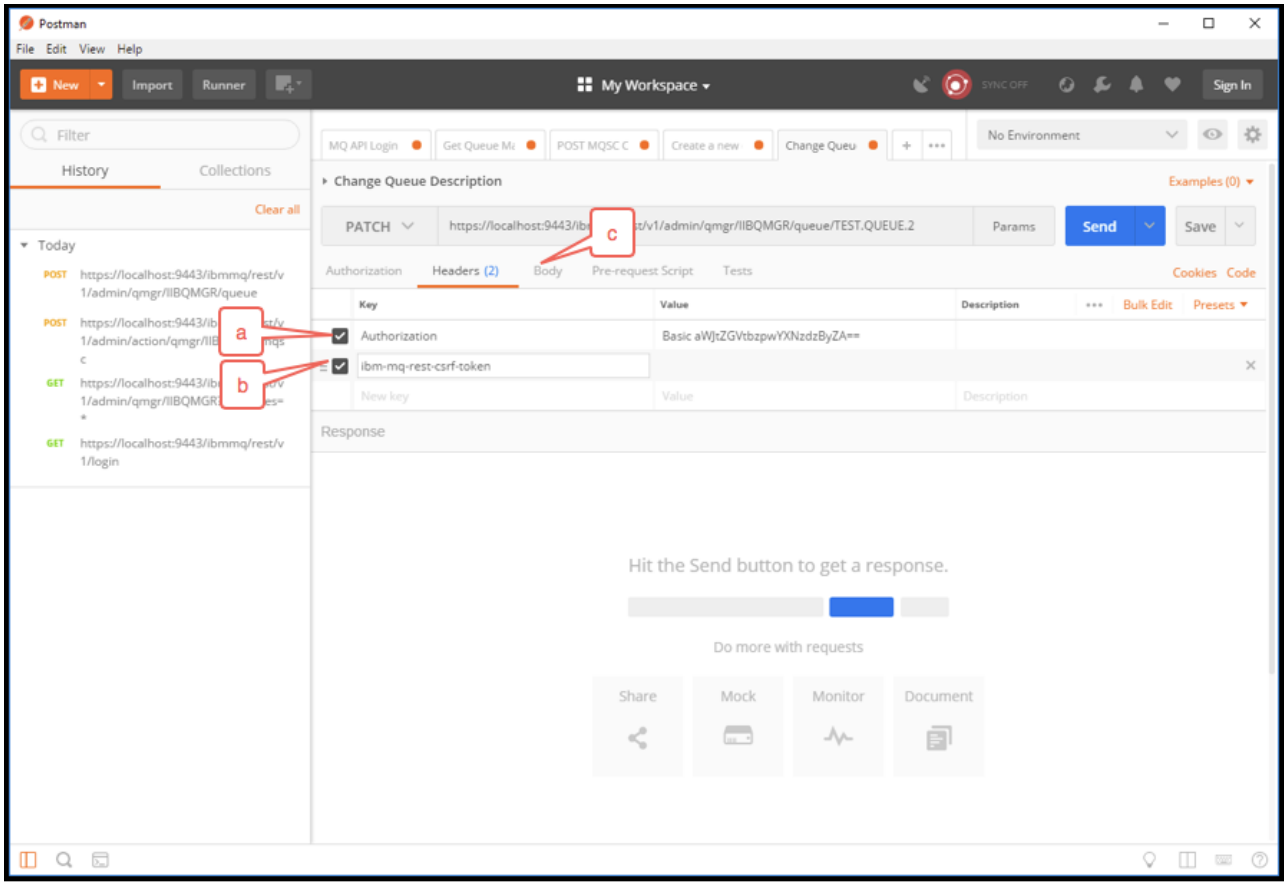
e. Click on the **Headers** tab.



f. Add the following headers:

- The **Auhorization** header with the value returned from the Login request.
- The **ibm-mq-rest-csrf-token** header to enable CRSF protection for the MQWeb REST API. Note that it is not necessary to set a value for this header.

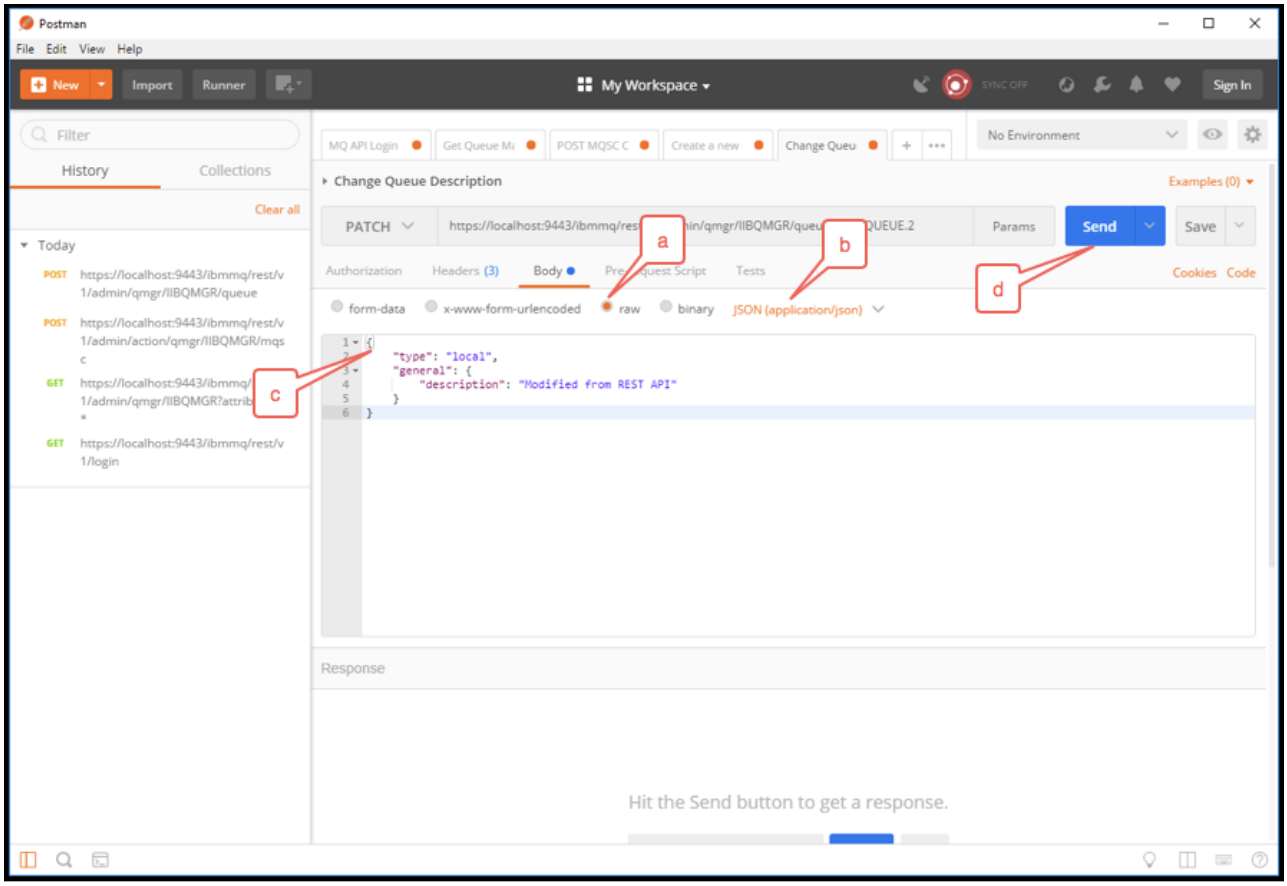
g. Click on the **Body** tab.



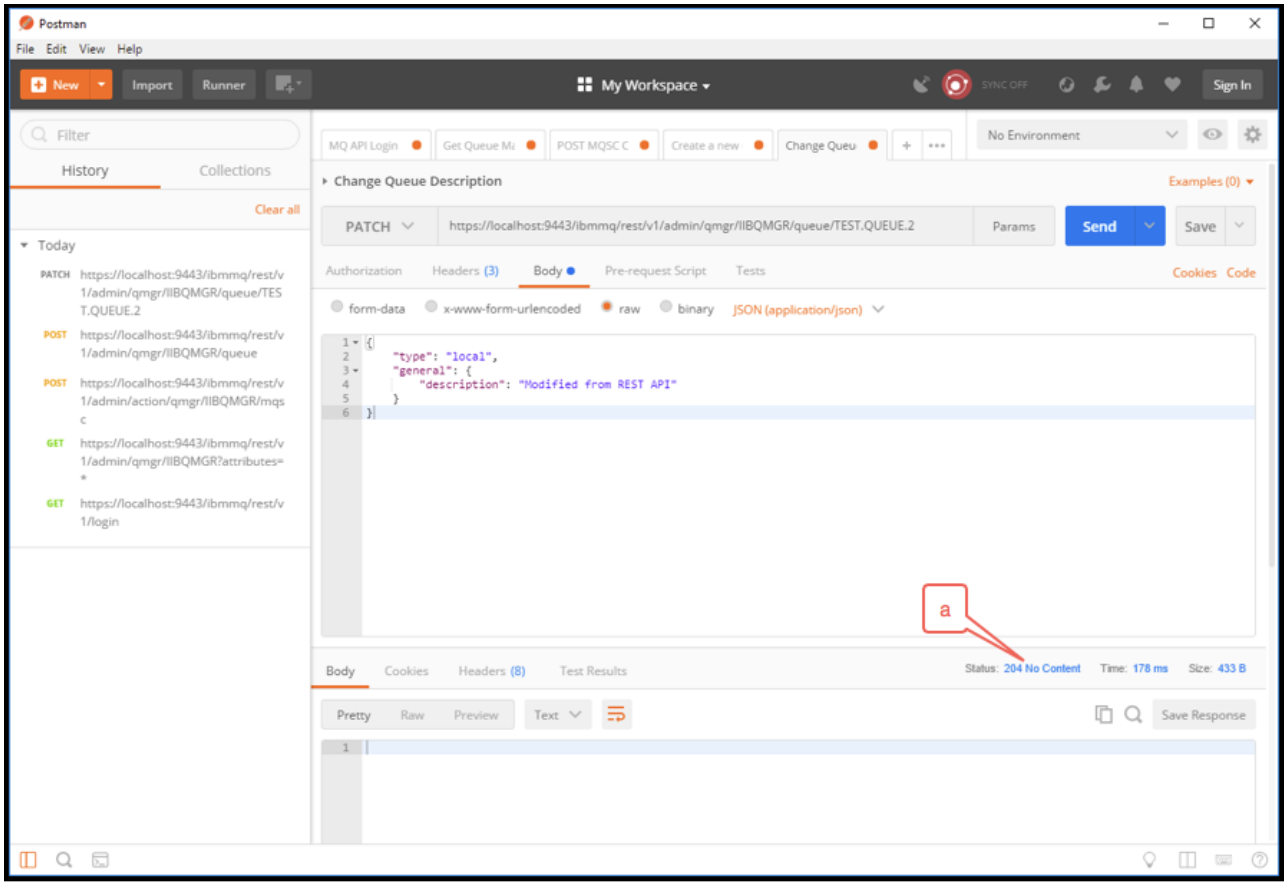
h. You will need to send a JSON formatted message with the parameters to use for modifying the queue. In order to send JSON you must first specify **raw** as your message body type and **JSON (application/json)** as the **Content-Type**. Next you need to add your JSON message. Enter the following JSON text in the Body section:

```
{
    "type": "local",
    "general": {
        "description": "Modified from REST API"
    }
}
```

i. Click on the **Send** button.



j. You should see an HTTP Response code of **204: No Content** indicating that your API call completed successfully.

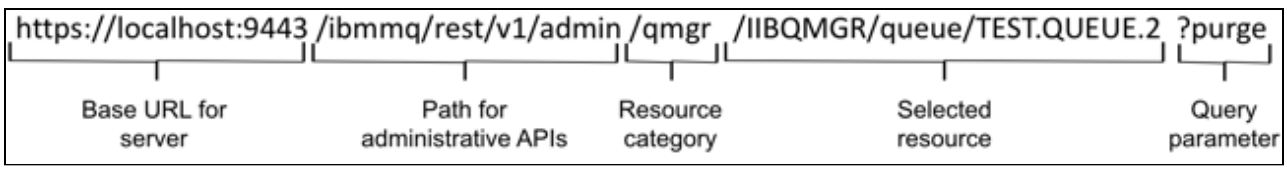


> **Tip:**
> Receiving a **204: No Content** response may not give you enough assurance that the API functioned as desired. You can always refer to the IBM MQ Console or the MQ Explorer to verify that the change took effect.

5. The last administrative API to review is using an HTTP DELETE request on the **queue** API resource in order to delete the queue that you had previously created. The steps to follow are similar to the previous labs where you sent an HTTP POST or PATCH request. For this step you will include a query parameter on the URL to indicate that you want to purge the queue of any messages as it is deleted.

    a. Create a new HTTP DELETE request in Postman and ensure that you save it to your Collection.

    b. Compose an appropriate URL to delete an IBM MQ queue and then enter it in the URL field. Ensure you add the query parameter to purge the queue:

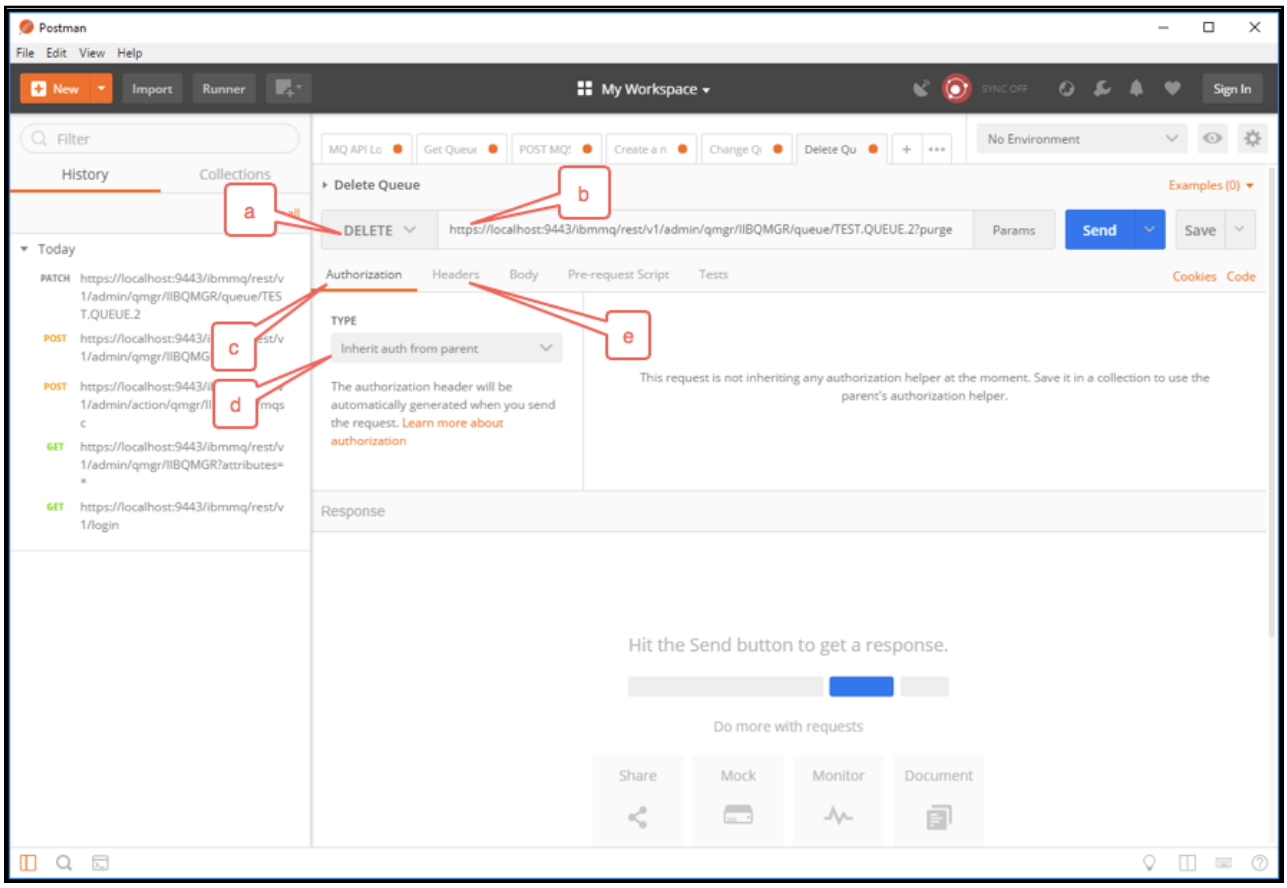


```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/IIBQMGR/queue/TEST.QUEUE.2?purge
```

c. Select the **Authorization** tab.

d. Select **Inherit auth from parent** for the Type.

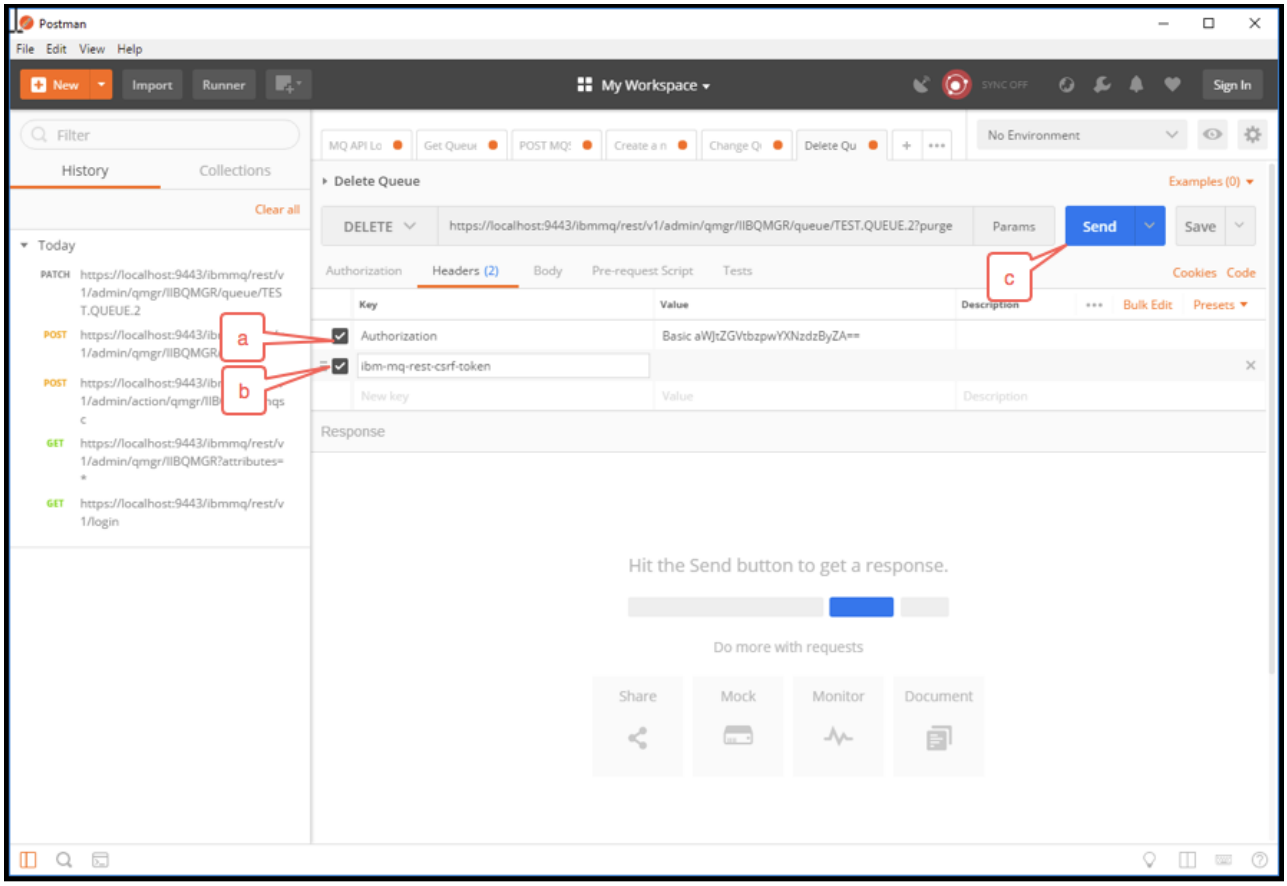
e. Click on the **Headers** tab.



f. Add the following headers:

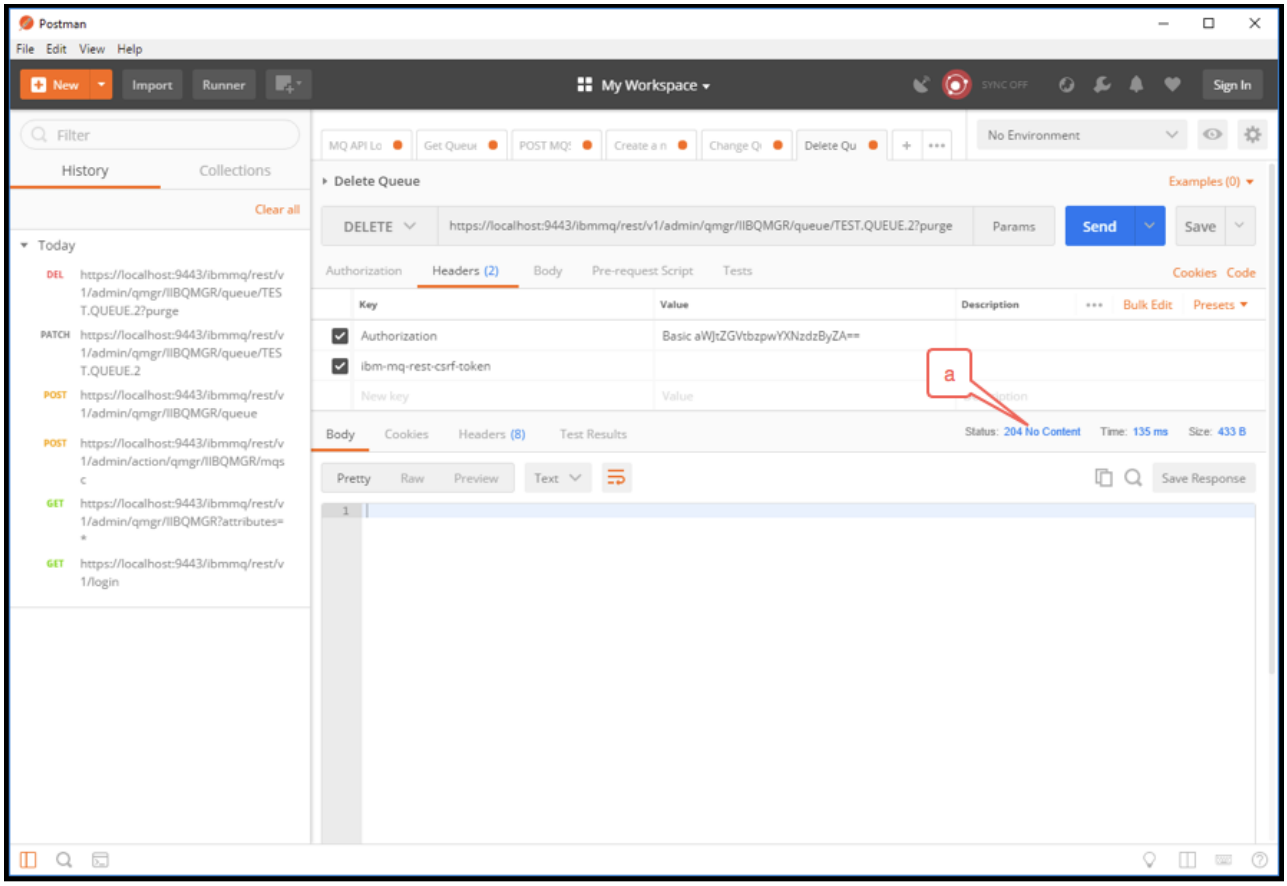
- The **Auhorization** header with the value returned from the Login request.
- The **ibm-mq-rest-csrf-token** header to enable CRSF protection for the MQWeb REST API. Note that it is not necessary to set a value for this header.

g. Click on the **Send** button.



> **Note:**
> The HTTP DELETE request does not require a JSON payload.

h. You should see an HTTP Response code of **204: No Content** indicating that your API call completed successfully.



> **☑ Tip:**
> Receiving a **204: No Content** response may not give you enough assurance that the API functioned as desired. You can always refer to the IBM MQ Console or the MQ Explorer to verify that the change took effect.

You have explored several of the administrative APIs. Now you will look at the messaging APIs that are available.

Using Messaging APIs

IBM MQ Version 9.0.4 introduced additional REST APIs that allow applications to PUT and GET messages to or from IBM MQ queues. These APIs are implemented by the HTTP POST and DELETE methods. Full details on what IBM MQ features are currently implemented in the REST APIs may be found in the IBM MQ Version 9 Knowledge Center. In this lab you will perform basic **PUT** (via HTTP POST) and **GET** (via HTTP DELETE) operations against an IBM MQ queue.

Using the messaging APIs is similar to using the administrative APIs.

> **⚠ Important:**

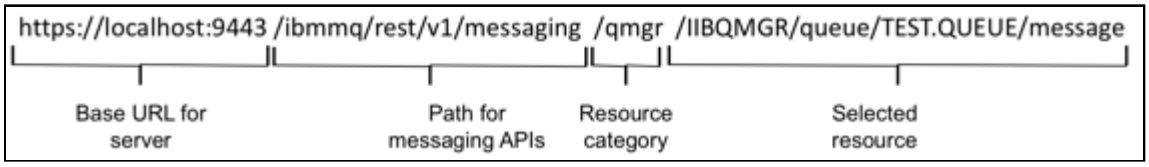
There is a subtle, but significant, difference in how authorization works for the messaging REST APIs. For the messaging REST APIs, the security principal (The userid that you used for the Login REST API) must be granted **PUT** authority to the queue to use the **POST** API, and must have **GET**, **INQ** and **BROWSE** authorities to the queue to use the **DELETE** API.

Other differences with the messaging APIs is that the **POST** method will need to include a number of HTTP headers to set properties that you would normally set in the **MQMD** message header when using programming languages. Also, the **DELETE** method performs a destructive read of the message from the queue, similar to how the **GET** method in a programming language works. With the **DELETE** method the message payload will be returned in the response body upon successful completion of the request.

1. Perform the following steps to use the REST API to put a message on an IBM MQ Queue.

    a. Create a new HTTP POST request in Postman and ensure that you save it to your Collection.

    b. Next, compose an appropriate URL to create the POST request:

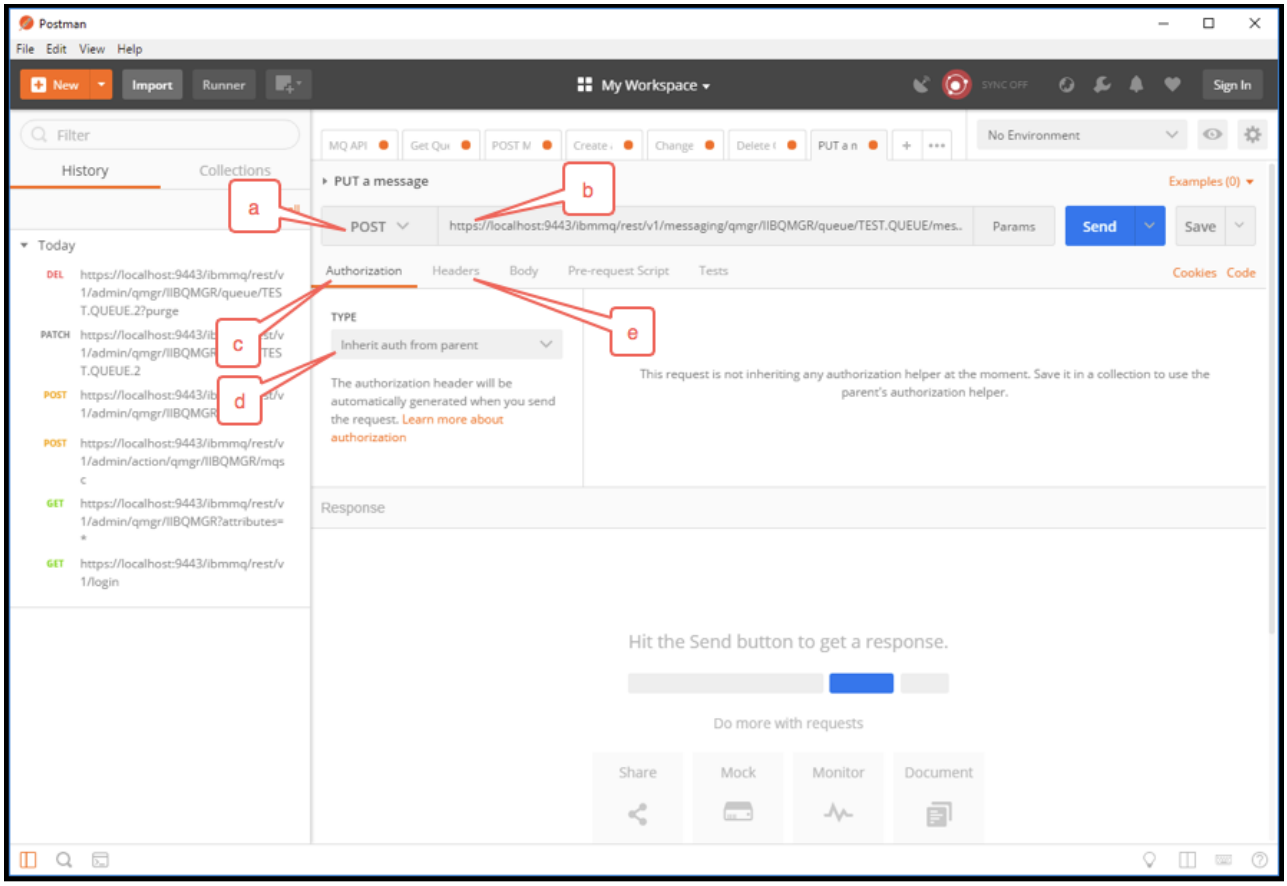



```
 https://localhost:9443/ibmmq/rest/v1/messaging/qmgr/IIBQMGR/queue/TEST.QU
EUE/message
```

    c. Select the **Authorization** tab.

    d. Select **Inherit auto from parent** for the Type.

    e. Click on the **Headers** tab.



f. The HTTP POST request requires the **Authorization**, **Content-Type** and **ibm-mq-rest-csrf-token** HTTP headers. You may include additional HTTP headers as desired. Click on the **Headers** section and add the HTTP headers shown below.

Mandatory HTTP Headers:

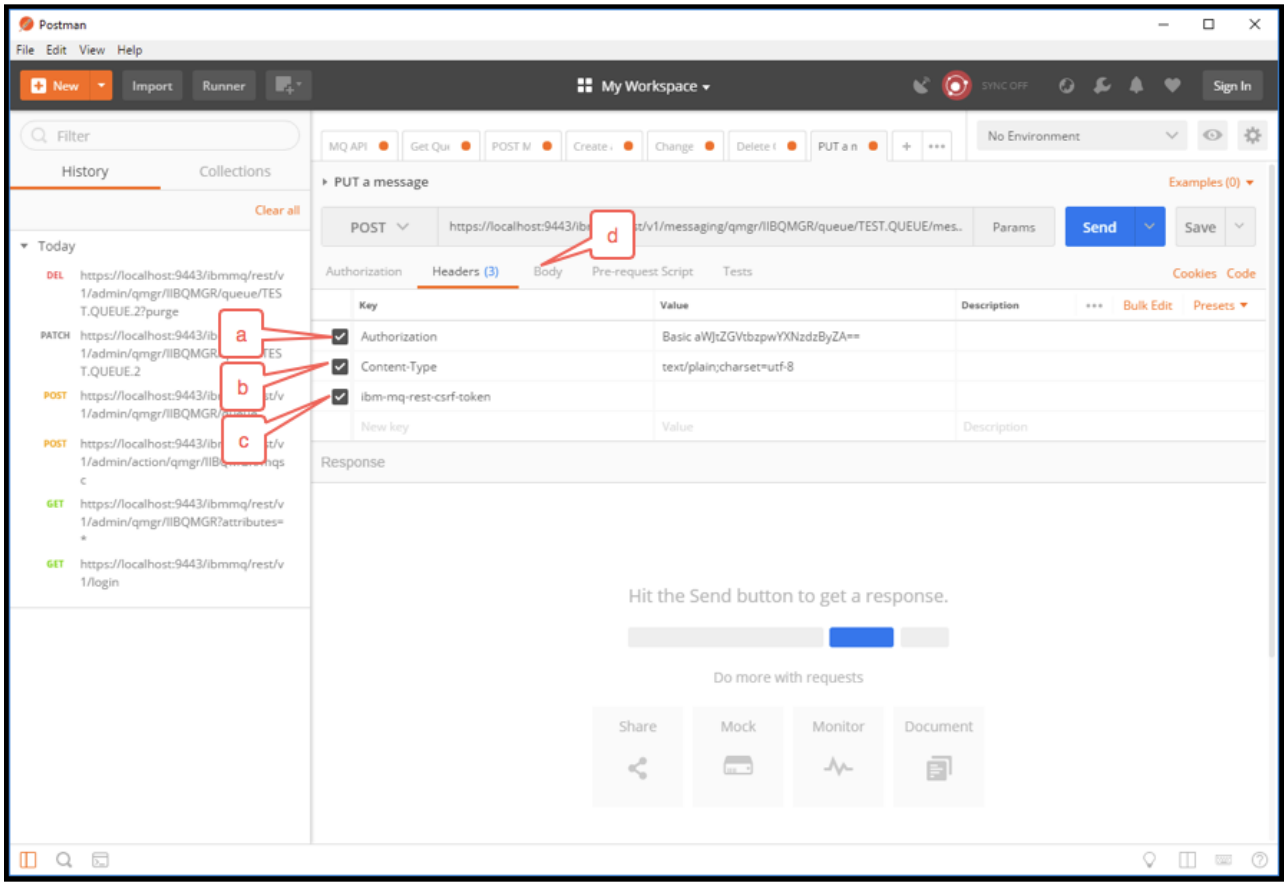
Header Name	Value
Authorization	*< Created when you invoked the Login REST API >*
Content-Type	text/plain;charset=utf-8 *(Other choices are available)*
ibm-mq-rest-csrf-token	*< leave blank >*

Optional HTTP Headers:

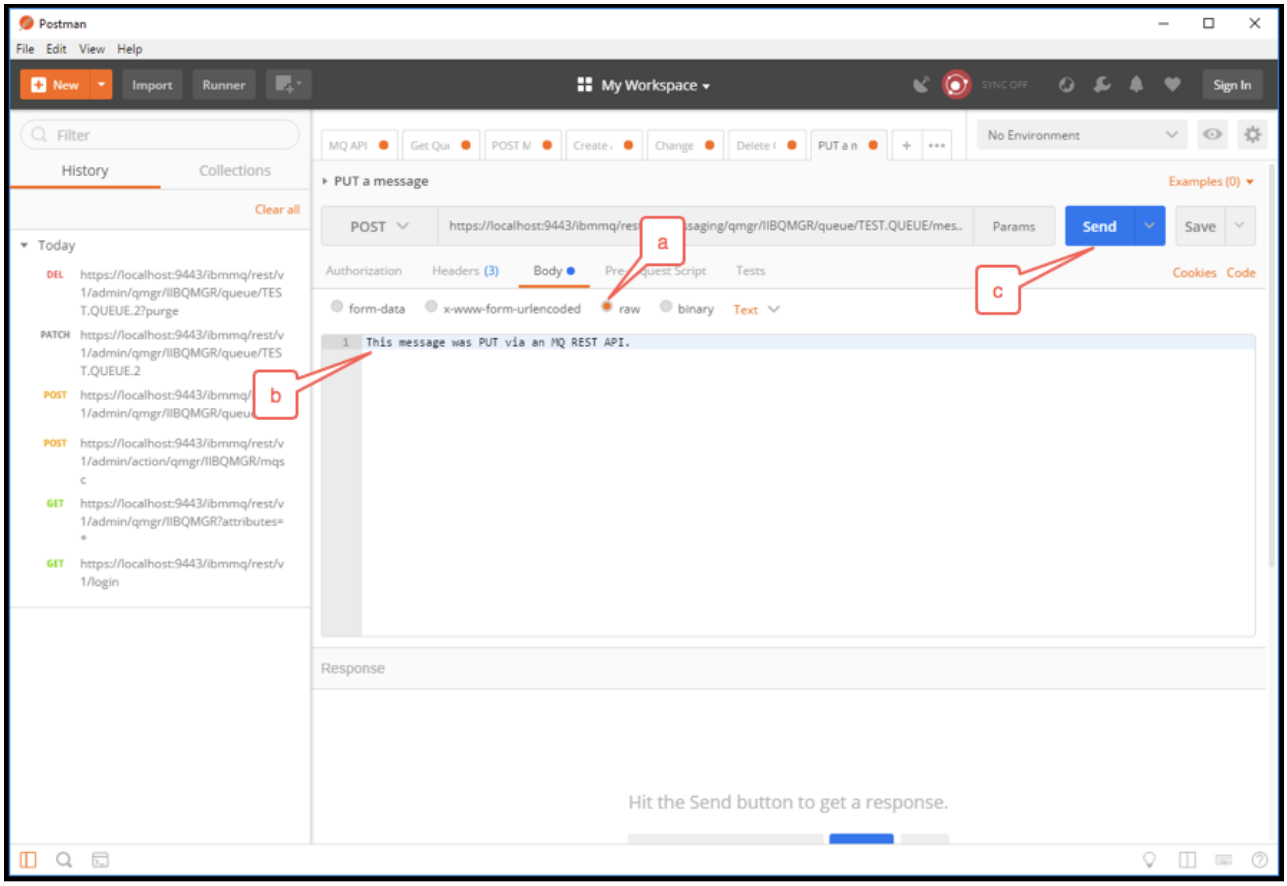
Header Name	Value
Accept-Language	*< language for exceptions or error messages >*

Header Name	Value
ibm-mq-md-correlationId	*< 48 character hex encoded string >*
ibm-mq-md-expiry	*< Integer value between 0 - 99999999900 >*
ibm-mq-md-persistence	nonPersistent (default) *or* persistent
ibm-mq-md-replyTo	*< myReplyQueue@myReplyQMgr >*

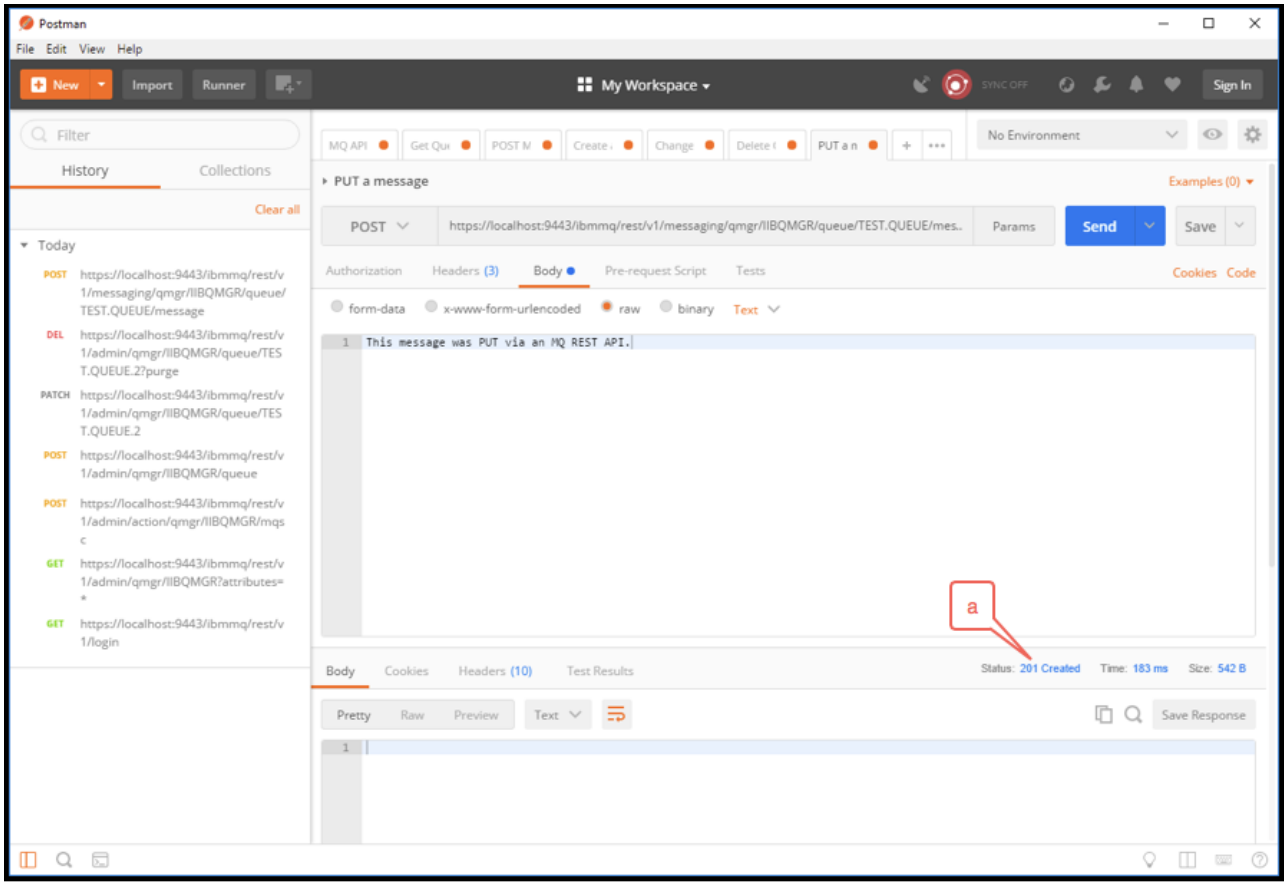
g. Enter the appropriate headers and then click on the **Body** tab.



h. You will send your message as a plain text message in the body of the PUT request. Specify **raw** as your message body type and **text/plain** as the **Content-Type**. Next enter the text of your message in the Body section and then click on the **Send** button.:



i. You should see an HTTP Response code of **201: Created** indicating that your message was successfully PUT to the queue.



2. Your last exercise is to use the HTTP DELETE method to perform a destructive read (MQ GET) of the message from the queue. The format of the API call is a bit different from the HTTP POST. With the HTTP POST you set properties in HTTP headers that map to the MQMD in an IBM MQ message payload. With the HTTP DELETE, you use query parameters on the URL that map to the MQGET options and the MQGMO structure. The following table lists the query parameters that may be used with IBM MQ 9.0.4

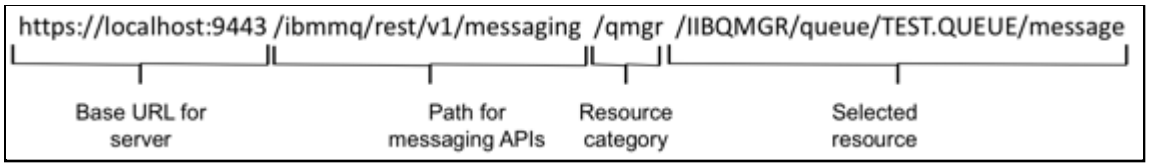
   Optional Query Parameters

Parameter Name	Value
correlationId	*< 48 character hex encoded string >*
messageId	*< 48 character hex encoded string >*
wait	*< Integer value between 0 - 2147483647 >*

   Perform the following steps to read the message from the queue.

   a. Create a new HTTP DELETE request in Postman and ensure that you save it to your Collection.

   b. Compose an appropriate URL to create the DELETE request:

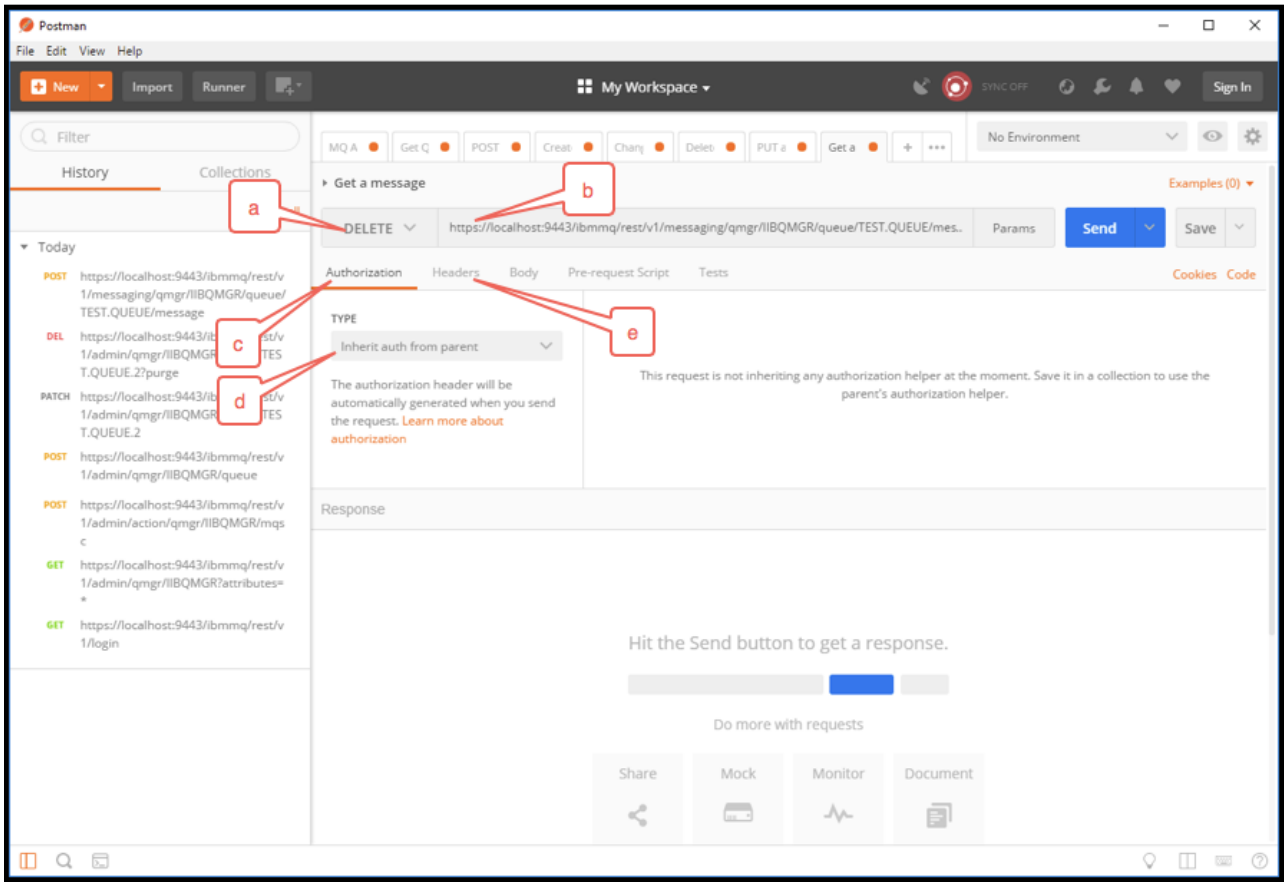



```
https://localhost:9443/ibmmq/rest/v1/messaging/qmgr/IIBQMGR/queue/TEST.QUEUE/message
```

c. Select the **Authorization** tab.

d. Select **Inherit auto from parent** for the Type.

e. Click on the **Headers** tab.

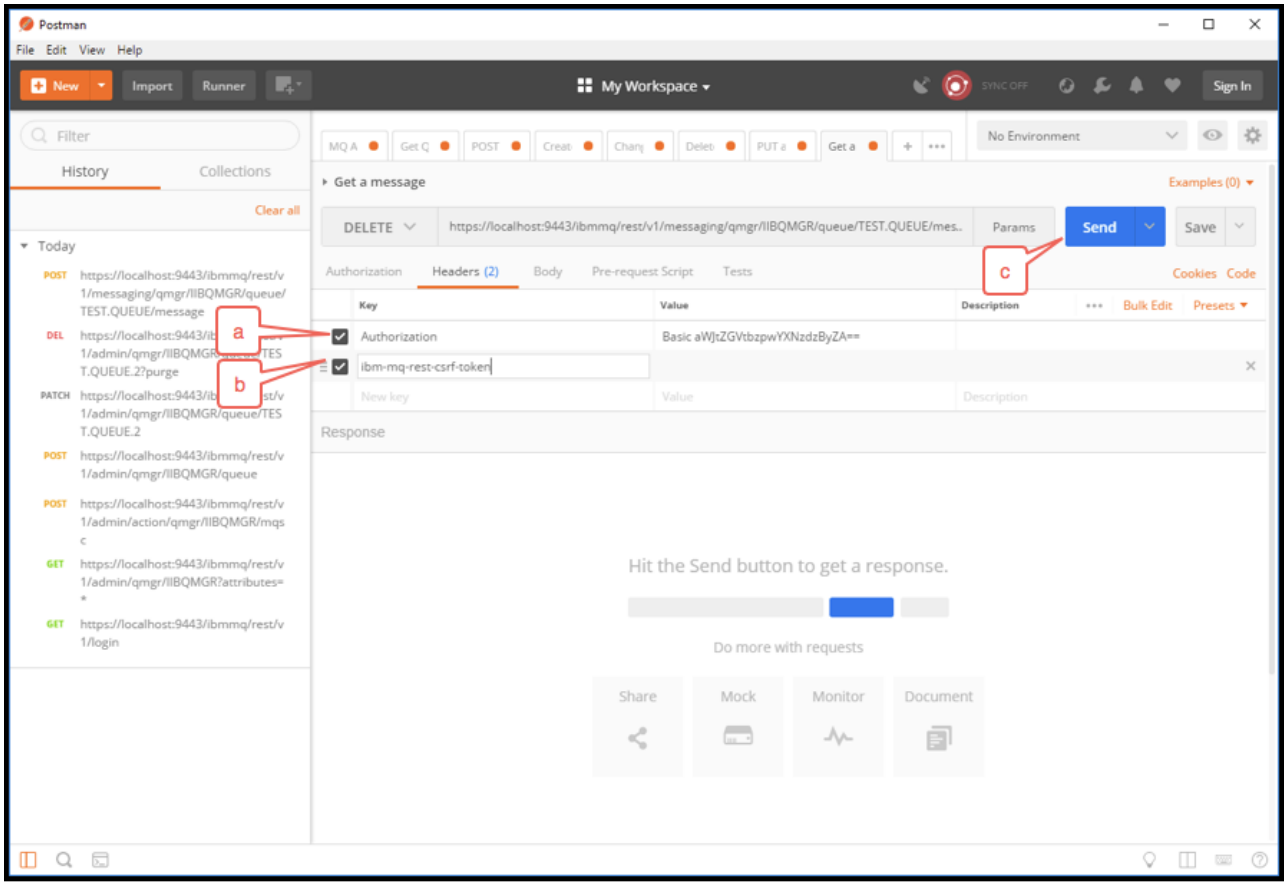


f. Add the following headers:

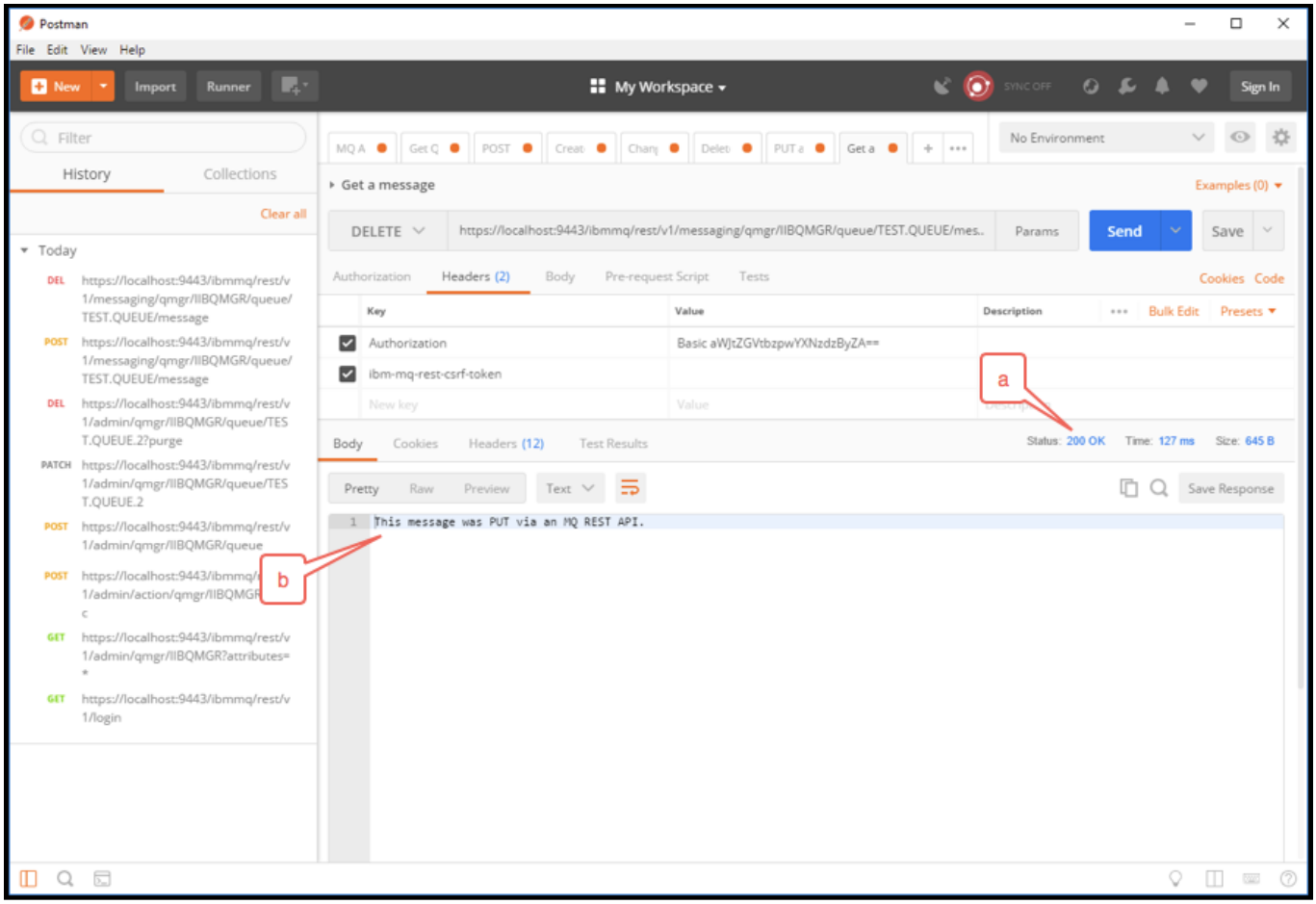
- The **Auhorization** header with the value returned from the Login request.

- The **ibm-mq-rest-csrf-token** header to enable CRSF protection for the MQWeb REST API. Note that it is not necessary to set a value for this header.

g. Click on the **Send** button.



h. You should receive an HTTP status code of **200 OK**. Check the response section and you should see the content of the message.



> **Tip:**
> Try using various options on the **POST** and **DELETE** requests while inspecting the message properties in the MQ Explorer to get a better understanding of how the APIs work.

You have successfully completed this lab.

Summary of using the IBM MQ REST API Interfaces

You have now explored a number of features of the IBM MQ REST API Interfaces. Some of these features include:

- Using an HTTP GET request to get attributes of an IBM MQ queue manager.
- Using an HTTP POST request to send an MQSC message to an IBM MQ queue manager.
- Using an HTTP POST request to create a new queue on an IBM MQ queue manager.
- Using an HTTP PATCH request to change a property of a queue on an IBM MQ queue manager.
- Using an HTTP DELETE request to delete a queue from an IBM MQ queue manager.

- Using an HTTP POST request to **PUT** a message to an IBM MQ message queue.
- Using an HTTP DELETE request to **GET** a message from an IBM MQ message queue.

This concludes Lab 5.

Continue to Lab 6 (mq_basic_pot_lab6.html)

Return MQ Basic Menu (mq_basic_pot_overview.html)

Site last generated: Apr 19, 2019