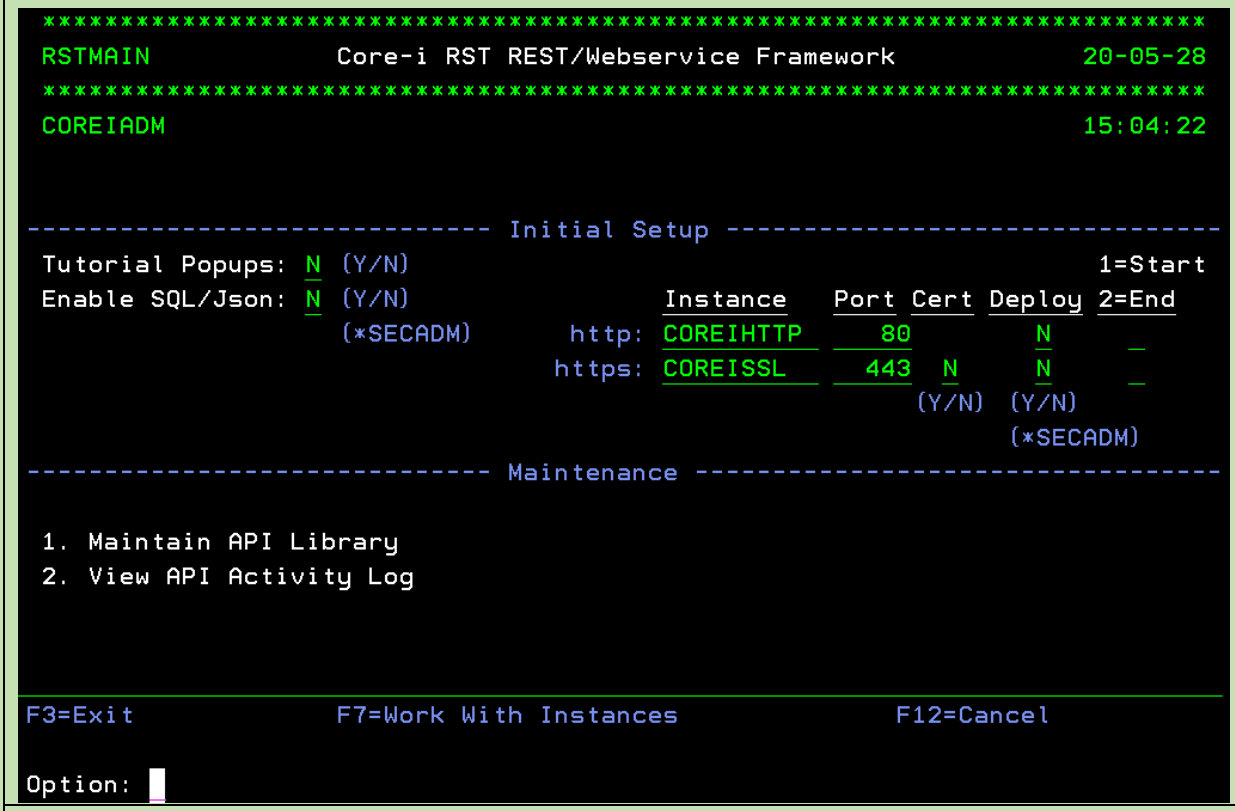
Core-i™ RST

REST/Webservice Framework

(middleware and API engine)

v 1.01 – Created 05/28/2020

CROSS - PLATFORM EXECUTION



Using our sample API's that shipped with Core-iRST, we will consume them cross-platform using Postman as the client side API consumer. Postman is a free 3rd party tool that allows for all types of enterprise development to test API consumption across networks. Basically it is simulating any client computer that is outside of your network that may want to call/consume your API.

- **http – userProfile/password authentication**
  - This option utilizes the COREIHTTP http server instance we deployed earlier in our documentation guides.

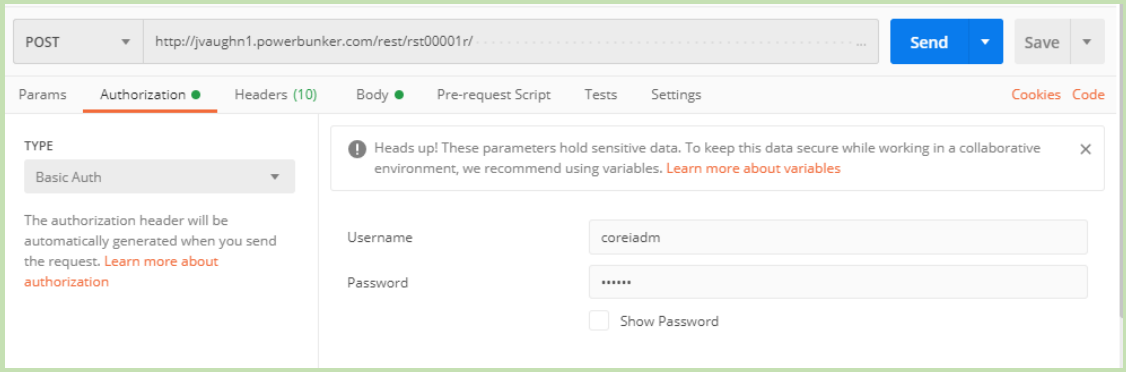
- We will first ensure that http instance is started and running prior to testing, So from the CoreiRST main screen, lets do the F7-Work With Instances and look for the COREIHTTP instance running

```
                          Work with Active Jobs                      JVAUGHN1
                                                          20-05-28  15:05:34 UTC
CPU %:      .0     Elapsed time:   00:00:00     Active jobs:   221
                    Current
Opt  Subsystem/Job  User        Type  CPU %  Function        Status
     QHTTPSVR       QSYS        SBS      .0                 DEQW
       ADMIN        QTMHHTTP    BCH      .0  PGM-QZHBMAIN    SIGW
       ADMIN        QTMHHTTP    BCI      .0  PGM-QZSRLOG     SIGW
       ADMIN        QTMHHTTP    BCI      .0  PGM-QZSRHTTP    SIGW
       ADMIN        JVAUGHN     BCI      .0  PGM-QYUNLANG    TIMW
       ADMIN        JVAUGHN     BCI      .0  PGM-QYUNLANG    TIMW
       ADMIN        JVAUGHN     BCI      .0  PGM-QYUNLANG    TIMW
       ADMIN1       QLWISVR     BCI      .0  JVM-/QIBM/Prod  THDW
       ADMIN2       QLWISVR     BCI      .0  JVM-/QIBM/Prod  THDW
       ADMIN3       QLWISVR     BCI      .0  JVM-/QIBM/Prod  THDW
       ADMIN4       QWEBADMIN   BCI      .0  JVM-/QIBM/Prod  THDW
       ADMIN5       QLWISVR     BCI      .0  JVM-/QIBM/Prod  THDW
       COREIHTTP    QTMHHTTP    BCH      .0  PGM-QZHBMAIN    SIGW
       COREIHTTP    QTMHHTTP    BCI      .0  PGM-QZSRLOG     SIGW
       COREIHTTP    QTMHHTTP    BCI      .0  PGM-QZSRLOG     SIGW
                                                                    More...
===>
F21=Display instructions/keys
```

- Here we see it up and running, ready to go. If we did not see it here, we would simply revert back to the prior main screen and place a “1” in the Start/End input for that COREIHTTP instance, then review with F7 again to be sure it is running.

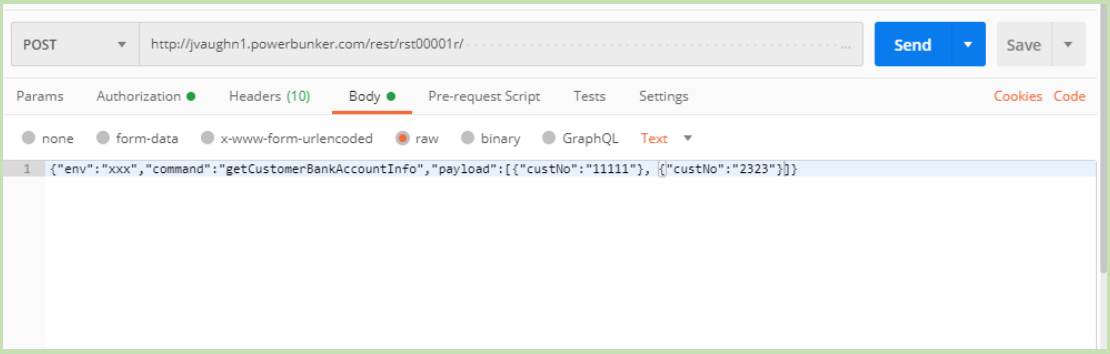
- If you do not have Postman, you can download it for free here.
  - https://www.postman.com/downloads/

- Once you have Postman installed, start it and let's specify a few parameters for the http API test.

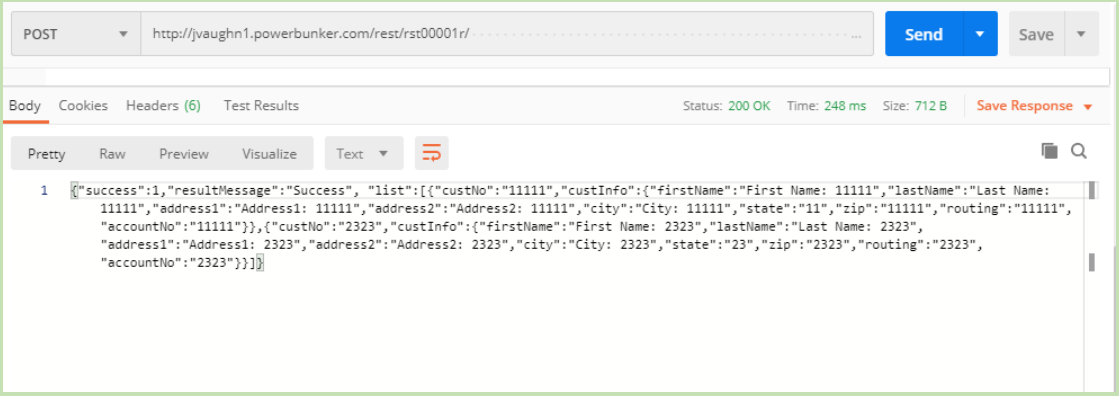


- **Non-REST JSON/XML Request**
  - Method
    - POST
    - NOTE: since we are NOT invoking a REST method, and we are going to be passing the request in JSON/XML format via the http body, we will always choose POST. Technically you can also pass the JSON request via the url after the endpoint, and in that case you would not have to choose POST, but using the body to pass the request is the recommended method.
  - Endpoint
    - This is going to be your IBMi
    - Your Apache HTTP server will handle requests sent to this endpoint.
  - Basic Authentication
    - When the Apache HTTP server detects a request, it will authenticate that request with an IBMi userProfile/password that is passed by the API consumer (in this case Postman utility). The userProfile/password is tokenized meaning it will not be transmitted in clear text representation for security reasons.
    - A suggested user profile for suitable use would be
      - **CRTUSRPRF USRPRF(COREIRST)**
        PASSWORD(<password>)
        USRCLS(*USER)
        INLMNU(*SIGNOFF)
        GRPPRF(QPGMR)
  - Request Body
    - Raw/Text – even though we are passing a JSON/XML request, Corei-RST will work with those formats in a string format.
    - Then of course the body will be the actual JSON request string.

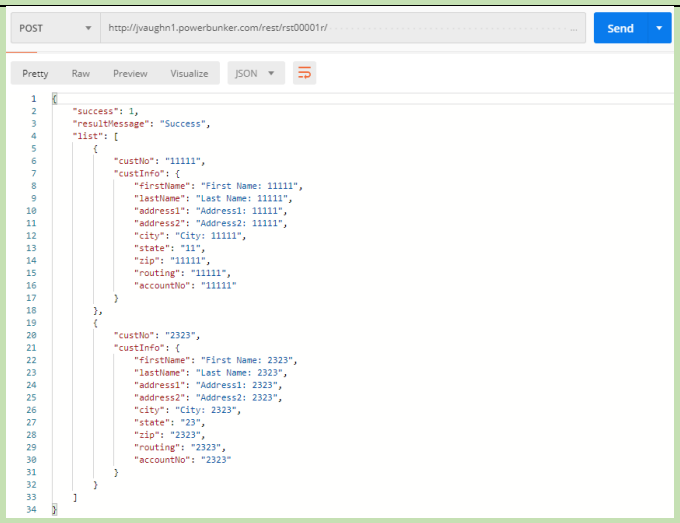
- If you happen to copy/paste it out of the CoreiRST 5250 screen, you may have trouble with its formats as it will be broken in many places with extracted. Instead, use iACS sql scripts and select it out of the control file.
    - select tjsonin
      from coreirst.rst00001t
      where tprogram = 'APIxxxxxxx'
- copy/paste the result from iACS to Postman – this is much cleaner.



- At this point, the request is ready to run. Click Send.
- When the request returns, you can view the response – scroll down a bit.

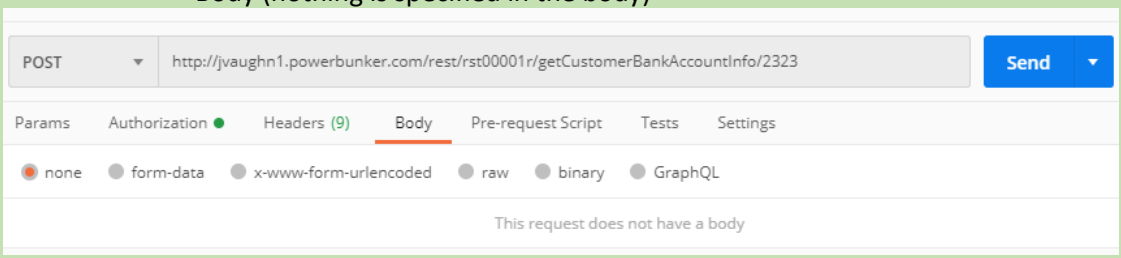


- Looks good… We see the Status 200 ok was returned with a measurement of time it took to return this response. 248ms
- Let's view the response in true json (pretty) format. On the drop down labeled Text, find JSON and click on that.

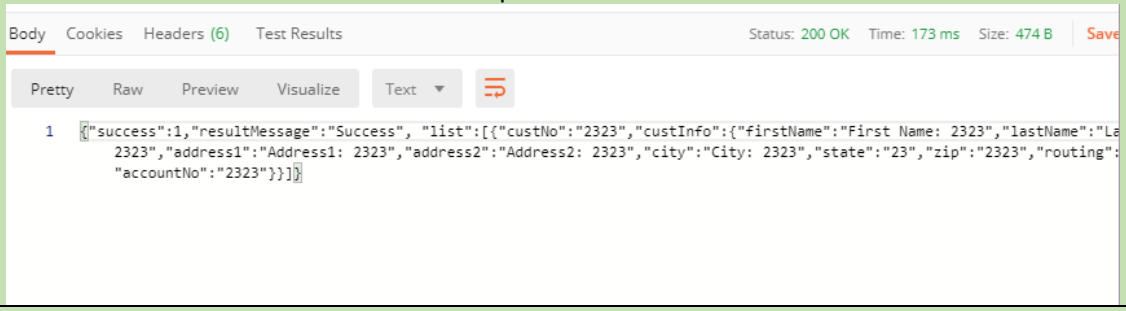


- Looks even better.

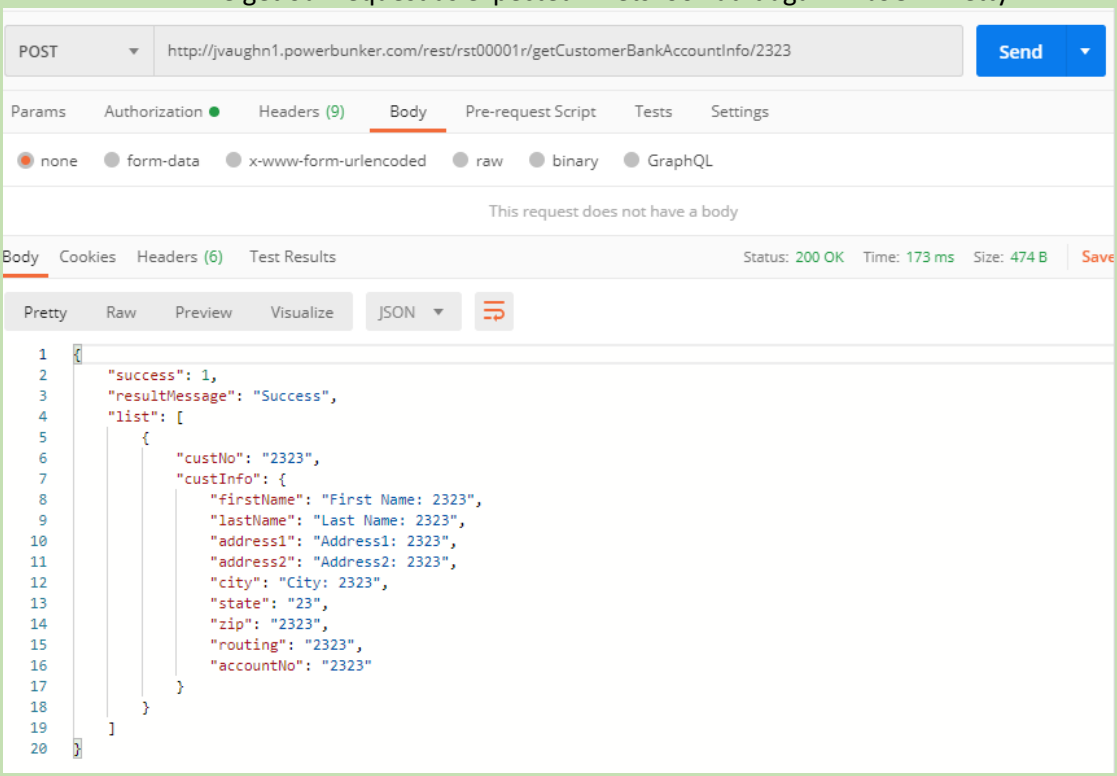
- **REST request**
  - Method
    - This is the action or *"verb"* in true RESTful API definition.
    - CoreiRST is not picky about the method being set to anything particular. The CoreiRST can detect the REST method, but the bottom line is that CoreiRST relies on the command being passed to dictate which backend API library program gets called.
    - So for our REST request demonstration, this can be set to anything.
  - Endpoint
    - This is the same as the JSON/XML request example, however with REST methodology, the request input parameters are defined after the endpoint, along with the command.
  - Basic Authentication (same)
  - Body (nothing is specified in the body)



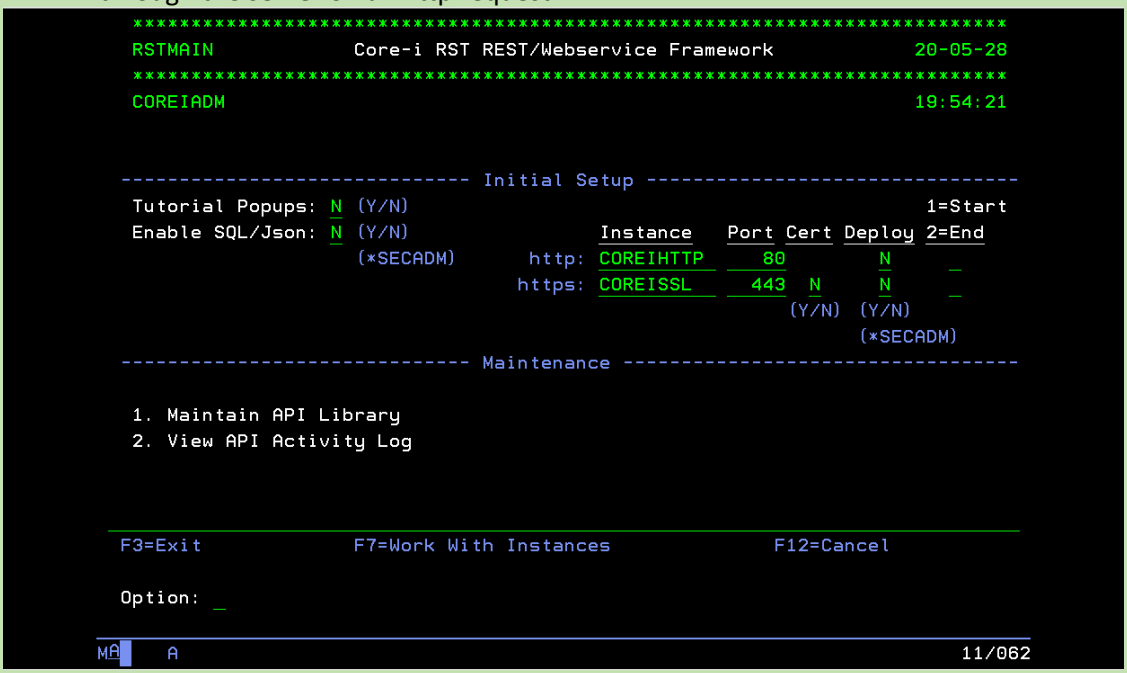
- Here is what the REST request will look like – Click Send.



- We get our request as expected – Lets look at it again in JSON Pretty



- **https – TLS encrypted with certificate Request**
    - This http method is the most secure and what you should absolutely consider when allowing resources outside of your organization's VPN (ie. Firewall), access your IBMi backend API's.
    - First, let's ensure that our COREISSL server instance we deployed has "Y" for Cert. This will instruct Apache http server to require a certificate authentication to pass through the server on an http request.



    - From the main Corei-RST screen, F7=Work With Instances – ensure there are not COREISSL (or your own named SSL) instances running.
    - With *SECADM privileges be sure the https Cert and Deploy are set to "Y" and press enter.

- 1.) Using IBMi DCM, create server certificate...
  - http://<yourIBMiServer.com>:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
    - NOTE: sometimes you have to close DCM and go back in for it to work right!
    - setup *SYSTEM certificate store (or be sure exists)
      - if creating (create a new certificate store)
      - no - do not create a certificate in the certificate store
      - set store password
      - NOTE: it may be best to close DCM and start it back up after creation.
    - create certificate authority (CA)

- SHA-256
- password - corei
- CA name - <server name> (ie. yourIBMiServer.com)
- organization - corei
- days - 7300
- NOTE: it may be best to close DCM and start it back up after creation.

- select *SYSTEM cert store
- create certificate
- server or client certificate
- local certificate authority (CA)
- certificate label - coreissl certificate
- common name - <server name> (ie. jvaughn1.powerbunker.com
- issuer - xyz (must be different that CA organization)

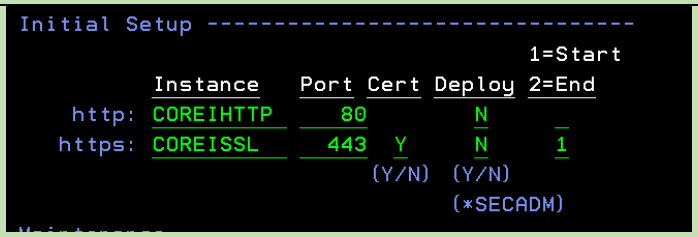
- Manage Applications> Add Application
- server
- Application ID = QIBM_HTTP_SERVER_COREISSL
- Application description = CoreiRST COREISSL Server
- Exit Program = QZHBDCME
- Exit Program Library = QHTTPSVR
- Threadsafe = Not known
- Multithread Job Action = Run program and send message
- Client authentication required = No
- Define the CA trust list = Yes
- Certificate Revocation List (CRL) checking = No
- accept all other defaults and click add
- assign certificate to the qbim_http_server_coreissl server application

- NOTE: exit DCM
- start server instance
- can test with ssllabs.com/ssltest

2.) export certificate to IFS (with password)
- select certificate store *system
- manage certificates
- export certificate
- server or client
- choose certificate - select file (not certificate store)
- export to filename: /tmp/<server name>.pfx (.pfx very important)
- password - whatever

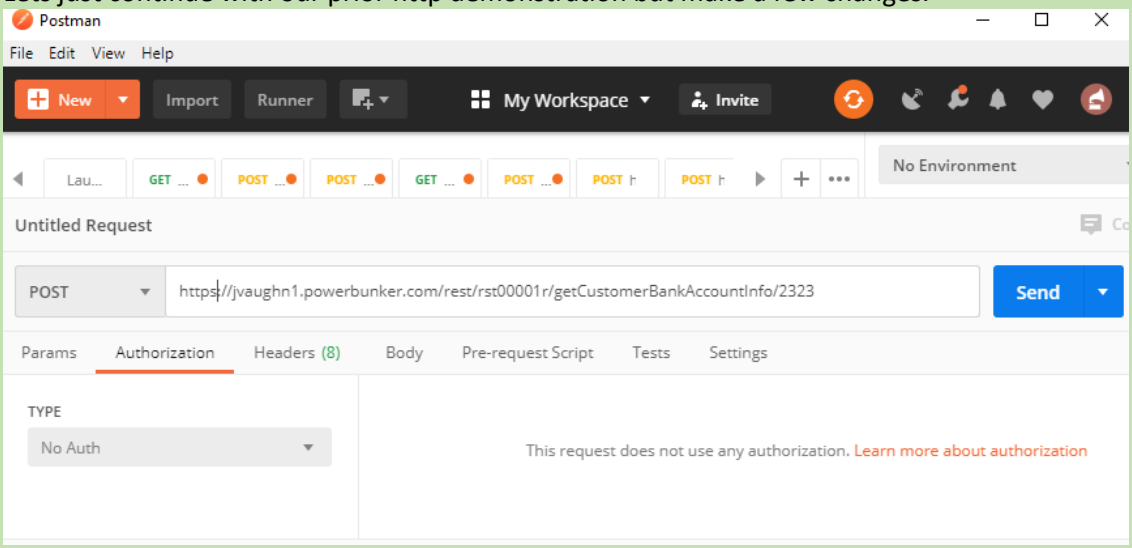
3.) use this file with postman or any client web app.

- With the certificate created and attached to the Apache https instance, we will now take the .PFX file that was exported and use this with ANY client we want to allow pass the server to consume one of our CoreiRST API's.
- But before we test with Postman, lets return back to the main CoreiRST screen to be sure the https server instance is running now that the certificate is attached to the server.
- Specify option 1 for Start/End to start the https server instance. Use F7=Work With Instances to ensure the server instance started all jobs and is maintaining activity.

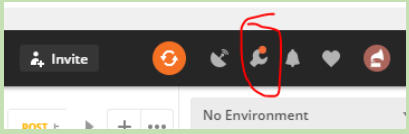


- Ok, lets go back to Postman now.

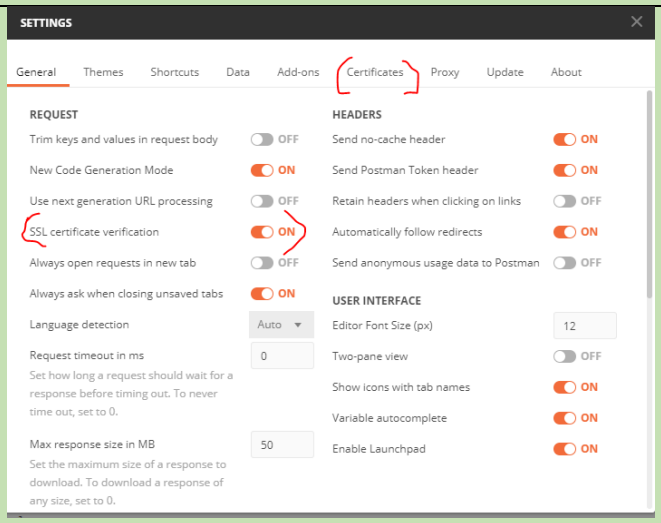
- Lets just continue with our prior http demonstration but make a few changes.



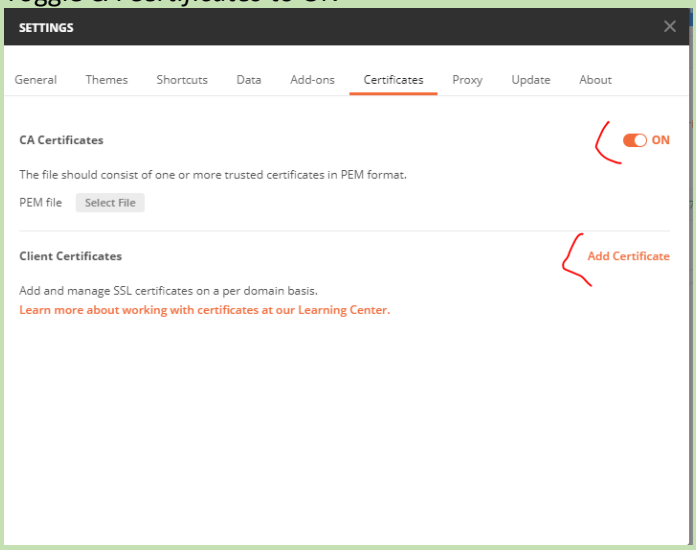
- Our first tweak is to change the http to https in the endpoint. We are now using an encrypted protocol which is secure. For details on what level of encryption will be used, refer to the *CoreiRST_Introduction_InitialSetup* document under https.
- Secondly we will specify No Auth under Authorization.
  - For authorization we will utilize the certificate which Postman supports in it's testing. Click the wrench head in the upper right corner of the screen above which represents settings in Postman.



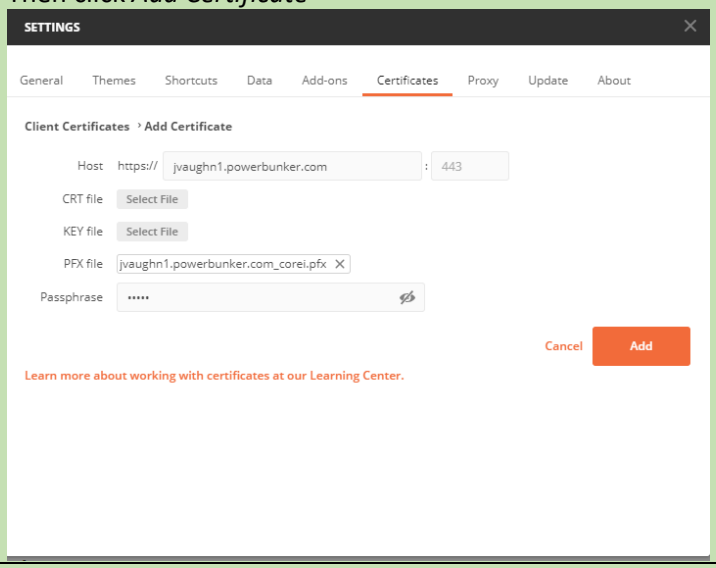
  - Click the wrench, then click settings.
  - Enable *SSL Certificate Verification*



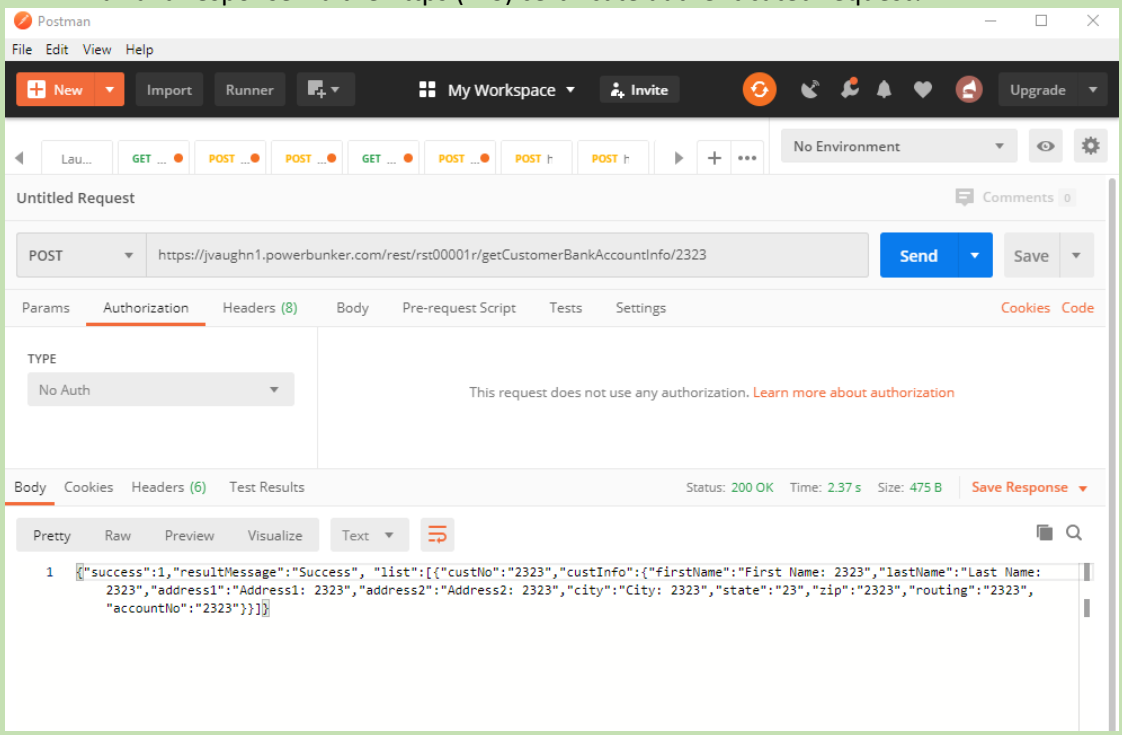
- Then click *Certificates* at the top.
- Toggle *CA Certificates* to ON



- Then click *Add Certificate*



- Specify the following as illustrated above.
    - YOUR IBMi endpoint name (and port)
    - YOUR .PFX file that you exported from IBM DCM, along with the associated passphrase.
    - Click Add
- Return back and click Send – everything should be setup correctly and we should get a valid response via the https (TLS) certificate authenticated request!



End Of Document