

- **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, and 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.

```
GBL1004D-S1 ===== 20-05-28
COREIADM RST - Review Response 13:38:30
=====
Json Response...

{"success":1,"resultMessage":"Success", "list":[{"custNo":"11111","custInfo":{"firstName":"First Name: 11111","lastName":"Last Name: 11111","address1":"Address1: 11111","address2":"Address2: 11111","city":"City: 11111","state":"11","zip":"11111","routing":"11111","accountNo":"11111"}},{"custNo":"22222","custInfo":{"firstName":"First Name: 22222","lastName":"Last Name: 22222","address1":"Address1: 22222","address2":"Address2: 22222","city":"City: 22222","state":"22","zip":"22222","routing":"22222","accountNo":"22222"}}]}

More...

F5=Refresh F12=Previous
Inquiry Mode Only - Cannot Update!
MA A 06/002
```

- 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.

```
Json/XML/REST Request...

{"payload":[{"custNo":00000}, {"custNo":22222}], "env":"xxx", "command":"stomerBankAccountInfo"}
```

- Then re-run option 7=Execute Sample Request

```
GBL1004D-S1 ===== 20-05-28
COREIADM          RST - Review Response 13:44:29
=====
```

Json Response...

```
{"success":0,"resultMessage":"Error RST00001R - API0000001 - Input Parm Value Required For Parm custNo"}
```

More...

F5=Refresh

F12=Previous

Inquiry Mode Only - Cannot Update!

MA A

06/002

- 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
    -----
    5  endPointExecutionTimeOnly  COREIRST
    5  getCustomerBankAccountInfo  COREIRST  API0000001
    5  getTableLayout              COREIRST  API0000002
    
```

Bottom

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

MÂ 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></payload>
<env>xxx</env><command>getCustomerBankAccountInfo</command>

```

More...

F5=Refresh F12=Previous

MÂ A 06/004

- 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.

```

GBL1004D-S1  ===== 20-05-28
COREIADM          RST - Review Response 13:50:06
=====

Json Response...

<success>0 <resultMessage>Error RST00001R - API0000001 - Input Parm Value R
required For Parm custNo</resultMessage></success>

More...

F5=Refresh          F12=Previous
Inquiry Mode Only - Cannot Update!
MA A 06/00:

```

- 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.

```

Json/XML/REST Request...

<payload><custNo>111111</custNo></payload><payload><custNo>22222</cu:
payload><env>xxx</env><command>getCustomerBankAccountInfo</command>

```

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

Json Response...

```
<success>1 <resultMessage>Success</resultMessage><list><custInfo><zip>1
</zip><firstName>First Name: 11111</firstName><lastName>Last Name: 111
</lastName><routing>11111</routing><address2>Address2: 11111</address2
ty>City: 11111</city><address1>Address1: 11111</address1><accountNo>1
1</accountNo><state>11</state></custInfo><custNo>11111</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</adre
<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

```

RST1001R-01  ===== 20-05-28
COREIADM          RST - Maintain API Library 14:00:45
=====

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
    2  getCustomerBankAccountInfo  COREIRST  API0000001
    |  getTableLayout              COREIRST  API0000002

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

MA  A 11/004
GBL1004D-S1 ===== 20
COREIADM          RST - Maintain Request Example 14
=====
Json/XML/REST Request...

getCustomerBankAccountInfo/2323

F5=Refresh  F12=Previous

MA  A

```


- 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 methods. Lets run it.

```
GBL1004D-S1 ===== 20
COREIADM          RST - Review Response 14
=====

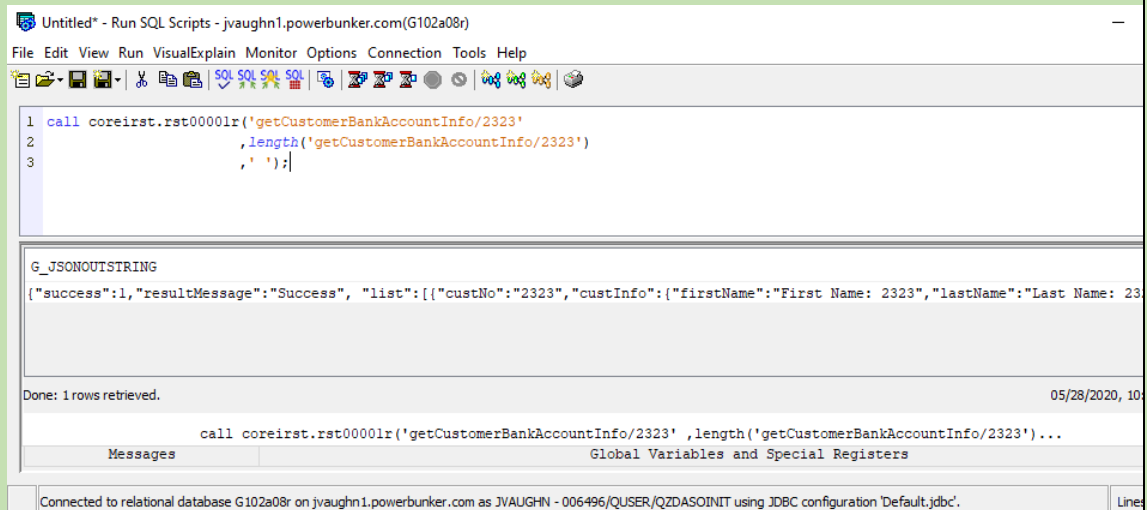
Json Response...

{"success":1,"resultMessage":"Success", "list":[{"custNo":"2323","custIn
:{"firstName":"First Name: 2323","lastName":"Last Name: 2323","address1'
ddress1: 2323","address2":"Address2: 2323","city":"City: 2323","state":'
,"zip":"2323","routing":"2323","accountNo":"2323"}}]}

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

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

▪ Call API as an SQL Stored Procedure



The screenshot shows a SQL client window titled "Untitled* - Run SQL Scripts - jvaughn1.powerbunker.com(G102a08r)". The window contains a SQL script with three lines of code:

```
1 call coreirst.rst00001r('getCustomerBankAccountInfo/2323'  
2     ,length('getCustomerBankAccountInfo/2323')  
3     , ' ');
```

Below the script, the results pane shows a JSON string:

```
G_JSONOUISTRING  
{"success":1,"resultMessage":"Success", "list":[{"custNo":"2323","custInfo":{"firstName":"First Name: 2323","lastName":"Last Name: 2323"}}
```

The status bar at the bottom indicates "Done: 1 rows retrieved." and "05/28/2020, 10:44:14 AM". The bottom-most status bar shows "Connected to relational database G102a08r on jvaughn1.powerbunker.com as JVAUGHN - 006496/QUSER/QZDASOINIT using JDBC configuration 'Default.jdbc'."

- - 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