# Consuming APIs / JSON from external sources

# JSON stands for JavaScript Object Notation – it allows us to represent state of objects and send it via stateless protocol such as HTTP and then retrieve data carried over by deparsing it in literally any programming language.

# *Instead of using ‘Web services – Consumer proxy’, APIs can be consumed in SAP which is faster than XML.*

Why you should consider moving from XML to JSON standard?

XML is a markup language (extensible Markup Language). It enables us to transfer documents very efficiently (text data and its formatting) via many protocols and keep document shape the same all the way. But there’s a downside of XML – huge amount of markups that are transferred together with data we really need.

Here’s an XML example:

<Document>

<Name>

Document\_Name

</Name>

<Content>

Document\_content\_here

</Content>

</Document>

And here’s its JSON counterpart:

{

"Name" : "Document\_Name",

"Content" : "Document\_content\_here"

}

Even with such simple example we can see that JSON is clearly shorter, because JSONs are constructed as key-value pairs.

Shorter content sent via the internet = faster download.

Suppose for FTTX installation, we need to make payments for Technicians. For the Payment, details of Technician (such as Name, address, Sender bank details, technician bank A/c No., IFSC Code etc.) needs to be sent bank via middleware system and Route is ***SAP → SECO System → Bank***.

For this we will be sending Technician data in JSON format to SECO System and SECO will route the data to Bank. After processing the request by Bank it will send the response to SECO and SECO will route the response to SAP.

For such type of requirement, we need to get the Technician details which needs to be sent to SECO. Also, SECO APIs is required with URL and PORT No., which will receive the data. APIs can have headers also, these header information will be provided by SECO.

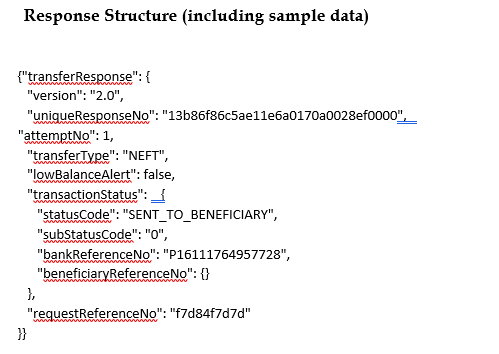
Once we get the API, it can be tested by POSTMAN (Google chrome extension).

Request and Response format will be provided by third party team. Based on the format we need to create structure / deep structure. Without creating structure, by concatenating the field name and values (key-value pairs) also we can create JSON format. We only need to map the data to request / response structure for sending and receiving information.

Sample Request:



Sample Response:



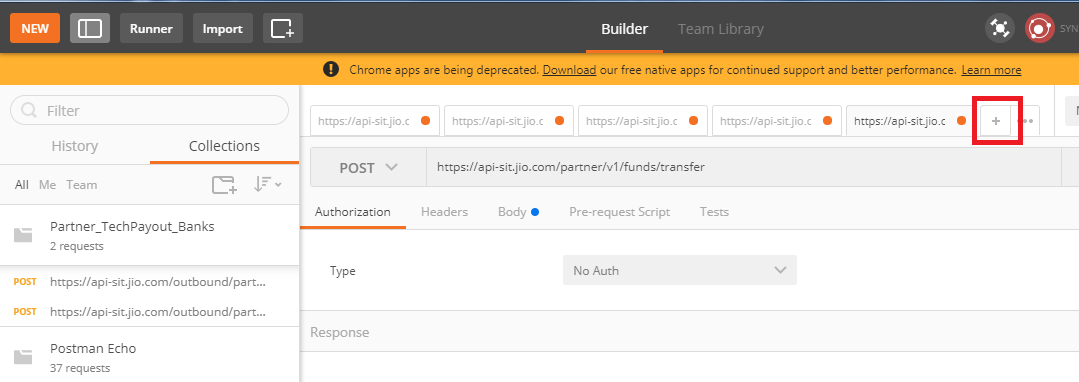
API: <https://api-sit.jio.com/partner/v1/funds/transfer>

Headers:

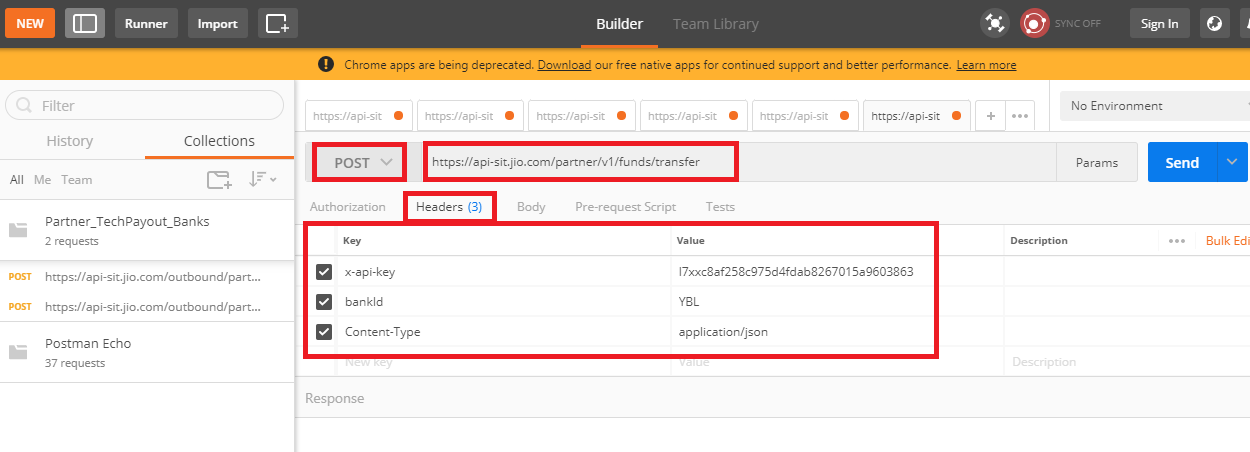
|  |  |
| --- | --- |
| x-api-key | l7xxc8af258c975d4fdab8267015a9603863 |
| bankId | YBL |
| Content-Type | application/json |

**Test the API in Postman:**

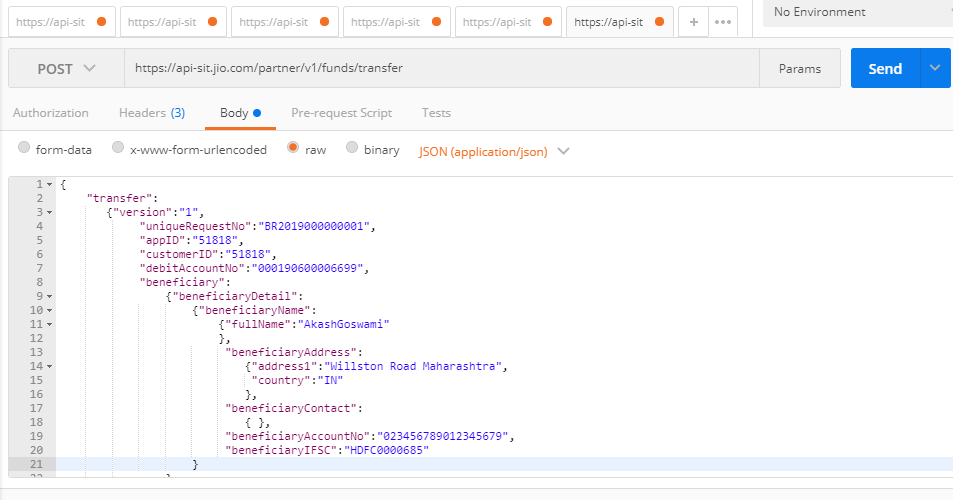
Open Postman →Create New



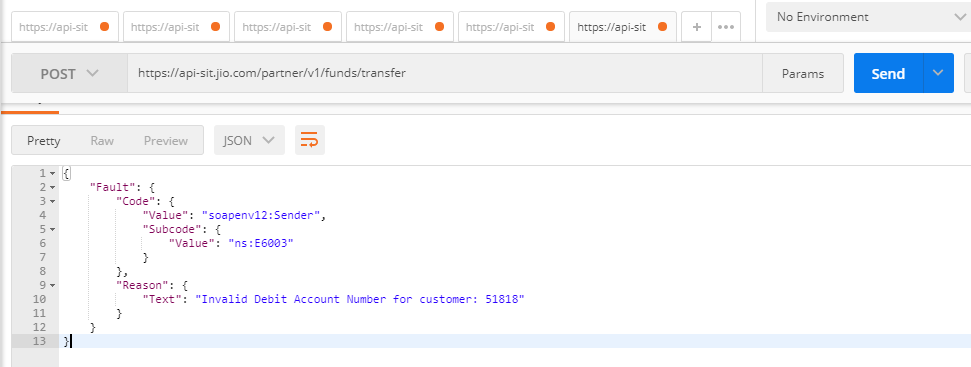
Select Desired Method →Enter API → Click on Header tab → Enter required headers



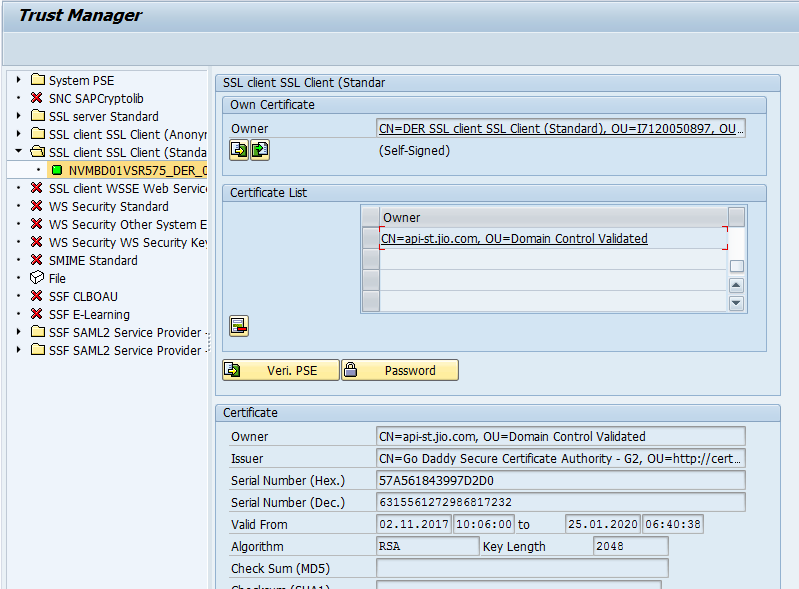
Click on Body Tab → Map the values → Click on Send



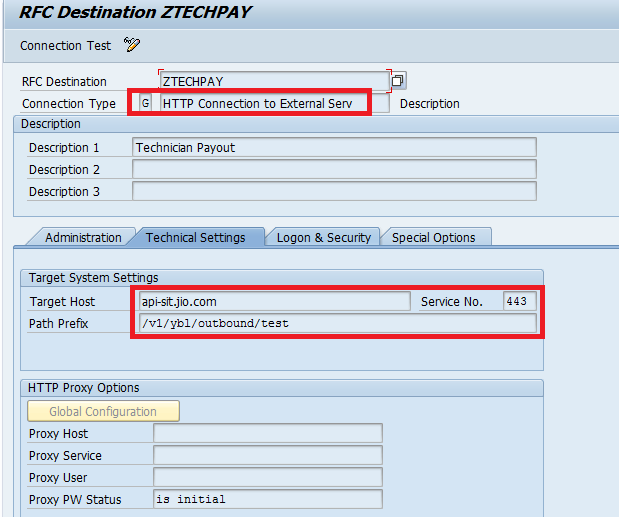
Got below Response from Bank.

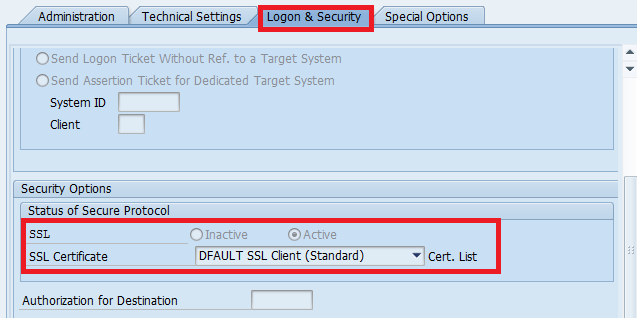


For HTTPS APIs, Certificates will be provided by 3rd party team (in this scenario, SECO Team) and this certificate needs to be installed in TCode: **STRUST**. It should be done by BASIS Team.

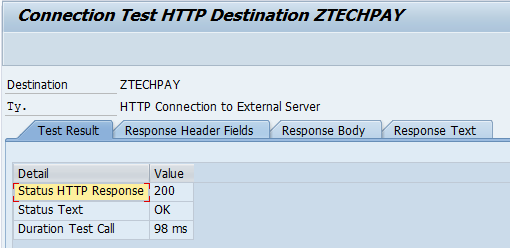


Connection from SAP Server to API link can be checked by creating RFC of type ‘G’ HTTP Connection to External Server. DER- 244 → SM59 → ZTECHPAY

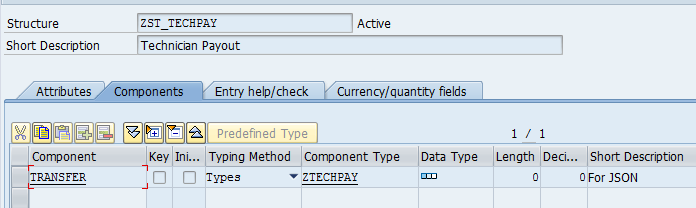


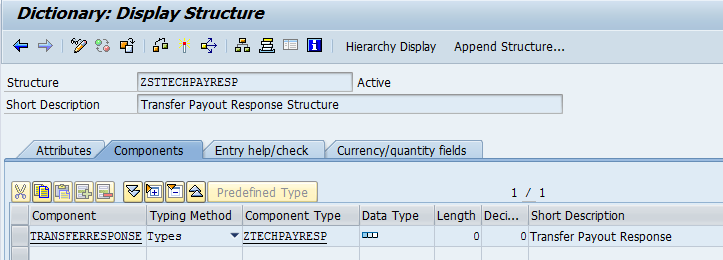


Click on ‘Connection Test’, if connection is not there then connect to BASIS Team.



Now, we need to create Structure for Request and Response format. DER – 244 → SE11 → ZST\_TECHPAY.





In Program, need to populate the Request Format with the desired values. Define a work area.

DATA : gw\_wsdl\_out TYPE zst\_techpay.

Object reference for IF\_HTTP\_Client.

DATA : oref\_http\_client TYPE REF TO if\_http\_client.

Define variable for API URL.

data : i\_url type string, “API URL

gv\_header\_name type string,

i\_authkey type string, “Header data

i\_bankid   type string, “Header data

gv\_json        TYPE string, “JSON Data

gv\_result   TYPE string, “To capture the response

E\_JSON\_IN  TYPE ZSTTECHPAYRESP. “Type of Response structure

Populate API URL in variable I\_URL.

i\_url = 'https://api-sit.jio.com/partner/v1/funds/transfer'.

i\_authkey = 'l7xxc8af258c975d4fdab8267015a9603863'.

Create object for IF\_HTTP\_Client.

\* Creation of New IF\_HTTP\_Client Object  
  CLEAR : oref\_http\_client.  
  cl\_http\_client=>create\_by\_url(  
    EXPORTING  
      url                = i\_url  
    IMPORTING  
      client             = oref\_http\_client  
    EXCEPTIONS  
      argument\_not\_found = 1  
      plugin\_not\_active  = 2  
      internal\_error     = 3  
      OTHERS             = 4 ).

Set Headers:

\* Set Content Type  
  oref\_http\_client->request->set\_content\_type( 'application/json' ).  
  
\* Set Method  
  oref\_http\_client->request->set\_method( method = 'POST' ).  
  
\* Auth Key  
  CLEAR : gv\_header\_name.  
  gv\_header\_name = text-h01. "X-API-Key  
  CALL METHOD oref\_http\_client->request->set\_header\_field  
    EXPORTING  
      name  = gv\_header\_name "X-API-Key  
      value = i\_authkey.  
  
\* Bank ID  
  CLEAR : gv\_header\_name.  
  gv\_header\_name = text-h02. "bankid  
 i\_bankid = 'YBL'.

  CALL METHOD oref\_http\_client->request->set\_header\_field  
    EXPORTING  
      name  = gv\_header\_name "bankid  
      value = i\_bankid.

To convert the Work area values into JSON format we need to call Method: serialize of Class: /ui2/cl\_json. This class is not available in DER, so Custom class: zui\_cl\_json have been created for same purpose.

\* Prepare JSON  
  zui\_cl\_json=>serialize(  
    EXPORTING  
      data             =  i\_json\_out  
      compress         =  abap\_false  
      pretty\_name      =  zui\_cl\_json=>pretty\_mode-low\_case  
    RECEIVING  
      r\_json           =  gv\_json  ).

\* Remove Spaces  
  SHIFT gv\_json RIGHT DELETING TRAILING space.

JSON may or may not be case sensitive. In this case, it is camel case.

So, need to convert the fields in camel case.

\* Convert in camelCase  
  PERFORM : f\_replace USING 'uniquerequestno'           'uniqueRequestNo'            CHANGING gv\_json,  
            f\_replace USING 'appid'                     'appID'                      CHANGING gv\_json,  
            f\_replace USING 'customerid'                'customerID'                 CHANGING gv\_json,  
            f\_replace USING 'debitaccountno'            'debitAccountNo'             CHANGING gv\_json,  
            f\_replace USING 'beneficiarydetail'         'beneficiaryDetail'          CHANGING gv\_json,  
            f\_replace USING 'beneficiaryname'           'beneficiaryName'            CHANGING gv\_json,  
            f\_replace USING 'fullname'                  'fullName'                   CHANGING gv\_json,  
            f\_replace USING 'beneficiaryaddress'        'beneficiaryAddress'         CHANGING gv\_json,  
            f\_replace USING 'beneficiarycontact'        'beneficiaryContact'         CHANGING gv\_json,  
            f\_replace USING 'mobileno'                  'mobileNo'                   CHANGING gv\_json,  
            f\_replace USING 'emailid'                   'emailID'                    CHANGING gv\_json,  
            f\_replace USING 'beneficiaryaccountno'      'beneficiaryAccountNo'       CHANGING gv\_json,  
            f\_replace USING 'beneficiaryifsc'           'beneficiaryIFSC'            CHANGING gv\_json,  
            f\_replace USING 'beneficiarymobileNo'       'beneficiaryMobileNo'        CHANGING gv\_json,  
            f\_replace USING 'beneficiarymmid'           'beneficiaryMMID'            CHANGING gv\_json,  
            f\_replace USING 'transfertype'              'transferType'               CHANGING gv\_json,  
            f\_replace USING 'transfercurrencycode'      'transferCurrencyCode'       CHANGING gv\_json,  
            f\_replace USING 'transferamount'            'transferAmount'             CHANGING gv\_json,  
            f\_replace USING 'remittertobeneficiaryinfo' 'remitterToBeneficiaryInfo'  CHANGING gv\_json.

\*&---------------------------------------------------------------------\*  
\*&      Form  F\_REPLACE  
\*&---------------------------------------------------------------------\*  
FORM f\_replace  USING  p\_old  
                       p\_new  
                CHANGING pv\_json TYPE string.  
  
  REPLACE ALL OCCURRENCES OF p\_old IN pv\_json WITH p\_new.  
  
ENDFORM.                    " F\_REPLACE

\* Set Data to IF\_HTTP\_CLIENT  
  oref\_http\_client->request->set\_cdata( gv\_json ).

\* Send JSON to SCEO  
  oref\_http\_client->send(  
    EXCEPTIONS  
      http\_communication\_failure = 1  
      http\_invalid\_state         = 2 ).  
  IF sy-subrc IS NOT INITIAL.  
\*    Error.  
  ENDIF.

\* Disable Logon PopUp  
  oref\_http\_client->propertytype\_logon\_popup = if\_http\_client=>co\_disabled.

\* Get Response from SECO  
  CLEAR : gv\_result.  
  oref\_http\_client->receive(  
    EXCEPTIONS  
      http\_communication\_failure = 1  
      http\_invalid\_state         = 2  
      http\_processing\_failed     = 3 ).  
  IF sy-subrc IS INITIAL.  
  
    gv\_result =  oref\_http\_client->response->get\_cdata( ).

    TRY.  
        zui\_cl\_json=>deserialize(  
          EXPORTING  
            json             = gv\_result  
          CHANGING  
            data             = e\_json\_in ).  
  
      CATCH cx\_root.  
    ENDTRY.

ENDIF.

e\_json\_in is TYPE of Response format. It will capture the data provided by bank.