**SageX3 – POS Integration Web service**

**Steps to Run Web service:**

1. Place the ‘**SageWS’** and ‘**SageWS App**’ folder inside” C: // “- drive.
2. Place the output text file (POS\_SO\_Input.txt) from POS, inside “C: //SageWS//POS OUTPUT// “- folder.
3. Run the ‘**SageWS.exe’** file from Desktop.

**Folder Structure:**

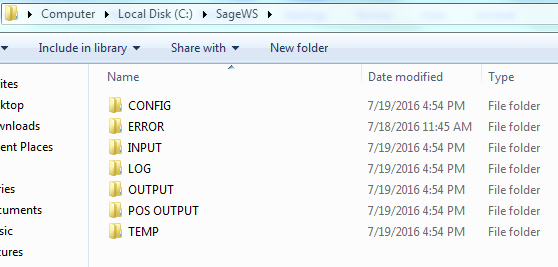
****

FIG: sample **POS Output file** is stored at C:\SageWS\POS OUTPUT\ POS\_SO\_Output.txt

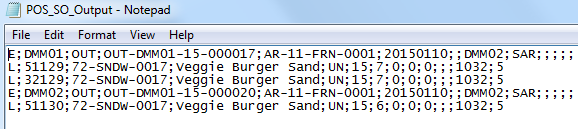
****

FIG: sample X3 Web service **Output file** is stored at C:\SageWS\OUTPUT\ X3\_SO\_Output.txt

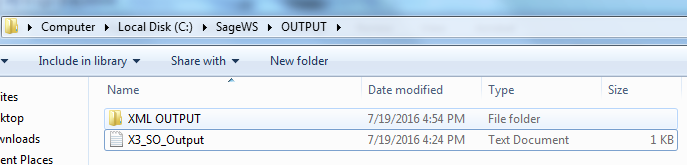
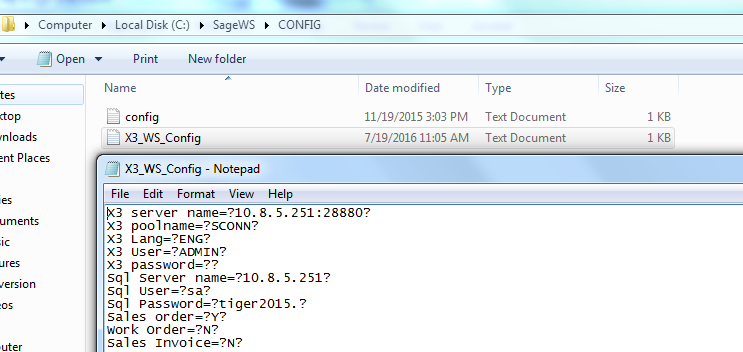
****

FIG: X3 Web service **config file** is stored at C:\SageWS\CONFIG\ X3\_WS\_Config.txt

****

X3 Web service **ERROR file** is stored at C:\SageWS\ERROR\X3\_POS\_SO\_ERROR.txt

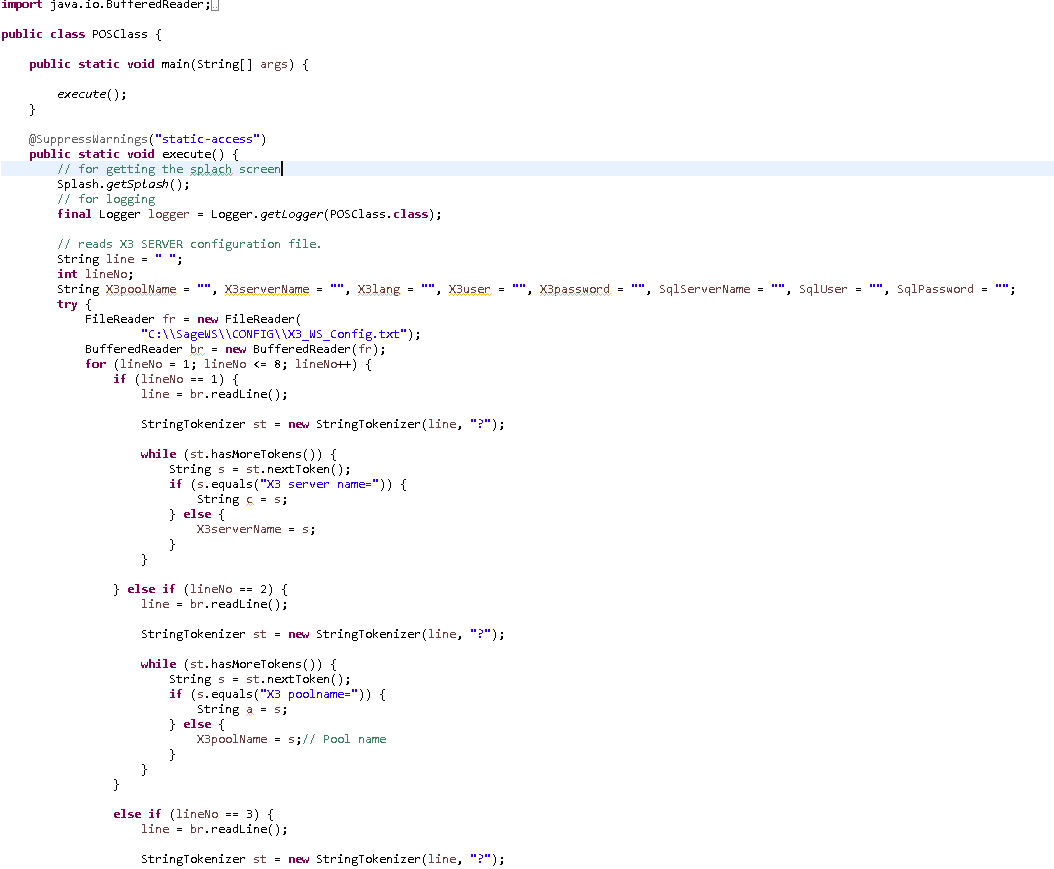
X3 Web service **LOG file** is stored at C:\SageWS\LOG\ X3\_SO\_LOG.txt.

X3 Web service **Temporary files** is stored at ‘C: \SageWS\TEMP\’ Folder.

**Code Wise Review**

**Work Flow for Sales Order Creation**:

POSClass.java - Main Class



* POSClass.java class has Two Methods:
  + Main method and Execute method.
  + Main method is for just invoking the execute method.
  + Execute method has the implementation for the web service.
    - Execute Method
      * Reads following ‘X3 SERVER’ configuration details from “C:\\SageWS\\CONFIG\\X3\_WS\_Config.txt”.
        + X3 server name
        + X3 poolname
        + X3 Lang
        + X3 User
        + X3 password
        + Sql Server name
        + Sql User
        + Sql Password
      * Reading the Input file ‘X3\_POS\_SO\_Input.txt’
        + Entering in a loop based on the number of lines.
        + Also checking the entry is a Header or Line by checking it by E or L.

Fig: Iterating the input txt and checking for Header (E).

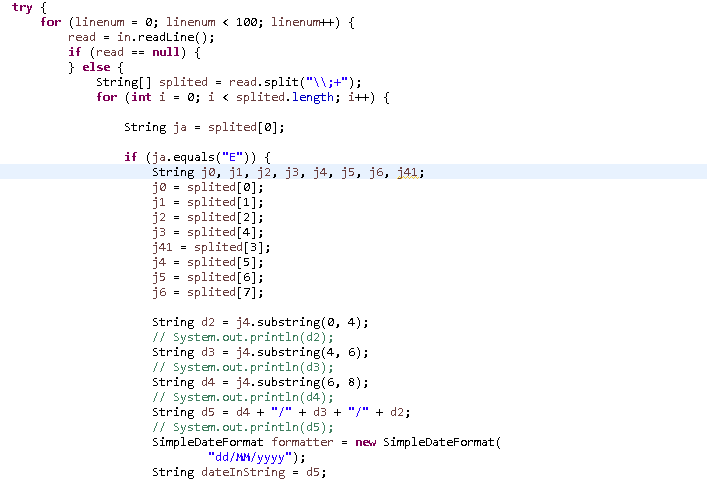


Fig: Table\_L – For Lines in Sage X3 SQL Server.

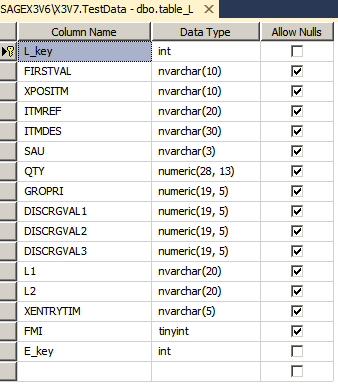
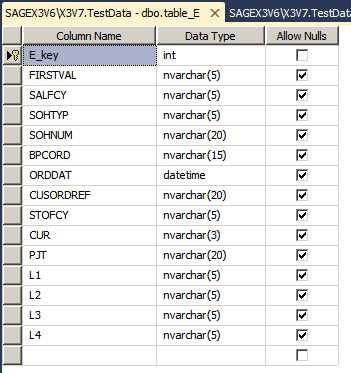
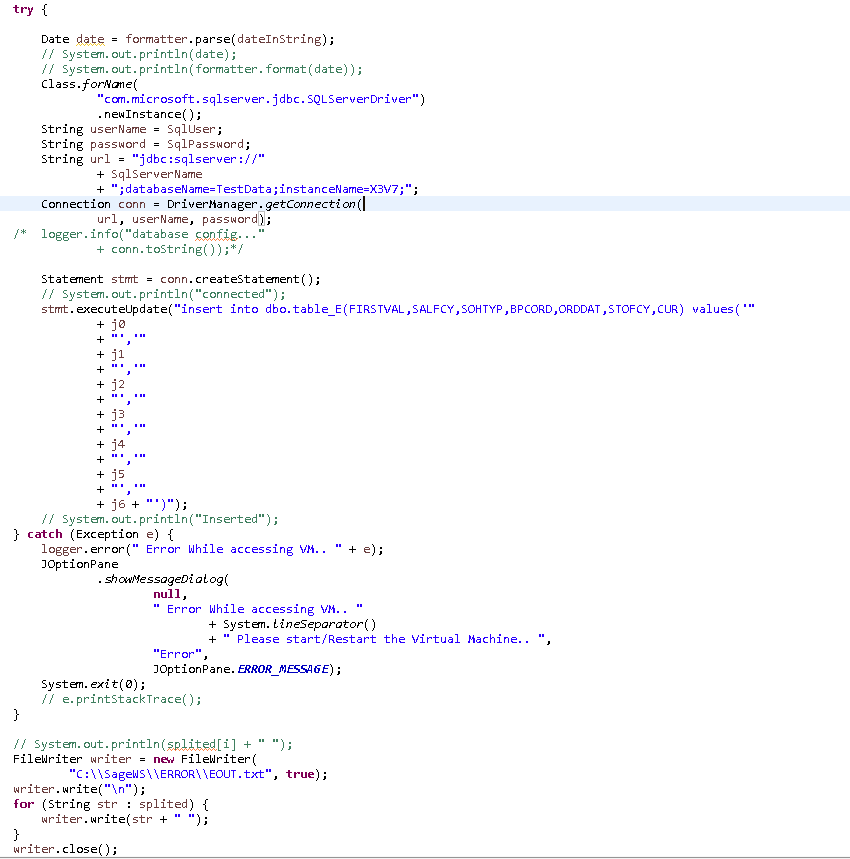


Fig: Table\_E – For Header in Sage X3 SQL Server.



* For Header(E),
  + Creating the connection with X3 Server SQL Database.
  + For this a temporary database is created, called ‘TestData’.
  + Deletes the previous values from table ‘table\_E’.
  + Insert the Values fetched from Input file into Table ‘table\_E’ if the line starts with ‘E’.
  + Also keeps the inserted entry into the ‘C:\SageWS\TEMP\EOUT.txt’.

Fig: Inserting the header details into Database.

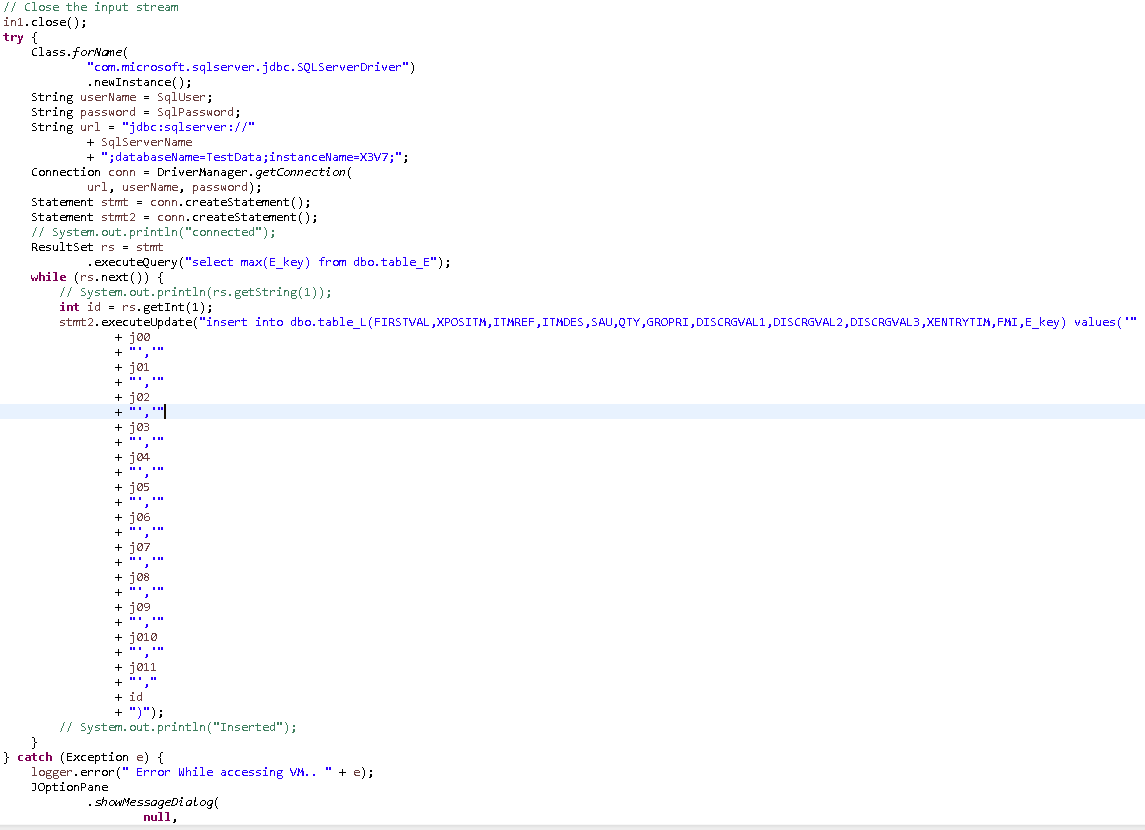


* For Lines (L),
  + Deletes the previous values from table
  + Insert the Values fetched from Input file into Table ‘table\_L’ if the line starts with ‘L’
  + Also keeps the inserted entry into the ‘C:\SageWS\TEMP\LOUT.txt’.

Fig: Deleting the previous header details from Database.



Fig: Inserting the line details into Database.

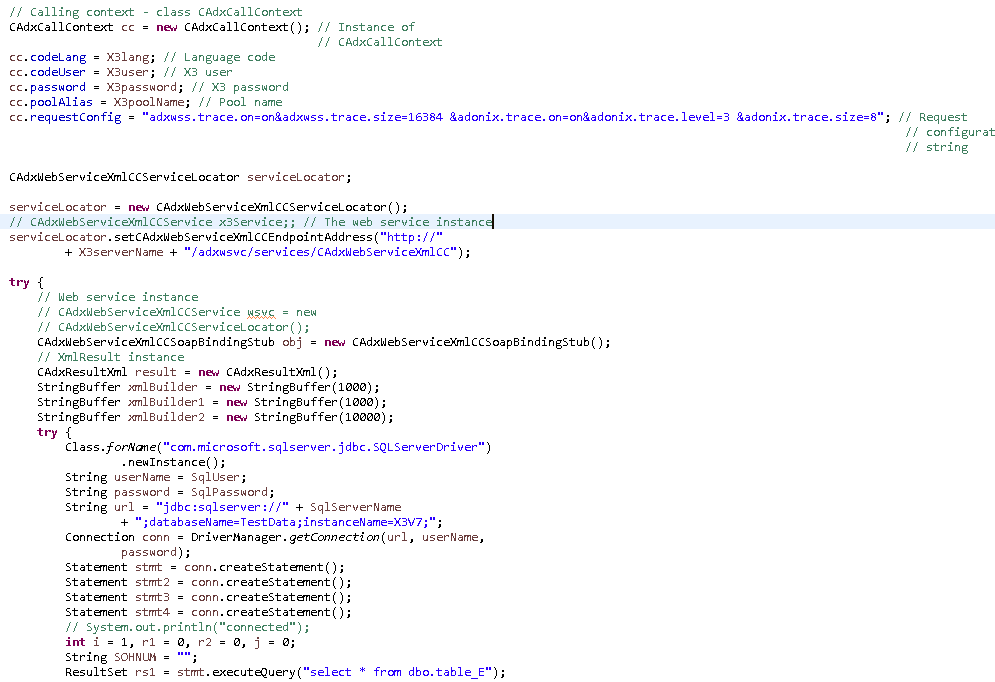


After These Operations we are deciding that which record we want to Create ie, Sales Order, Work Order etc.

**For Sales Order,**

* Make on object of Order.java Class.
  + Calls execute method.
    - Execute Method
      * Reads following ‘X3 SERVER’ configuration details from “C:\\SageWS\\CONFIG\\X3\_WS\_Config.txt”.
        + X3 server name
        + X3 poolname
        + X3 Lang
        + X3 User
        + X3 password
        + Sql Server name
        + Sql User
        + Sql Password
      * Create the CAdxCallContext.java Object and pass the configuration details into that object.
      * Create the connection to the Web Service using CAdxWebServiceXmlCCServiceLocator.java class.
      * This is achieved by calling, setCAdxWebServiceXmlCCEndpointAddress("http://"+ X3serverName + "/adxwsvc/services/CAdxWebServiceXmlCC") method.
      * This method will connect to the address that given as parameter.

Fig: Creating the context for connection and making the connection using Address.



* Fetch all values from header table in X3 server (table\_E).
* Then create the header part xml file.
* After that fetch the lines from table (table\_L)
* Then create the Line part xml file.

Fig: Creating the header part xml from table ‘table\_E’



Fig: Creating the lines part xml from table ‘table\_L’



* After Creating the xml file we have to call the save method of CAdxWebServiceXmlCCSoapBindingStub.java for creating the Order in sage X3.
  + This is the default adonix method for calling the sage X3 Web service’s create method.
  + save, delete, read, query, getDescription, modify, deleteLines, insertLines etc are the important methods available in this Class.
  + The parameters are context that we created, Public name of the web service, and the xml file that we created.
  + The output of this save method will be another xml file of type CAdxResultXml.java.
  + We have to convert this xml to String first and then to Document.
  + Then we will convert this to original xml file using TransformerFactory.transform (source, destination) method.
  + We will save the resulting xml file for getting the Status, Order Id’s that generated.
  + File is saved at ‘C:\\SageWS\\OUTPUT\\XML OUTPUT’ folder.
  + Then we will iterate through the xml and gets the Sales Order Header Number (‘SOHNUM’) that is generated.
  + After getting the Order Id we are saving the details in an external file (‘C:\\SageWS\\OUTPUT\\X3\_SO\_Output.txt’) for quick reference.

Fig: Calling the save method for creating the order through web service and creating the output xml.



Fig: save method implementation in CAdxWebServiceXmlCCSoapBindingStub.java.



Fig: important methods available in CAdxWebServiceXmlCCSoapBindingStub.java.

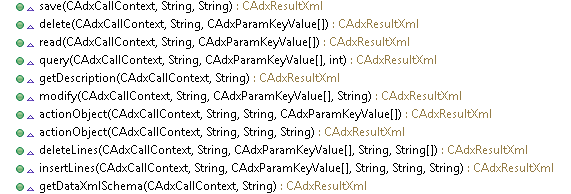


Fig: Fetching the xml file and getting the Sales order Ids generated.



Fig: Saving the output in a text file.



So we can use the newly generated Order Id’s to search in sage X3 application.