Core-i™ RST

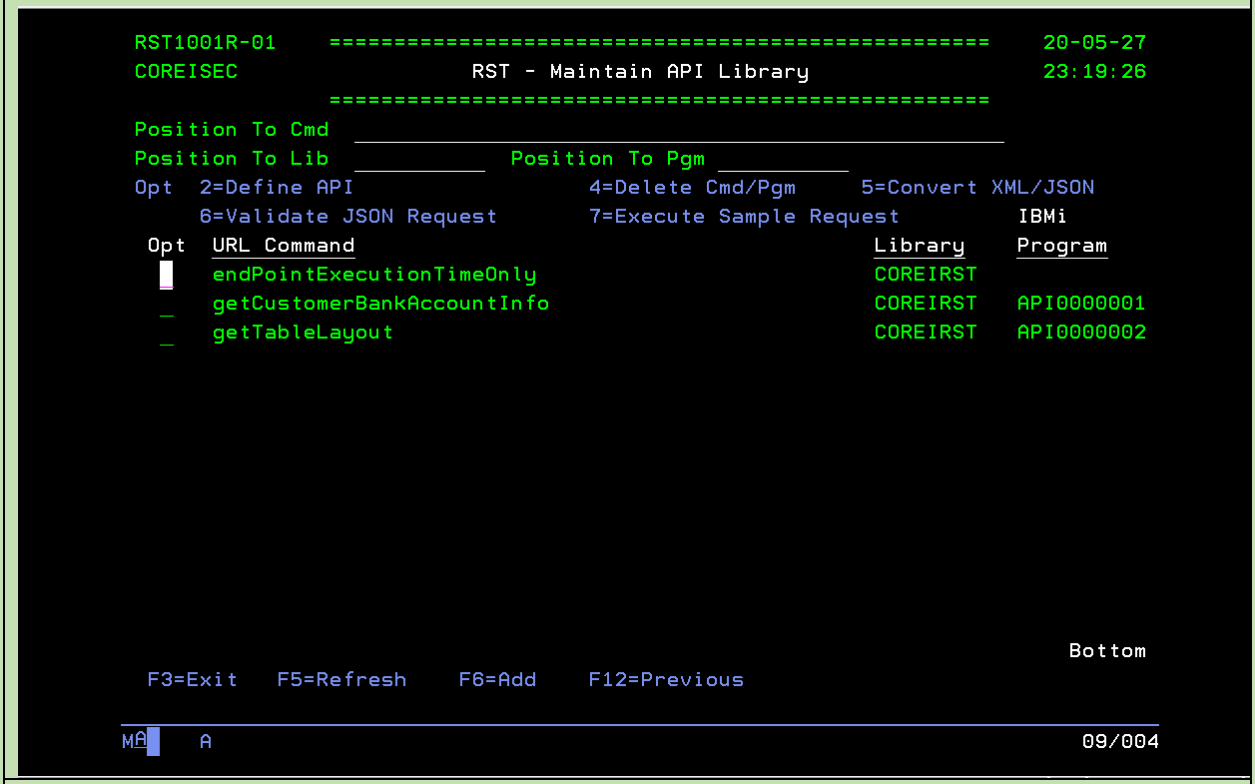
REST/Webservice Framework

(middleware and API engine)

v 1.01 – Created 05/28/2020

OPTION 1. MAINTAIN API LIBRARY

User Options



- **2=Define API**
  - Refer to the document *Corei_RST_MaintainAPILibrary* document as the F6=Add to create an API covers the same functionality as 2=Define API

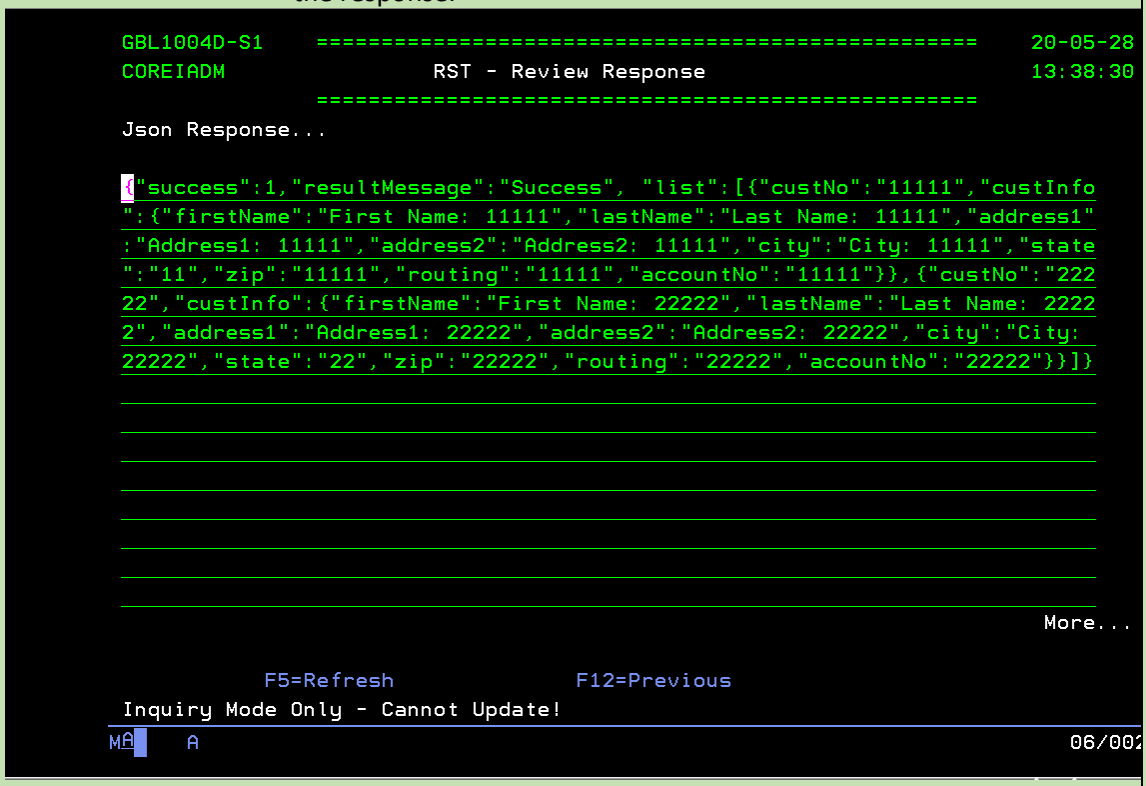
- **4=Delete Cmd/Pgm**
  - This option will simply remove the command/library/program from the control file.
  - Any corresponding source will remain in COREIRST/COREIRST_S.
  - Any corresponding *PGM object will remain in COREIRST.

- **5=Convert XML/JSON**
  - This option will toggle the defined JSON/XML request structure between XML or JSON.

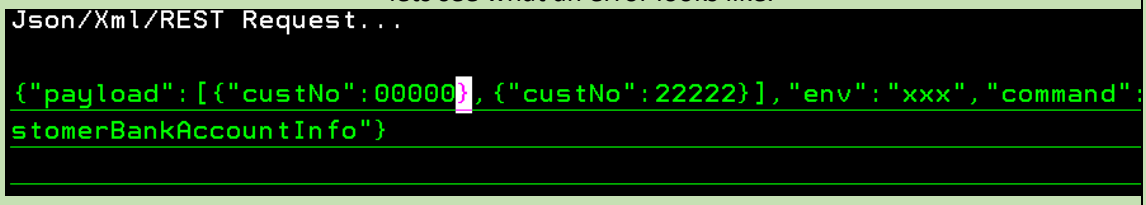
- **6=Validate JSON Request**
  - As we use the 2=Define to edit our JSON request, we should always proceed with using this option to validate and re-ensure the JSON format remains valid.
  - Choosing 6 for any API in the list, will simply evaluate the defined JSON request for it and determine if it is valid.
  - This option relies on the PTF requirements presented in the *CoreiRST_Introduction_InitialSetup* document, as it relies on the sql/json functions provided by IBM.
  - Outputs
    - *Json Request Is Valid*
    - *Invalid JSON Request - Select With Option 2 To Review/Edit*

- **7=Execute Sample Request**
  - Option 7 allows you to invoke your API server side, an allow it to run with the JSON/XML/REST request that you have specified with the *Maintain API Library* process.
  - When invoked, the response will be returned, just as it would from an off-platform http request.

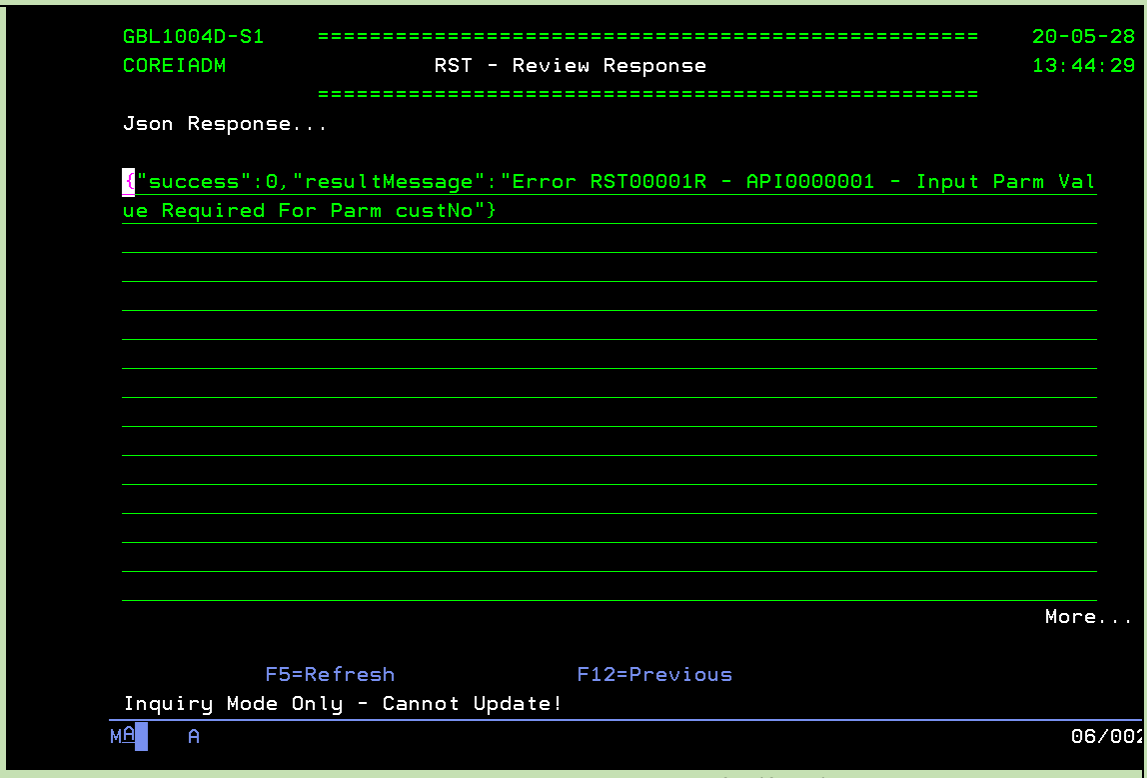
- **JSON Request – JSON Response**
  - When we choose 7=Execute Sample Request for getCustomerBankAccountInfo we are immediately presented with the response.



  - Here we can see that we had a successful response returned from the defined request.
  - Most front end developers like the option to simply check a leading Boolean telling them whether there was a successful response returned or not.
  - Let's alter the request now, specifying 0 for one of the customer numbers. We know that the definitions are specified to be sure a value exists for customer number, so lets see what an error looks like.



  - Then re-run option 7=Execute Sample Request



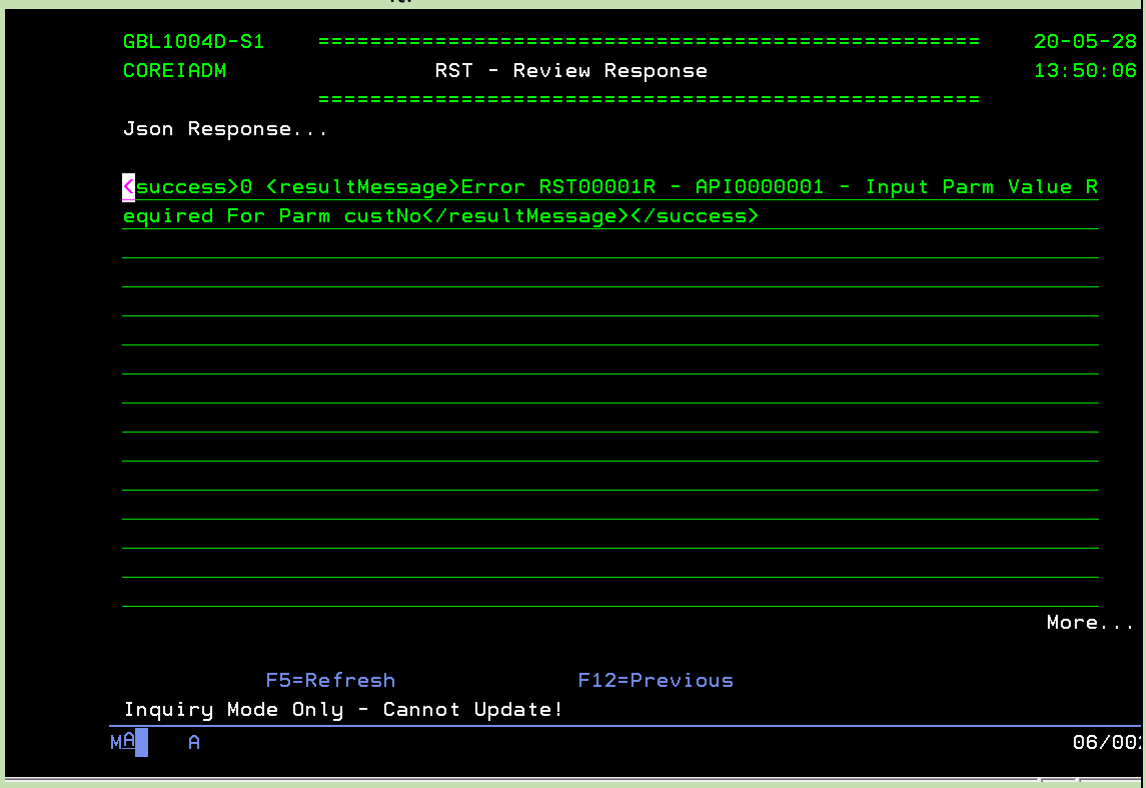
-
    - We get a success Boolean of 0 (false) and we return to the consumer the JSON response error that we expected, and that is found in our auto-API definitions.

- **XML Request – XML Response**

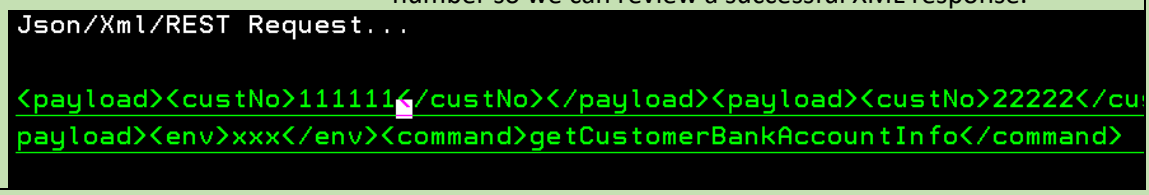
```
RST1001R-01    ================================================    20-05-28
COREIADM                RST - Maintain API Library                 13:46:03
               ================================================

Position To Cmd  ____________________________________________________
Position To Lib  __________  Position To Pgm  __________
Opt  2=Define API                4=Delete Cmd/Pgm     5=Convert XML/JSON
     6=Validate JSON Request     7=Execute Sample Request         IBMi
 Opt  URL Command                                    Library    Program
      endPointExecutionTimeOnly                      COREIRST
  5   getCustomerBankAccountInfo                     COREIRST   API0000001
      getTableLayout                                 COREIRST   API0000002

                                                                    Bottom
  F3=Exit    F5=Refresh    F6=Add    F12=Previous

MA  A                                                          11/004
```

```
GBL1004D-S1    ================================================    20-05-28
COREIADM               RST - Maintain Request Example              13:47:40
               ================================================
Json/Xml/REST Request...

<payload><custNo>0</custNo></payload><payload><custNo>22222</custNo></paylo
ad><env>xxx</env><command>getCustomerBankAccountInfo</command>
                                                                    More...

          F5=Refresh             F12=Previous

MA  A                                                          06/002
```

- NOTE: The JSON/XML converter does utilize JAVA so the initial invocation is a bit delayed. Subsequent requests, are much more responsive.
- We now see that we have a converted XML request. Lets run it.



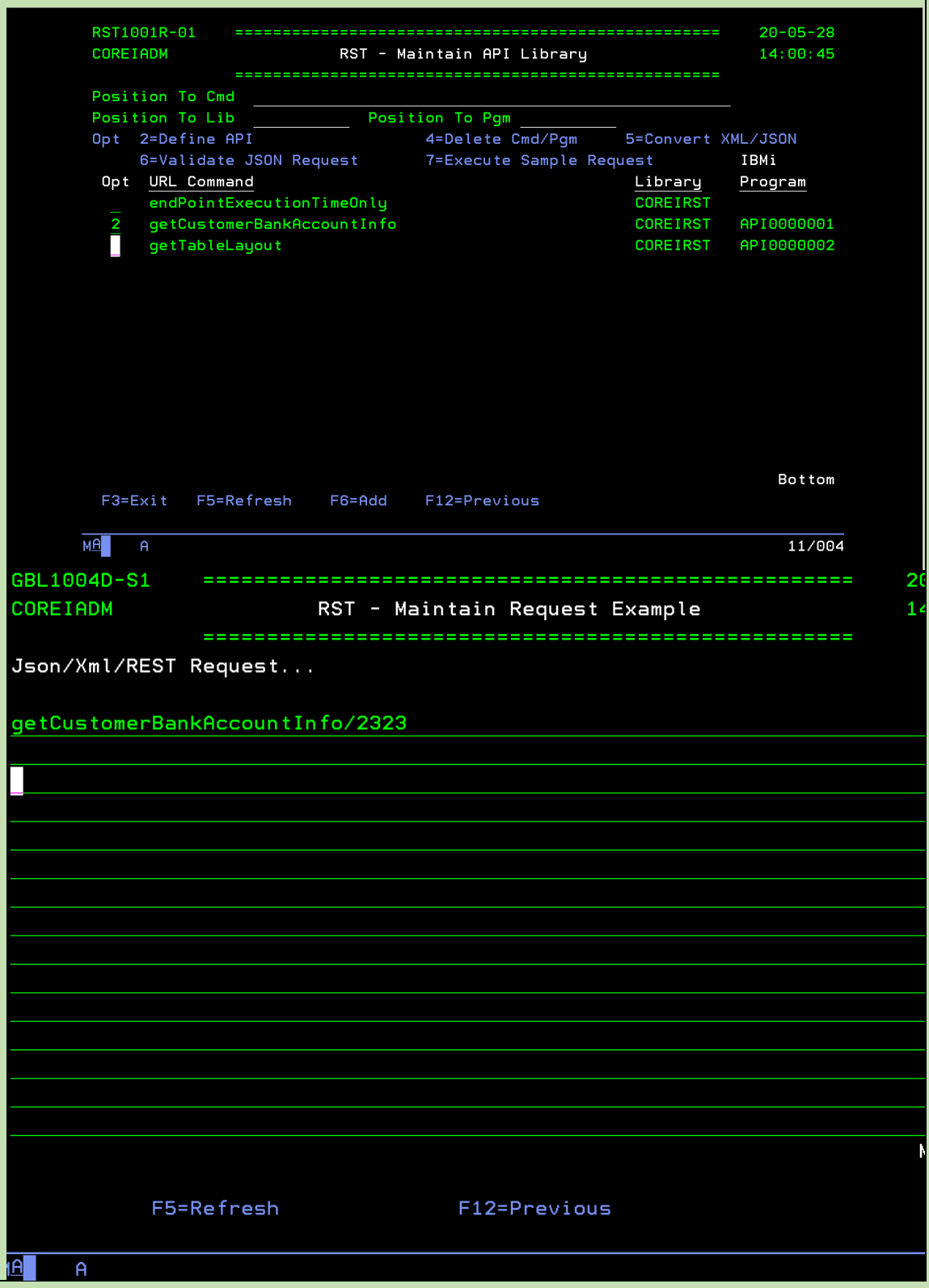
- We now have a valid XML response. CoreiRST knows that since the request was XML, it needs to produce an XML response.
- Let's fix our request input value to specify a valid customer number so we can review a successful XML response.



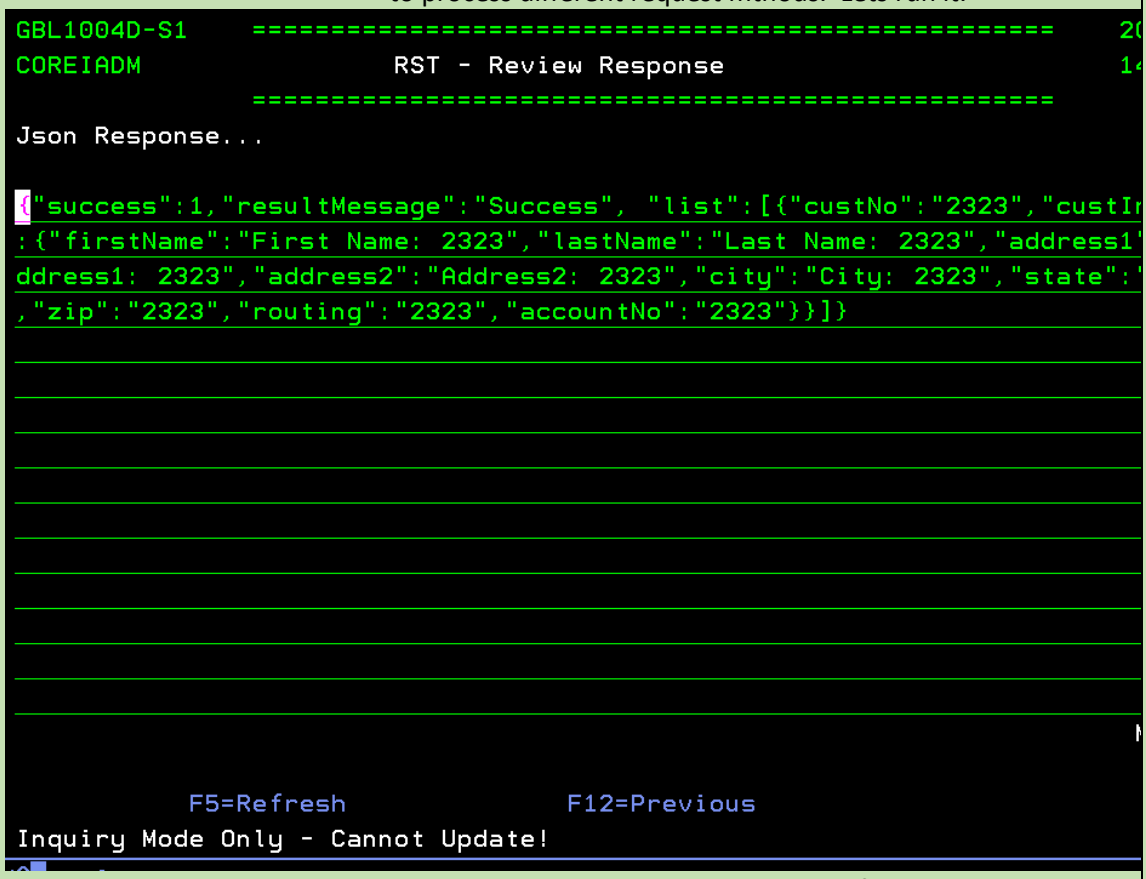
```
GBL1004D-S1    ======================================================    2
COREIADM                  RST - Review Response                          1
               ======================================================
Json Response...

<success>1 <resultMessage>Success</resultMessage><list><custInfo><zip>1
</zip><firstName>First Name: 111111</firstName><lastName>Last Name: 111
/lastName><routing>111111</routing><address2>Address2: 111111</address2
ty>City: 111111</city><address1>Address1: 111111</address1><accountNo>1
1</accountNo><state>11</state></custInfo><custNo>111111</custNo></list>
t><custInfo><zip>22222</zip><firstName>First Name: 22222</firstName><la
me>Last Name: 22222</lastName><routing>22222</routing><address2>Address
2222</address2><city>City: 22222</city><address1>Address1: 22222</addre
<accountNo>22222</accountNo><state>22</state></custInfo><custNo>22222</
No></list></success>
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________



          F5=Refresh              F12=Previous
Inquiry Mode Only - Cannot Update!
```

- We have a successful XML response.

- **REST Request – JSON Response**

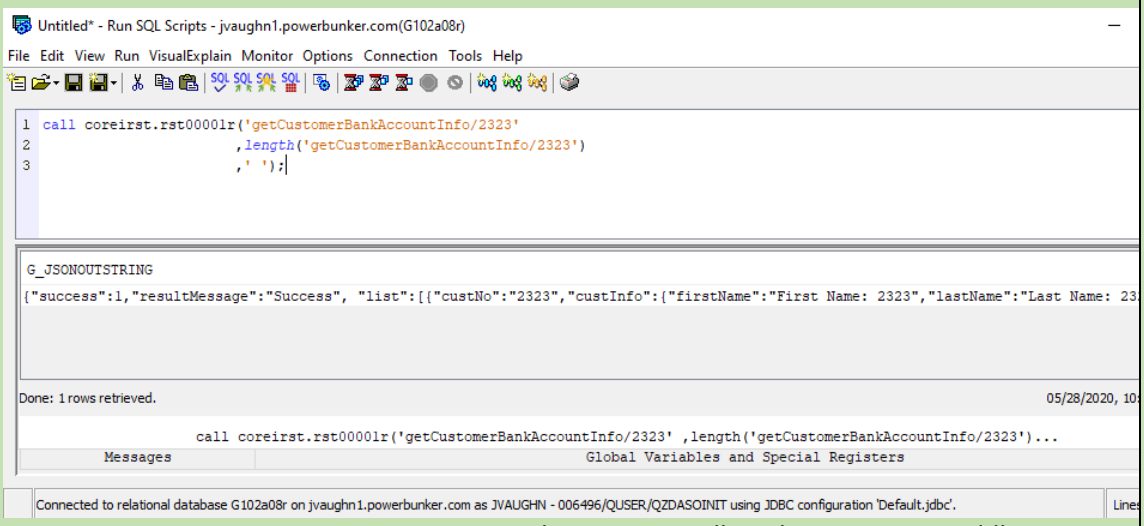


- The request has now been altered to specify a REST formatted request. After the above screen you will be presented with the Maintain API Anatomy screen of auto-API definitions – you can F12=Return on this screen as nothing needs to change. Corei-RST created the API with the ability to process different request mthods. Lets run it.



- We get the request we were expecting for customer number 2323.

- **Call API as an SQL Stored Procedure**



- Here we see how we are calling the Core-iRST middleware program and passing the parms it expects in order to execute this API as an sql stored procedure
- The response is as expected and returned to us as an sql result set.

Summary

With these execution examples you should now be familiar with the different ways a Core-iRST API program can be invoked on the server side without even using http. This is great for testing your API to ensure it is functioning exactly how it is expected before you add in another layer of complexity which in this case is the http cross platform process.

We will cover cross-platform API testing in another document called CoreiRST_APICrossPlatform.

End Of Document