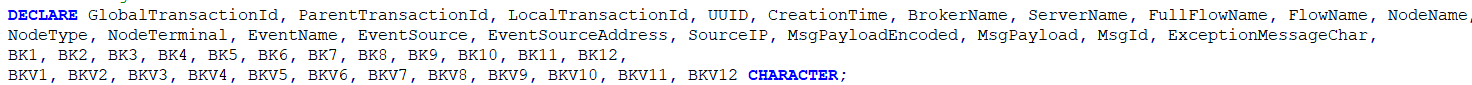
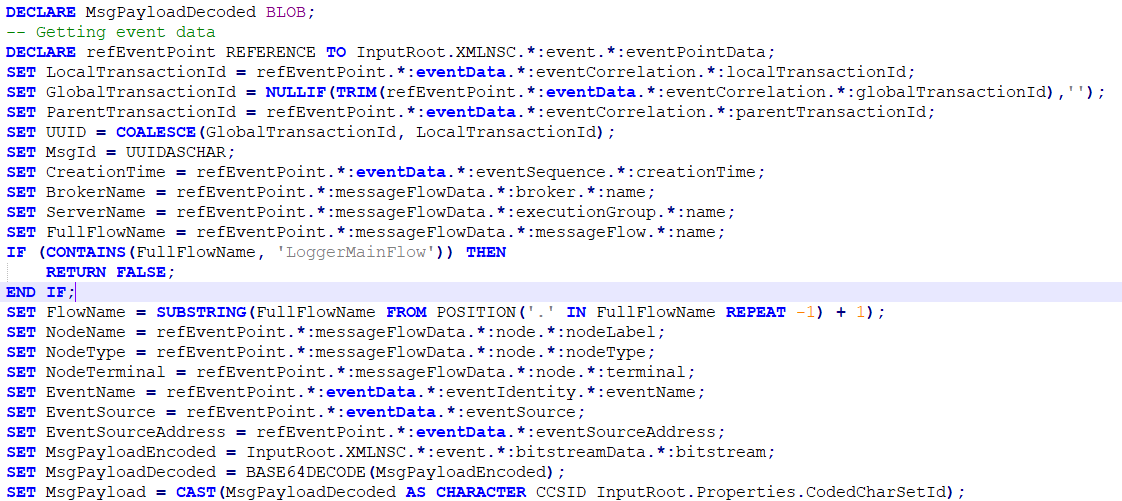
**Logger Documentation**

# 1.1 Variable Declarations

The logger starts by declaring the following character variables to store event information, BK means Business Key, and BKV means Business Key Value:

****

A blob variable named MsgPayloadDecoded is declared to hold the decoded message Payload, then the variables declared are all set to the corresponding event data in the input message tree as shown below, during the setting process the logger will check if the name of the flow that fired the event is the logger itself, then it will not log the event else it continues normally.

****

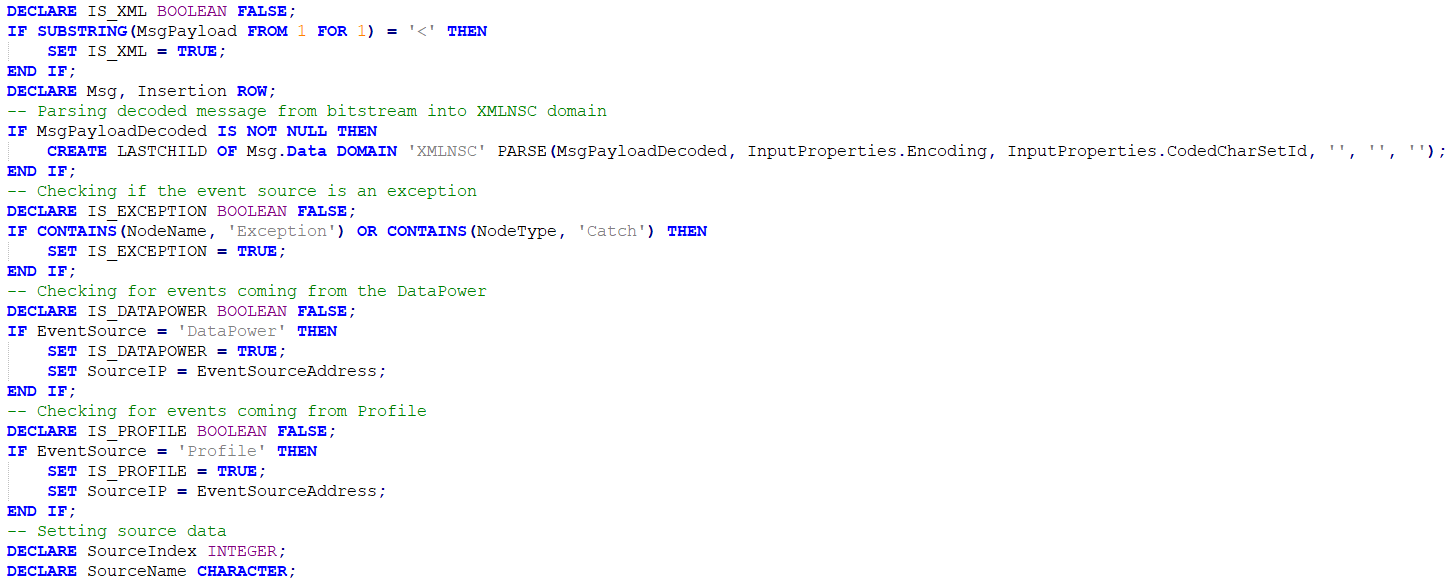
# 1.2 Detect Event Source (IIB, DP, Profile)

The logger checks the payload to ensure it is in xml format as logger can log XML and non XML messages, 2 rows are created, the Msg row Hold the payload and the Insertion row will hold the result of the query that will be sent to the database.

If the decoded message payload is not null it will be parsed to XMLNSC domain and added as last child of the element Data in Msg row. The logger then creates Boolean variable named IS\_EXCEPTION, then it will check if the event source is an exception or not and based on the result will set the variable.

Then two other Boolean variables named IS\_DATAPOWER and IS\_PROFILE are created, the logger will check where the event is coming from by checking the EventSource variable and based on the result will update the Boolean variable and will set the SourceIP variable with the ip address of the client that triggered the event.

Two variables SourceIndex and SourceName are created, the first will tell us at which stage in the flow the message is at, and the second will tell us the name of the stage.

****

# 1.3 Select Service Entry From Property File & Detect Event Type (REQ, IREQ, IRES, RES)

If the event source is not a profile, a row named Services is declared, the logger then retrives the service logging property from the logger property file by searching for a matching flow name or Discriminator(Parent Transaction ID) and sets the details retrieved in the Services row. if no matching flowname or discriminator is found then nothing will be logged, if a flow name is found but no exact discriminator is found(discriminator value is “ALL”) then the results are queried again from the Services row and are set into an enviroment variable named ServiceProp, if again no results are found then the very first element of the Services row will be set into the ServiceProp Variable.

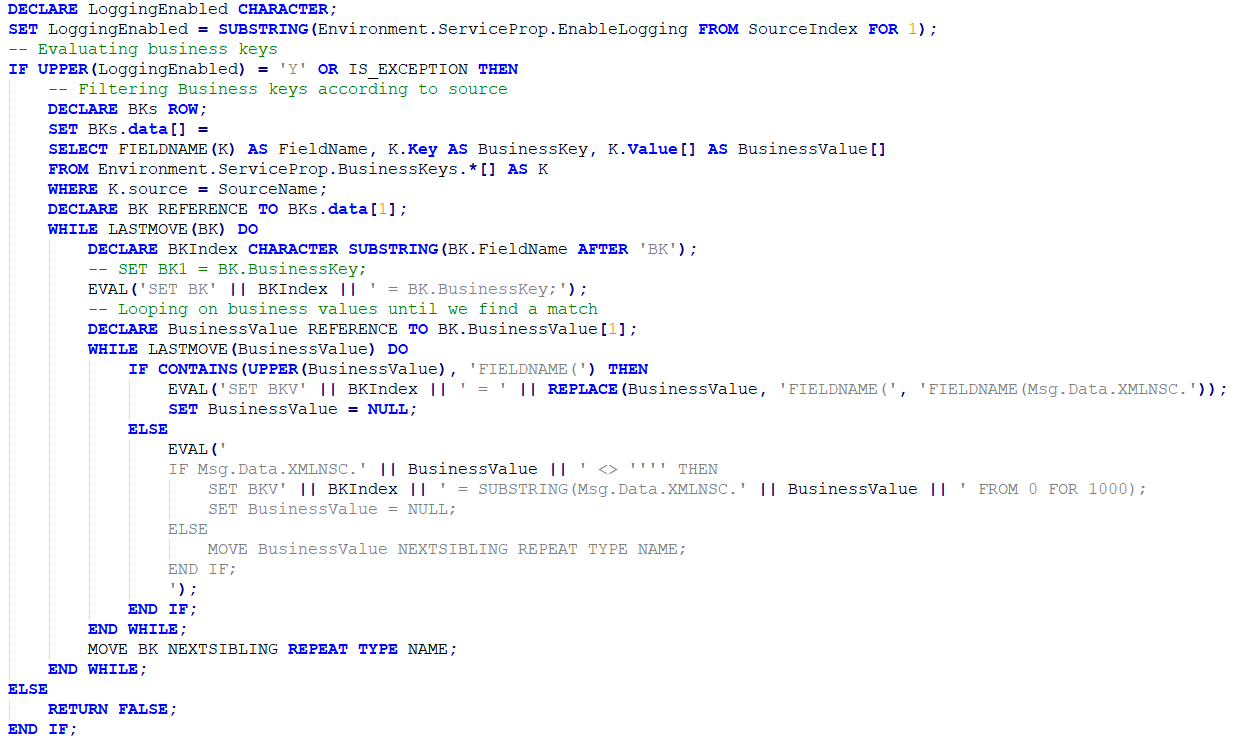
Now if the event comes from datapower, then logger will check for the type of the node from which the message payload comes, if it contains “\_REQ” or “\_RQ”, this means it is a request message payload, so the SourceIndex is set to 1 and SourceName to “REQ”, else if it contains “\_RES”,”\_RS”,”\_ERR”,”\_FAULT” then it is a response message payload so SourceIndex is set to “2” and SourceName is set to “RES”.

Now if the event comes directly from IIB, there will be 4 possible outcomes, that is because unlike datapower which has 2 stages only (Request and Response), IIB has 4 stages(Request, Internal Request, Internal Response, Response). If the node type contains “Input” only or the event is an exception in the first place, then SourceIndex is set to “1” and SourceName is set to “REQ” meaning it is a request message payload, if the node type contains “Reply” then SourceIndex is set to “4” and SourceName is set to “RES” meaning it is a response message payload, if the node type contains “Request” and “In” at the same time, then SourceIndex is set to “2” and SourceName is set to “IREQ” meaning it is an internel request message payload, last case is if it is an internal response message payload, which will set the SourceIndex and SourceName to “3” and “IRES” respectivly.

Graphical user interface, text, application, email

Description automatically generated

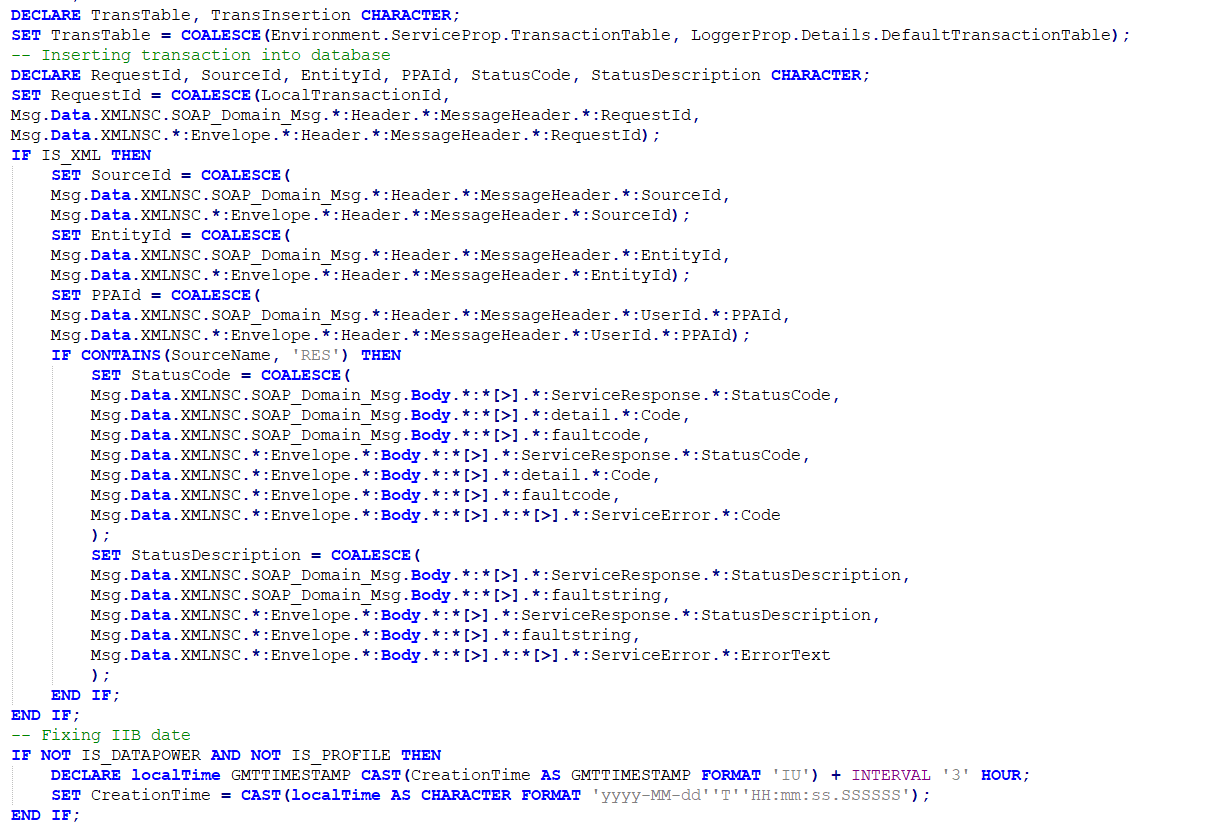
# 1.4 Business keys Evaluations

After the logger determined the service flow name and from which stage the message payload is coming from, it will check if logging is enabled for it or not, a character variable named LoggingEnabled is created, then logger will retrive EnableLogging Property from the ServiceProp variable that contains the service logging details we retrieved earlier from the logger property file. If LoggingEnabled equals “Y” then the logger proceeds, else it does not log.

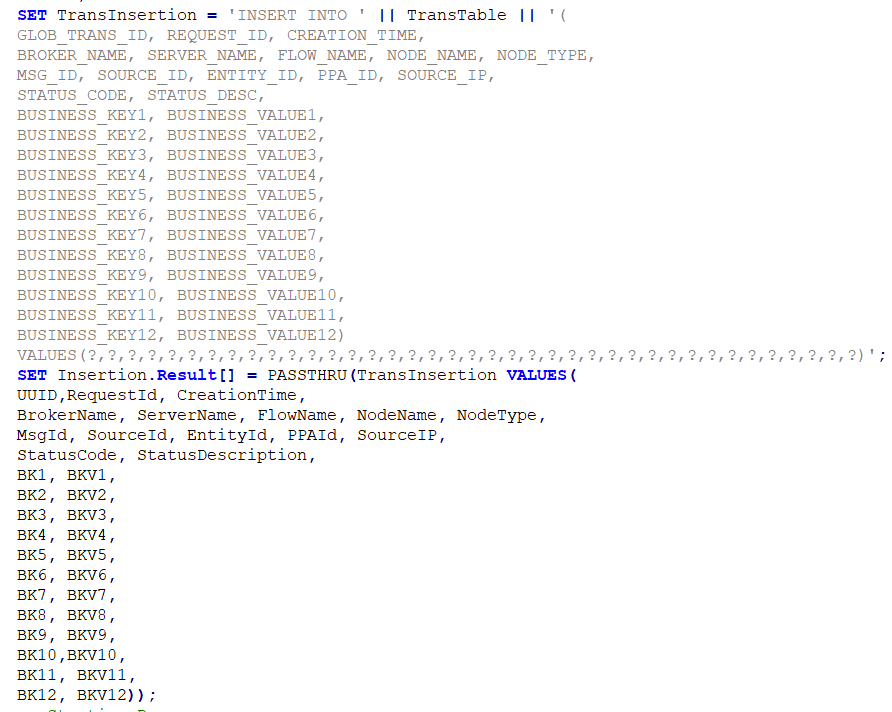
When it proceeds, it will retrieve the business keys and their values, first a row named BKs is created, inside it a field named data is added, it will hold the Field name, the business key itself, and its value that are retrived from the BusinessKeys field in the ServiceProp variable by a query that matches the SourceName. After retrieving the results, a reference to them is created named BK, the logger will loop over the results, in each loop, a character variable named BKIndex is created to hold the index of the buisness key, which is retrieved by taking a substring from the field name after the letters “BK”, then we use the EVAL function to set the business key dynamically with each loop, because the EVAL function returns what is inside the bracket as a single string to be excuted. Then we declare a reference to the Business value field in the BK row called BusinessValue, and will loop over the values in it until a matching value is found, this is done by checking if the BusinessValue contains “FIELDNAME(”, if it does then we use EVAL again to dynamically set BusinessValue of each index, we use REPLACE inside it to replace all “FIELDNAME{” in BusinessValue with “FIELDNAME(Msg.Data.XMLNSC” before setting it, then BusinessValue is set to null. If BusinessValue does not contain “FIELDNAME(”, then EVAL is used to pass an if condition that will checks if Msg.Data.XMLNSC.BusinessValue is not equal to ‘ ’ ’ ’, then BusinessValue is set to equal substring of Msg.Data.XMLNSC.BusinessValue FROM 0 FOR 1000, then BusinessValue is set to null, else loop continues.

# 1.5 Prepare Transaction Table Parameters

Two character variables are created, TransTable and TransInsertion, TransTable will be set to equal a specified database(TransactionTable) or a default database, this is done by using COALESCE function which returns the first element in it that is not null. Now we declare 5 character variables, RequestId, SourceId, EntityId, PPAId, StatusCode, and StatusDescription. RequestId is set using the COALESCE function to assign the first not null value from one of the sources shown in the code below, if the payload is in xml format, SourceId, EntityId, and PPAId are all set the same way using the COALESCE function to choose the first not null value from one of the sources as shown in the code below. If the payload is a response message, then we set the StatusCode and the StatusDescription using COALESCE function again as shown in the code below. If the message does not come from datapower and is not a profile, then it must be an iib message, so we have to fix the creation date variable before inserting it into the database, this is done by declaring a GMTTIMESTAMP varible named localtime, and then assigning to it the CreationTime variable after casting it to GMTTIMESTAMP with format ‘IU’ and adding 3 hours to it. The last step is setting the CreationTime equal to localtime after casting it to CHARACTER with format ‘yyyy-MM-dd’ ‘T’ “HH:mm:ss.SSSSSS”.

****

# 1.6 Transaction Table Insertion

Now after all variables have been assigned values successfully it is time to insert them into the database, first we are going to insert into the Transaction database, an insert SQL query is written and assigned to the TransInsertion variable, then we create a field called Result in the Insertion row we created earlier, this field will hold the result of the query inside TrasnInsertion after we excute it by sending it to the database using the PASSTHRU function, with this the values have been inserted successfully into the Transactions database.

# 1.7 Dump Table Insertion

Now we want to insert the message payload(Dump) into the Transactions\_Dump database, but before that the logger will check if dumping is enabled for this service and this stage, this is done by creating a character variable called DumpEnabled, then it is assigned a substring from the EnableDump Field in the ServiceProp variable, if it is equal to “Y” or the event is an exception or coming form a profile, then we declare two character variables, DumpTable and DumpInsertion, DumpTable is assigned either a specific dump database or a default dump database using the COALESCE function. If the event is an exception and coming from a profile, then set the ExceptionMessageChar equal to the MsgPayload, else if it is an exception not coming from a profile, then a reference to the complexContent field in the input message tree called refApplicationData is created. Last child of this path is set to null, a blob variable named EXCEPTIONBLOB is declared and assigned a generates bitstream of refApplicationData using ASBITSTREAM function, then ExceptionMessageChar is set equal to EXCEPTIONBLOB after casting it to CHARACTER and encoding it using CCSID 1208. Now it is time to insert the data into the Transactions\_Dump database, an insert SQL query is written and assigned to DumpInsertion, then the Result field created earlier in the Insertion row is used again to store the result of the query that was excuted by sending it to the database using the PASSTHRU function. With this the logging activity has been successfully completed.

****