

User manual for API registration Data Submission (Testing Phase)

Part A: Registration for API

1. Details of UAT member registration are as under:

URL: <https://uat.connect2nsccl.com/MemberPortal/>

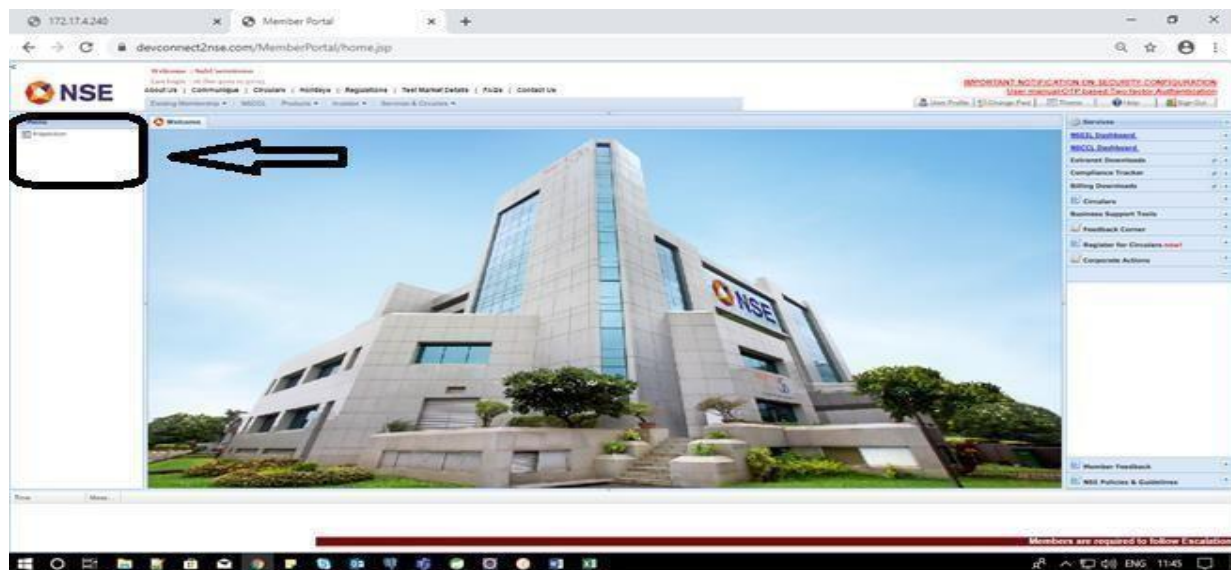
or <https://www.devconnect2nse.com/MemberPortal/>

Member Code: Your 5 digit Member Code

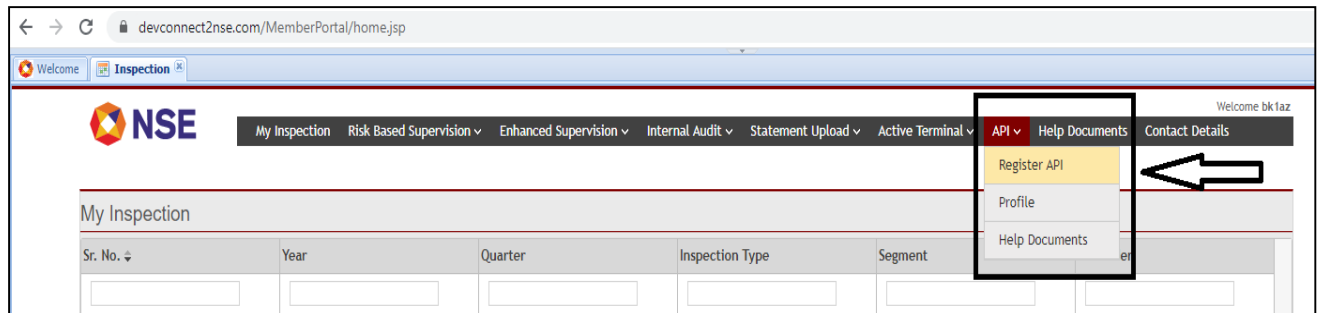
Password: Abcd@1234

If first time login, members will have to change the password and re-login with new password.

2. After successfully re-login, click on “Inspection” tab and proceed for API registration.



3. Navigation: Inspection---> API---> Register API. Click “Register API”



4. After clicking the “Register API” button and member will have to submit the following information:

Sr. No.	Field	Validation points
1	Login Id	Login Id should be alphanumeric and 6 to 15 characters long. Special characters are not allowed.
2	Member PAN	Alpha-numeric Trading member PAN Character (10)
3	Email Id	Trading member’s Email Id
4	Phone No.	Mobile number Character (10) and should start with 9,8 or 7 only.
5	Password	Characters (12) (At least one capital character, one small character, one numeric and one special character from @,#,\$,%,^,&,* ,=)

Note:

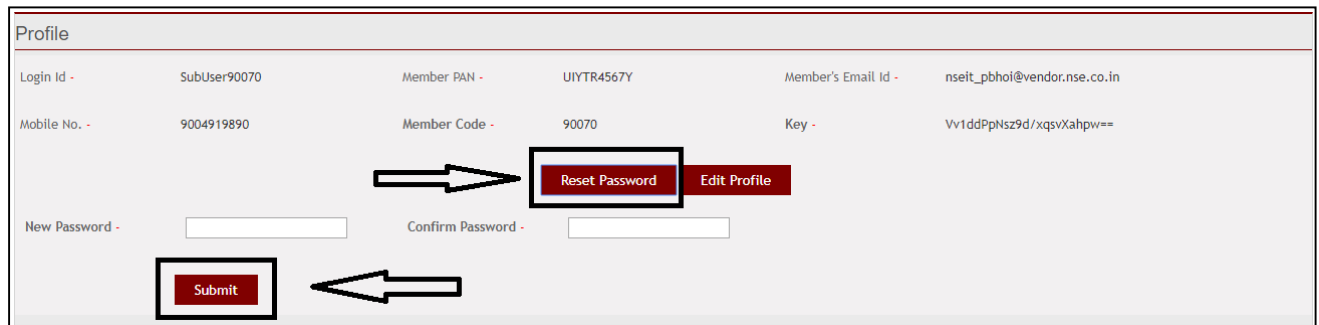
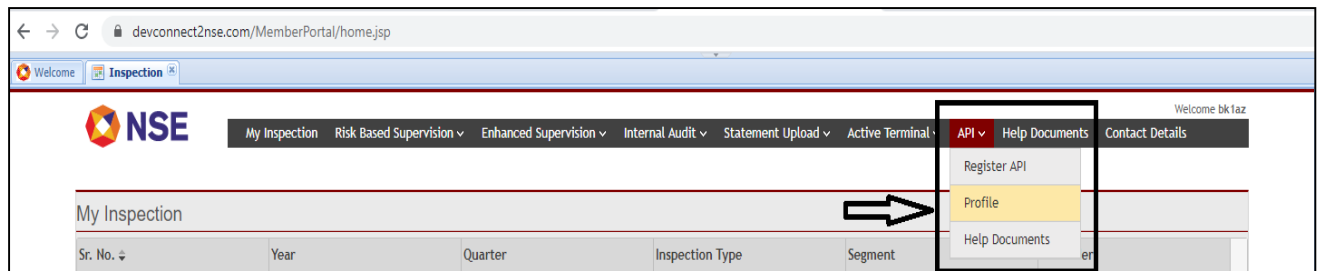
Login Id: ApiUser<5 digit member code>

(For example, if member code is 12345 then User id will be ‘ApiUser12345’)

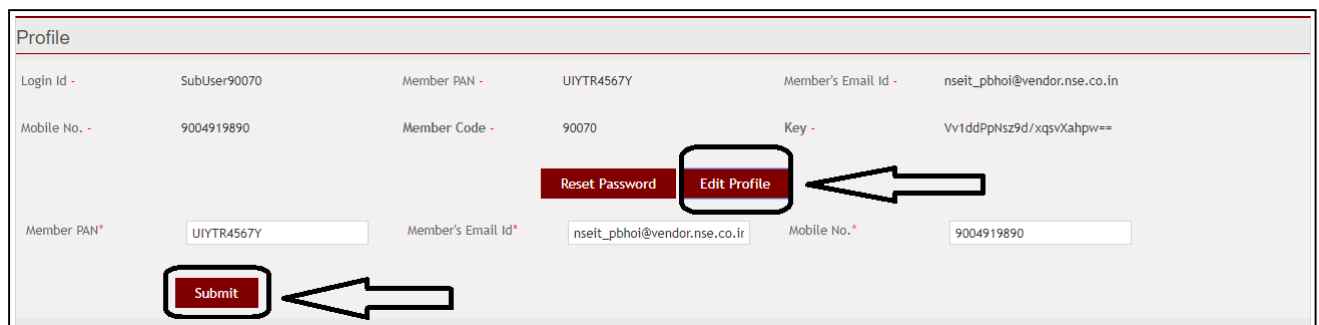
After providing inputs for the information sought, click on “Register” button.

The screenshot shows the 'Register API' form in the NSE Member Portal. The form includes the following fields: Login Id*, Member PAN*, Email Id*, Phone No.*, Password*, and Confirm Password*. At the bottom of the form, there are two buttons: 'Register' and 'Reset'. The 'Register' button is highlighted with a red box and an arrow.

5. After registration, member can view the profile by clicking “Profile” button. Further, member can reset the API password, Email id and Phone no. by clicking “Reset” button.



6. Member can change “Member PAN”, “Member’s Email ID”, and “Mobile No.” on click of “Edit Profile” button.



7. After successful registration, member will receive an e-mail, on the e-mail address provided by member while registration, which contains Login Id, Password, and secret key.

Note: Secret key will change on every password reset.

8. 'URL' for API are mentioned below:

Note:

- The below URL's/endpoint cannot be called (used) directly from the browser (like google chrome, internet explorer and etc.).
- Members will have to develop 'Consumer/Client API' to call (use) the above URL's.
- Refer part B and consult your IT Team/ back-office Software vendor for developing the 'Consumer/Client API'.

Login:

<https://www.devconnect2nse.com/inspectionapi/login>

Holding Upload:

<https://www.devconnect2nse.com/inspectionapi/tradingHoldingUpload/{version}>

Cash and Cash Equivalent Upload:

<https://www.devconnect2nse.com/inspectionapi/tradingCashAndCashEquivalentUpload/{version}>

EOD Bank Balance Upload:

<https://www.devconnect2nse.com/inspectionapi/tradingEodBalanceUpload/{version}>

Logout:

<https://www.devconnect2nse.com/inspectionapi/logout>

Note: Now version is 1.0

Part B: Other points for Login through API (Below exercise to be carried out by IT team/back-office software vendor of member)

9. Process to generate encrypted password with key

To generate Encrypted Password with key you have to use the transformation "AES/ECB/PKCS5Padding" where **AES** is cipher with **ECB** is mode of operation and **PKCS5** is padding scheme with **256 bit key**.

Note: -

1. IT team/back-office software vendor needs to develop code snippet for generating encrypted password with key. Password encrypted using online tool/online sites may differ from site to site and can be incorrect.

Note:-

- ☐ POSTMEN tool is used for testing purpose only. Members will have to develop 'Consumer/Client API' to submit different statements data in live environment.
- ☐ Use 7Zip tool to generate .gzip files for testing through postman.
- ☐ For testing API's though Postmen, please refer point no. 10 to 13.
- ☐ Code snippet of AES and GZip is added at the end of this document.

10. How to login with /login API endpoint?

- Create **login text file** with below Jason code having details Member Code, Login ID, and Password.

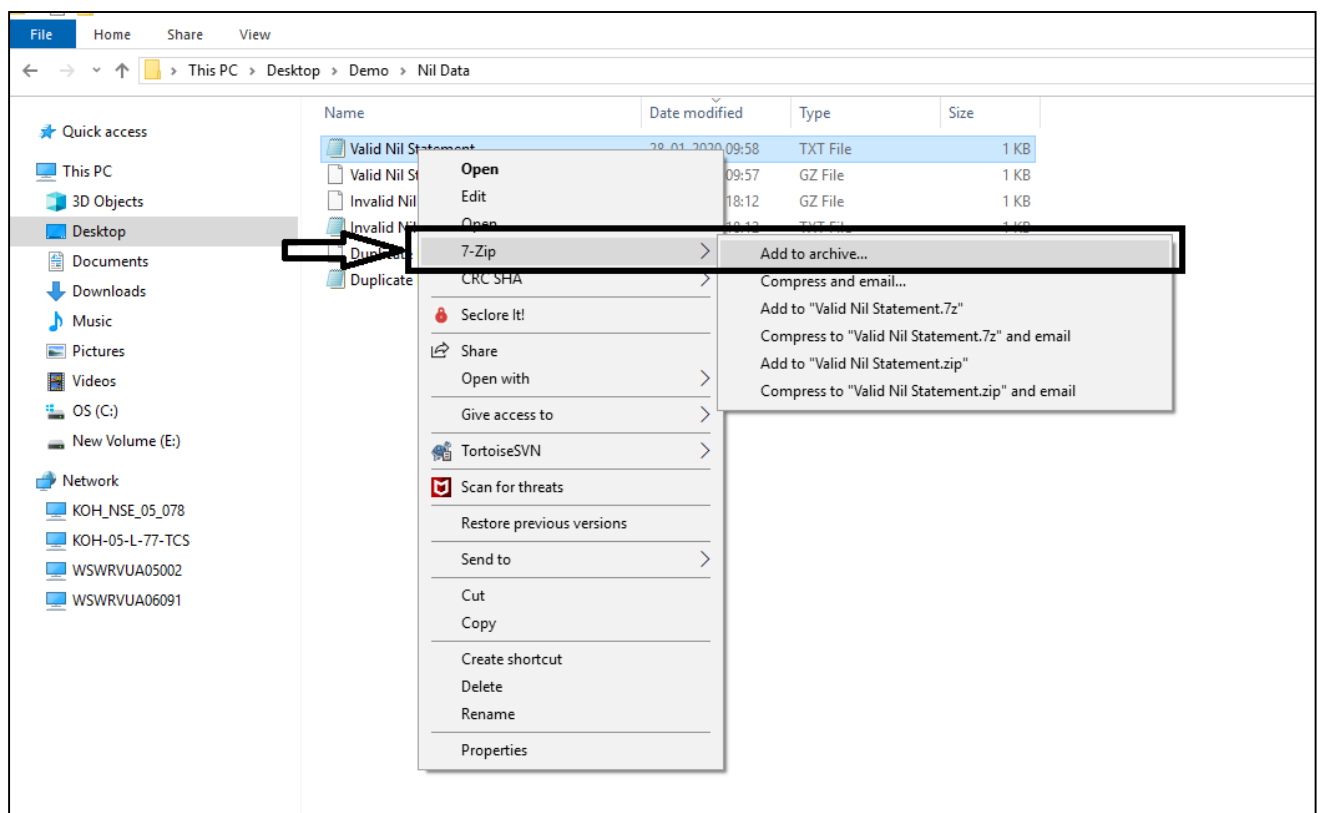
Login Request Object Structure (JSON)

```
{
  "memberCode": "06000",
  "loginId": "ApiUser06000",
  "Password": "qi8H8R7OM4xMUNMPuRAZx1Y"
}
```

Note: Use login Id and encrypted password.

- **How to convert above text file into gzip File format?**

Following is the diagrammatical representation of conversion of text file into .gzip file format using 7-Zip tool,



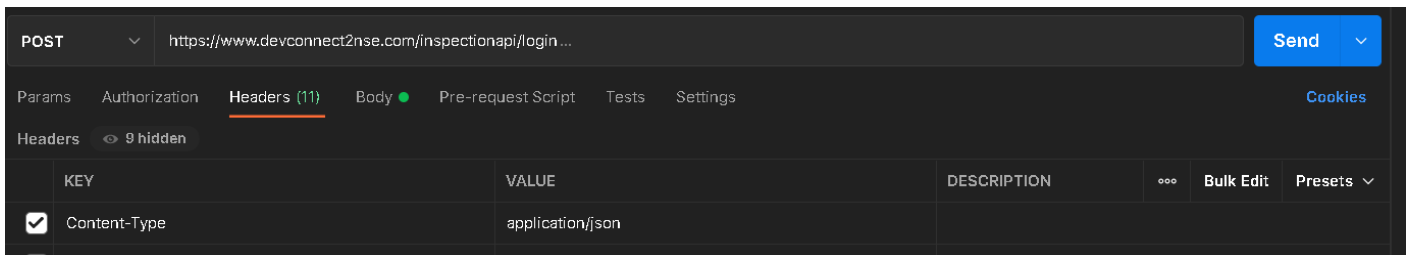
- ❑ After opening the Postmen tool/software, please select “POST” method & enter login URL in address bar.

LOGIN:

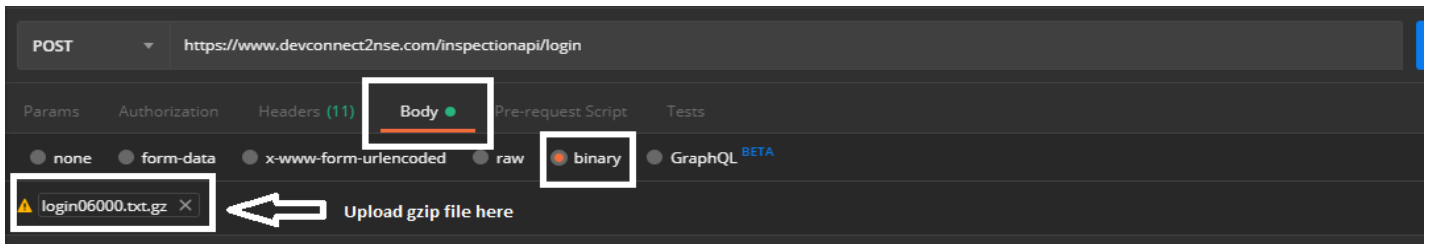
<https://www.devconnect2nse.com/inspectionapi/login>

- Click on “Headers” tab [Check below image]

In “Key” column enter Content-Type and in “VALUE” column enter application/json.



- Click on “Body” tab ==>> Select “binary” radio button [Please check below image]
- Click on “Select File” button and upload your gzip File. [Please check below image]



- Click on “Send” button. Following response will be display in json format.

Login Response Object Structure (JSON)

```
{
  "memberCode": "06000",
  "code": ["601"],
  "loginId": "ApiUser06000",
  "token": "eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIwOTEwMCIsImIzcyI6IkkFwaVVzZXIwOTEwMCIsImV4cCI6MTYzMjIwNzYyOCwiaWF0IjoxNjMyMjA0MDI4LCJqdGkiOiJhNWl0YjZiZi1lZTQwLTQ4NzItOGM4ZC0xNjE5ZjA2Nzk2NTEifQ.gGmBfpIbtP59X6gK28z41611OkFgOBOIGjAX-WhjkY5DPJRxHJWtc-TOcDyPjLGAvnHNivh-SZRRawxFSNgYw",
  "status": "success"
}
```

11. How to upload holding statement data with /tradingHoldingUpload/{version} API endpoint?

- Create holding statement text file in JSON format.

Holding Statement Request Object Structure (JSON)

```
{
  "memberPan": "JULYY2019T",
  "lastWeekDay": "23-01-2021",
  "weekDay": "22-01-2021",
  "holdingData": [
    {
      "recId": "2",
      "dmat": "IN30012600078641",
      "accountType": "POOL",
      "ucc": "271876",
      "clientName": "PRAVIN CHANDRA RATILAL MEHTA",
      "pan": "ACVPM0782E",
      "isin": "IN0020200286",
      "securityType": "BOND",
      "totalPldgQty": "0",
      "freeBalQty": "1",
      "totalQty": "1",
      "action": "A",
      "nameOfSecOrCommodity": "NA",
      "unitType": "NA"
    },
    {
      "recId": "3",
      "dmat": "IN30012600078641",
      "accountType": "POOL",
      "ucc": "281896",
      "clientName": "JYOTI RAMASWAMY",
      "pan": "AFRPR5608B",
      "isin": "IN0020180488",
      "securityType": "EQ",
      "totalPldgQty": "0",
      "freeBalQty": "94",
      "totalQty": "94",
      "action": "A",
      "nameOfSecOrCommodity": "NA",
      "unitType": "NA"
    },
    {
      "recId": "4",
      "dmat": "IN30012600078641",
      "accountType": "POOL",
      "ucc": "333155",
      "clientName": "KAJARI CHAKRABARTI",
      "pan": "AESPC1984M",
      "isin": "IN0020040039",
      "securityType": "BOND",
      "totalPldgQty": "0",
      "freeBalQty": "15",
      "totalQty": "15",
      "action": "A",
      "nameOfSecOrCommodity": "NA",
      "unitType": "NA"
    }
  ]
}
```


12. Convert above text file into gzip File format. [Check procedure shown for /login endpoint.]

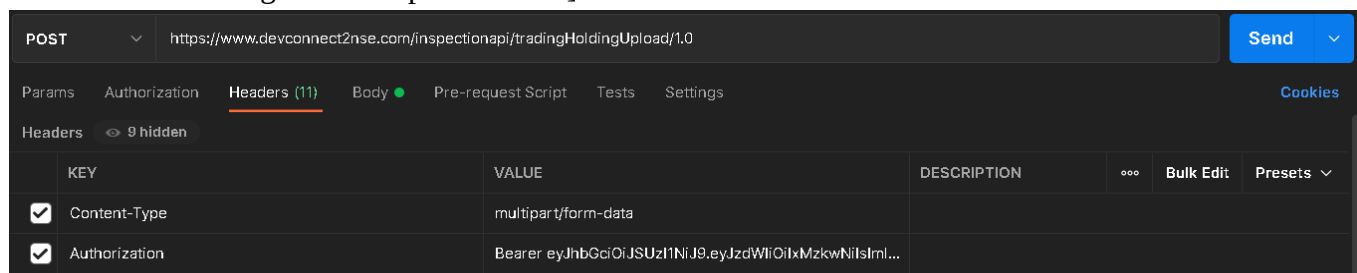
Open Postmen select “POST” method & enter holding upload URL in address bar.

<https://www.devconnect2nse.com/inspectionapi/tradingHoldingUpload/{version}>

- Click on “Headers” tab [Check bellow image]

In “Key” column enter Content-Type and in “VALUE” column enter multipart/form-data.

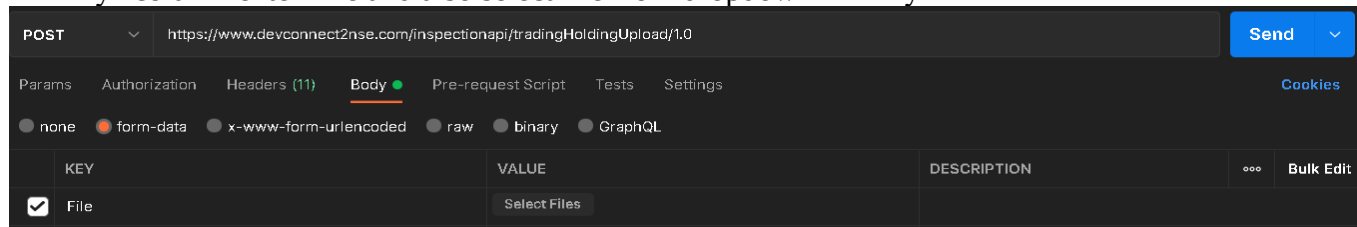
In “Key” column enter Authorization and in “VALUE” column enter token which is generate in Login api.[Note: - You need to add a “Bearer” before token. It should have space between Bearer & Token. E.g. Bearer<space>Token]



The screenshot shows the Postman interface with the 'Headers' tab selected. The URL bar contains 'https://www.devconnect2nse.com/inspectionapi/tradingHoldingUpload/1.0'. The 'Headers' tab is active, showing a table with two headers: 'Content-Type' with value 'multipart/form-data' and 'Authorization' with value 'Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMzkwNiIsIm...'. There are buttons for 'Bulk Edit' and 'Presets'.

- Click on “Body”_tab ==>> Select “form-data” radio button [Please check below image]

In “Key” column enter File and also select file from dropdown in “Key”



The screenshot shows the Postman interface with the 'Body' tab selected. The URL bar contains 'https://www.devconnect2nse.com/inspectionapi/tradingHoldingUpload/1.0'. The 'Body' tab is active, showing radio buttons for 'none', 'form-data' (selected), 'x-www-form-urlencoded', 'raw', 'binary', and 'GraphQL'. Below the radio buttons, there is a table with one header: 'File' with a 'Select Files' button.

- In “VALUE” column Click on “Select Files” button and upload your .gzip File.
- Click on “Send” button. Following response will be display in JSON format.

Holding Statement Response Object Structure (JSON)

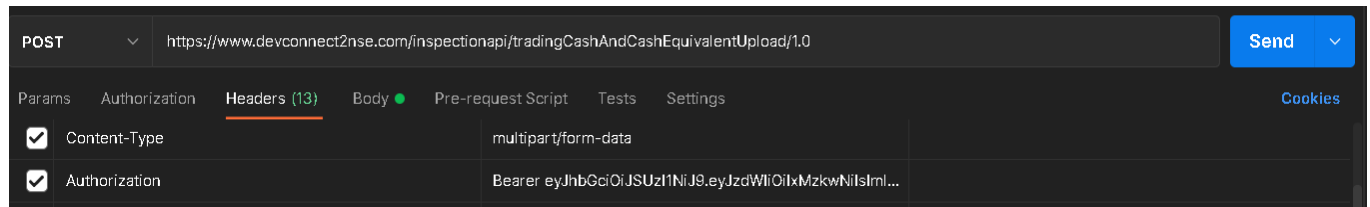
```
{
  "hdrCode": ["601"],
  "ackId": "091002201202126",
  "memberPan": "JALYY2019T",
  "hdrStatus": "success",
  "weekDay": "22-01-2021",
  "holdingData": [
    { "recId": "2", "code": ["601"], "status": "success" },
    { "recId": "3", "code": ["601"], "status": "success" },
    { "recId": "4", "code": ["601"], "status": "success" }
  ],
  "lastWeekDay": "23-01-2021"
}
```

Note:-

- "lastWeekDay" should be Saturday date only.
- "weekDay" should be dates from holding period only. i.e if week start from 09-12-2019 TO 14-12-2019 then "lastWeekDay" should be "14-12-2019" and "weekDay" should be from 09-12-2019 TO 13-12-2019, other dates result into the error.

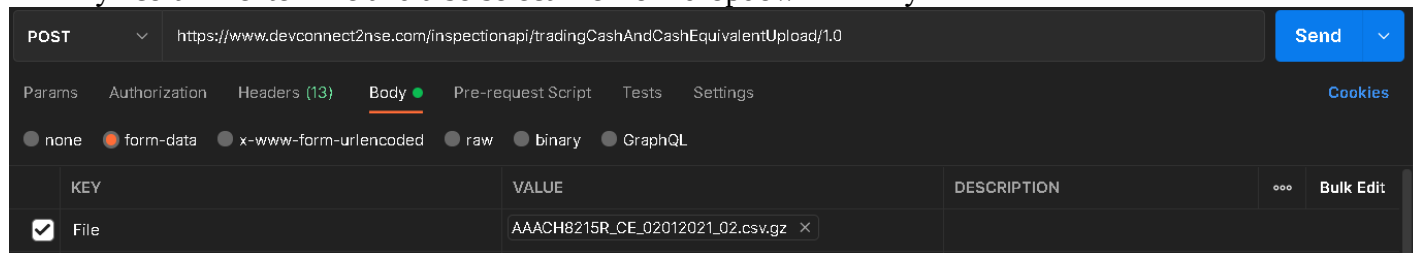
13. How to upload holding statement data with /tradingCashAndCashEquivalentUpload/{version} API endpoint?

- ☐ For Cash and Cash Equivalent, csv file need to be uploaded .Convert above csv file into gzip File format. [Check procedure shown for /login endpoint.]
- ☐ Open Postmen select “POST” method & enter holding upload URL in address bar.
<https://www.devconnect2nse.com/inspectionapi/tradingCashAndCashEquivalentUpload/{version}>
- Click on “Headers” tab [Check bellow image]
In “Key” column enter Content-Type and in “VALUE” column enter multipart/form-data.
In “Key” column enter Authorization and in “VALUE” column enter token which is generate in Login api.[Note: - You need to add a “Bearer” before token. It should have space between Bearer & Token. E.g. Bearer<space>Token]



The screenshot shows the Postman interface with the 'Headers' tab selected. The URL is `https://www.devconnect2nse.com/inspectionapi/tradingCashAndCashEquivalentUpload/1.0`. Two headers are defined: 'Content-Type' with value 'multipart/form-data' and 'Authorization' with value 'Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMzkwNiIsIm1...

- Click on “Body”_tab ==>> Select “form-data” radio button [Please check below image]
In “Key” column enter File and also select file from dropdown in “Key”



The screenshot shows the Postman interface with the 'Body' tab selected. The 'form-data' radio button is chosen. A single key-value pair is added: 'File' as the key and a file named 'AAACH8215R_CE_02012021_02.csv.gz' as the value.

- In “VALUE” column Click on “Select Files” button and upload your .gzip File.
Note: CSV file Nomenclature should be in this format PAN_CE_DDMMYYYY_Seqno.gz
- Click on “Send” button. Following response will be display in JSON format.

CashAndCashEquivalent Response Object Structure (JSON)

```
"TRADING MEMBER PAN",DATE,"UNIQUE CLIENT CODE","CLIENT PAN","CLIENT NAME","MTF/ NON  
MTF INDICATOR","FINANCIAL LEDGER BALANCE-A","FINANCIAL LEDGER BALANCE (CLEAR)-  
B","PEAK FINANCIAL LEDGER BALANCE (CLEAR)-C","FINANCIAL LEDGER BALANCE-  
MCX","FINANCIAL LEDGER BALANCE-NCDEX","FINANCIAL LEDGER BALANCE-ICEX","BANK GUARANTEE  
(BG)","FIXED DEPOSIT RECEIPT (FDR)","GOVERNMENT OF INDIA SECURITIES (GSEC)","GILT  
FUNDS","CREDIT ENTRY IN LEDGER IN LIEU OF EPI","POOL ACCOUNT","UNCLEARED  
CHEQUES","VALUE OF COMMODITIES","CASH COLLATERAL FOR MTF  
POSITIONS","UNCLAIMED/UNSETTLED CLIENT FUNDS","CLIENT BANK ACCOUNT NO","LAST  
SETTLEMENT DATE","ES INFORMATION TYPE",VALUE,REMARKS  
  
ABCDEF0000C,01-01-2021,1000007,AEDCB0000F,"TARUN KUMAR JHA","NON  
MTF",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2345.123,NO,22,NA,NA,NA,OK  
ABCDEF0000C,01-01-2021,1000011,APFPF0000D,"MUKESH RAMLAL  
PATEL","MTF",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4534.432,NO,22,NA,NA,NA,OK
```

14. How to upload holding statement data with /tradingEodBalanceUpload/{version} API endpoint?

- Create holding statement text file in JSON format.

EOD Bank Balance Request Object Structure (JSON)

```
{
  "BankBalanceData": [{
    "Bank Account No": 00990610014806,
    "IFSC": "HDFC0000060",
    "Bank Account Type": ANY OTHER,
    "03-05-2021": 102007496,
    "04-05-2021": 477298321,
    "05-05-2021": 134428920,
    "06-05-2021": 167572707,
    "07-05-2021": 307590962,
    "08-05-2021": 298195332
  },
  {
    "Bank Account No": 00990630003430,
    "IFSC": "HDFC0000060",
    "Bank Account Type": "SETTLEMENT ACCOUNT",
    "03-05-2021": 196163955,
    "04-05-2021": 275308037,
    "05-05-2021": 262304676,
    "06-05-2021": 416129907,
    "07-05-2021": 212682960,
    "08-05-2021": 462731157
  }
]
```

- ☐ Convert above text file into gzip File format. [Check procedure shown for /login endpoint.]

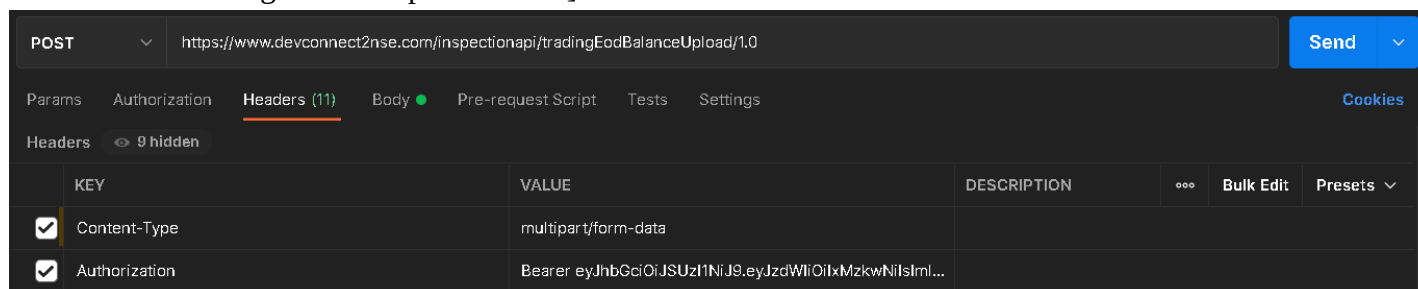
- ☐ Open Postmen select “POST” method & enter holding upload URL in address bar.

<https://www.devconnect2nse.com/inspectionapi/tradingEodBalanceUpload/{version}>

- Click on “Headers” tab [Check bellow image]

In “Key” column enter Content-Type and in “VALUE” column enter multipart/form-data.

In “Key” column enter Authorization and in “VALUE” column enter token which is generate in Login api.[Note: - You need to add a “Bearer” before token. It should have space between Bearer & Token. E.g. Bearer<space>Token]



- Click on “Body”_tab ==>> Select “form-data” radio button [Please check below image]

In “Key” column enter File and also select file from dropdown in “Key”

In “VALUE” column Click on “Select Files” button and upload your .gzip File.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `https://www.devconnect2nse.com/inspectionapi/tradingEodBalanceUpload/1.0`
- Buttons:** Params, Authorization, Headers (11), **Body** (selected), Pre-request Script, Tests, Settings, Cookies, and a blue **Send** button.
- Body Type:** A row of radio buttons includes **none**, **form-data** (selected), **x-www-form-urlencoded**, **raw**, **binary**, and **GraphQL**.
- Table:** A table with columns KEY, VALUE, and DESCRIPTION. The first row has a checked checkbox in the KEY column, the value "File" in the VALUE column, and a file selection button "AABCZ2616B_BA_31072021.txt.gz" with a close icon. The DESCRIPTION column is empty.

- Click on “Send” button. Following response will be display in JSON format.
Eod Bank Balance Response Object Structure (JSON)

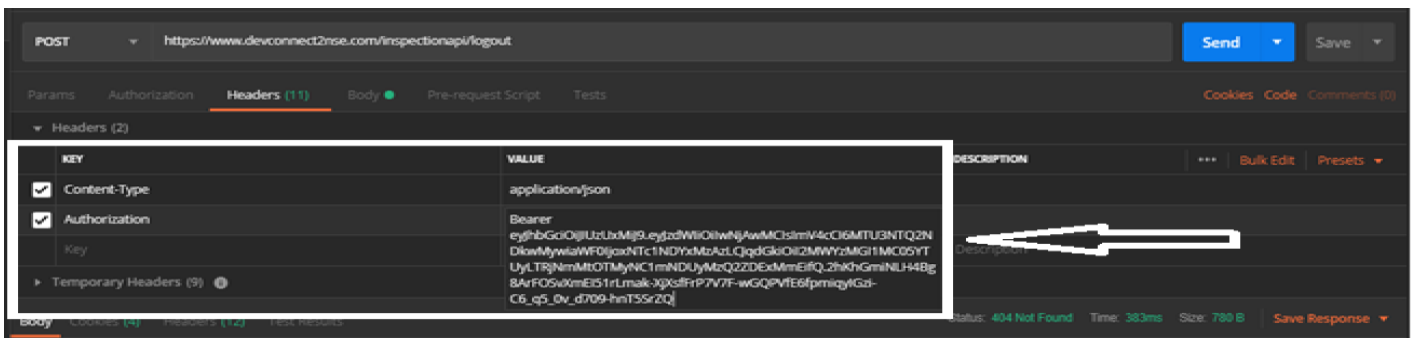
```
[ {
  "Bank Account No" : "00990610014806",
  "IFSC" : "HDFC0000060",
  "Bank Account Type" : "ANY OTHER",
  "03-05-2021" : "102007496",
  "04-05-2021" : "477298321",
  "05-05-2021" : "134428920",
  "06-05-2021" : "167572707",
  "07-05-2021" : "307590962",
  "08-05-2021" : "298195332",
  "Status" : "Success"
}, {
  "Bank Account No" : "00990630003430",
  "IFSC" : "HDFC0000060",
  "Bank Account Type" : "SETTLEMENT ACCOUNT",
  "03-05-2021" : "196163955",
  "04-05-2021" : "275308037",
  "05-05-2021" : "262304676",
  "06-05-2021" : "416129907",
  "07-05-2021" : "212682960",
  "08-05-2021" : "462731157",
  "Status" : "Success"
} ]
```

15. How to Logout with /logout API endpoint?

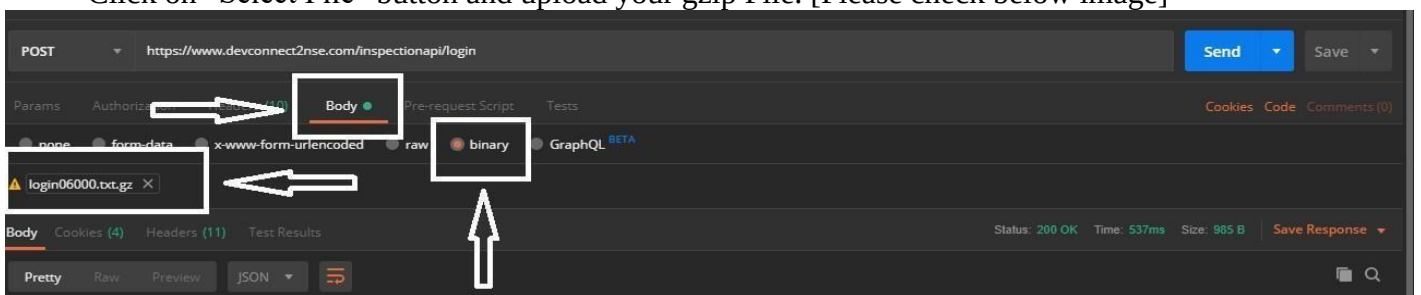
- ☐ **Create logout text file with below JSON code such as, Logout Request Object Structure (JSON)**

```
{
    "memberCode": "06000",
    "loginId": "ApiUser06000"
}
```

- ❑ **Convert above text file into gzip File format [Check procedure shown for /loginendpoint.]**
- ❑ **Open Postmen select “POST” method & enter holding upload URL in address bar.**
<https://www.devconnect2nse.com/inspectionapi/logout>
- Click on “Headers” tab [Check bellow image]
In “Key” column enter Content-Type and in “VALUE” column enter application/json.
In “Key” column enter Authorization and in “VALUE” column enter token which is generate in Login api.[Note: - You need to add a “Bearer” before token. It should have space between Bearer & Token. E.g. Bearer<space>Token]



- Click on “Body” tab ==> Select “binary” radio button [Please check below image]
- Click on “Select File” button and upload your gzip File. [Please check below image]



- Click on “Send” button. Following response will be display in JSON format.
- Logout Response Object Structure (JSON)**

```
{
  "code": ["601"],
  "status": "success"
}
```

NOTE:

- 1) If account is inactive due to 5 incorrect password attempts, reset the password to reactivate the account. A new key will be generated on reset password.
- 2) Password encrypted using online site may be incorrect, develop own code snippet for encrypting password using steps specified in point no 9. Kindly don't use online sites for encrypting password.
- 3) JSON text file for testing generated online may be incorrect. Kindly don't use online sites for generating text files.

16. AES Code Snippet [Java]

```
public class CodeSnippet
{
    public static void main(String[] args) throws UnsupportedOperationException, NoSuchAlgorithmException,
    NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException
    {
        //Plain Text Password
        String password = "Nseitjan@123";

        //sample key is - "XBaNb0xmK2TNRIfcHA3F306Oi14HWAeYmtUd0qRheTc="
        String key = "XBaNb0xmK2TNRIfcHA3F306Oi14HWAeYmtUd0qRheTc=";
        //Key is converted to byte array
        byte[] keyByteArray = new Base64().decode(key.getBytes("UTF-8"));

        //SecretKeySpec is used to construct a SecretKey from a byte array
        SecretKeySpec secretKeySpec = new SecretKeySpec(keyByteArray, "AES");
        Cipher cipher = Cipher.getInstance("aes/ecb/pkcs5padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec);

        //pass plain text that is to be encrypt
        String encrypt = (new Base64()).encodeAsString(cipher.doFinal(password.getBytes()));

        //actual key in base64 format
        System.out.println("encrypted string:" + encrypt);
    }
}
```

17. Gzip Code Snippet [Java]

```
public class NetClientGet
{
    public static void main(String[] args) throws IOException
    {
        byte[] body = readTextFile();
        System.out.println("gzip bytes "+body);
    }
    public static byte[] readTextFile() throws IOException
    {
        //LoginJson.text contains json login Request example listed bellow
        //{
        // "memberCode": "06000",
        // "loginId": "SubUser06000",
        // "password":"sMzvV6fvENaQfRedWYd07w=="
        //}

        InputStream inputStream = new FileInputStream("<path>\\login06000.txt");
        @SuppressWarnings("resource")
        BufferedReader bufferReader = new BufferedReader(new InputStreamReader(inputStream));
        String line = bufferReader.readLine();
        StringBuilder stringBuffer = new StringBuilder(); while (line != null)
        {
            stringBuffer.append(line).append("\n"); line = bufferReader.readLine();
        }
        //Creates a gzip file compressString(stringBuffer.toString());
        //create gzip compress bytes.
        return compress(stringBuffer.toString());
    }

    public static byte[] compress(final String stringToCompress)
    {
        try (final ByteArrayOutputStream baos = new ByteArrayOutputStream();
            final GZIPOutputStream gzipOutput = new GZIPOutputStream(baos))
        {
            gzipOutput.write(stringToCompress.getBytes(StandardCharsets.UTF_8)); gzipOutput.finish();
            return baos.toByteArray();
        } catch (IOException e)
        { throw new UncheckedIOException("Error while compression!", e);}
    }

    public static String compressString(String str) throws IOException
    {
        if (str == null || str.length() == 0)
        {
            return str;
        }
        BufferedWriter writer = null;
        try{
            File file = new File("your.gzip");
            GZIPOutputStream zip = new GZIPOutputStream(new FileOutputStream(file)); writer = new
            BufferedWriter(new OutputStreamWriter(zip, "UTF-8")); writer.append(str);
        }
        Finally{
            if(writer != null){ writer.close();
            }
        }
        return str;
    }
}
```


AES Code Snippet [.Net]

```
using System;
using System.Security.Cryptography; using System.Text;

namespace AES256{
class Program{
private static string getString(byte[] b)
{
return Encoding.UTF8.GetString(b);
}
static void Main(string[] args){
byte[] data = Encoding.UTF8.GetBytes("NseitJan@201");
byte[]a = Convert.FromBase64String("AAECAwQFBgcICQoLDA0ODw==");
Console.WriteLine("Key : {0}", getString(a));
byte[] enc = Encrypt(data, a);
string result = Convert.ToBase64String(enc);
Console.WriteLine("Encrypted text", result);
byte[] dec = Decrypt(enc, a);
Console.WriteLine("Encrypted : {0}", getString(enc));
Console.WriteLine("Decrypted : {0}", getString(dec));
// Console.ReadKey();
}
public static byte[] Encrypt(byte[] data, byte[] key){
using (RijndaelManaged csp = new RijndaelManaged())
{
csp.KeySize = 256;
csp.BlockSize = 128; csp.Key = key;
csp.Padding = PaddingMode.PKCS7;
csp.Mode = CipherMode.ECB;
CryptoTransform encrypter = csp.CreateEncryptor();
return encrypter.TransformFinalBlock(data, 0, data.Length);
}
}
private static byte[] Decrypt(byte[] data, byte[] key){
using (RijndaelManaged csp = new RijndaelManaged())
{
csp.KeySize = 256;
csp.BlockSize=128;
csp.Key = key;
csp.Padding = PaddingMode.PKCS7;
csp.Mode = CipherMode.ECB;
CryptoTransform decrypter = csp.CreateDecryptor();
return decrypter.TransformFinalBlock(data, 0, data.Length);
}
}
}
}
```

Technology Used:

Language: - java version "11.0.12" 2021-07-20 LTS

End of document