Universe

And

Microsoft Sequel/MySQL

George Gallen

g_gallen@hotmail.com

Last Updated 04/06/2016

This Guide is to help with importing data from a Microsoft Sequel Server or a MySQL Server into a Universe Program. It is not meant for interactive communication with these servers, but strictly importing data. It currently does not have methods for writing data to these servers, it is read only.

Also, be advised, these routines do not incorporate any security functionality. If you put data directly from user input into the SQL statement, you can open yourself up for SQL Injection issues. Also with the MxSQL.EXECUTE.COMMAND – while this command is running, the username and password is visible to anyone who issues a 'ps' command or who has read access to the directory that the files are written to.

Last caveat – when this document was converted to a PDF the quotes were converted as well – making it very difficult to cut and paste the programming. Sorry about that.

Any questions/comments can be directed to me at:

g_gallen@hotmail.com

Thanks

George

Universe/Microsoft Sequel Integration

Requirements:

1. **Universe (Linux)**
2. **FreeTDS (OpenSource ODBC)**
3. **PERL**
4. **Linux ‘wc’ command**

This is a programmatic solution to query and import data from a Microsoft Sequel Server, it does not give you a direct ODBC connection to the SQL server from the command line. There is no expense (aside from time) to implement this solution.

This solution is based on Universe on Linux, and utilizes OpenSource FreeTDS package to interface with the Sequel Server. It then uses a PERL statement to eliminate cr/lf combinations, this can be accomplished by many means, I just chose to use a PERL statement. The unix command ‘wc’ is also used to determine how many lines are in the output data file – again this was the method I chose.

I have not attempted to implement this solution using Universe on a windows platform.

Feel free to modify these to suit your needs.

Last Updated 04/06/2016

Setting up FreeTDS (see http://www.freetds.org)

If you do not already have a tsql utility:

From Unix (as root):

yum install freetds (or) aptget install freetds

you should not need to change /etc/odbcinst.ini
you will need to change /etc/freetds.conf

format:

[SERVERNAME]
host = IP/domainname of server
port = 1433 (or port# used if non-standard)
client charset = UTF-8 (change as needed)
tds version = 8.0

To test:

$tsql -S SERVERNAME -U USERNAME –P PASSWORD

If you get the MSSQL “>” enter “quit” <ENTER> - you are good.

Last Updated 04/06/2016

Universe Subroutines

MSSQL.EXECUTE.COMMAND() – Will Execute the SQL command, setup an output file from MSSQL and return a Query Control Block (QCB).

MSSQL.RETURNHEADER() – When passed the QCB, will return a delimited string of the column names.

MSSQL.GETNEXTLINE() – When passed the QCB, will return a delimited string of the next row of data.

MSSQL.CLOSEFILE() – When passed the QCB, will close the files used to read the SQL output data.

The QBC is a dimensioned Array which contains elements related to the specific SQL Query. Each QBC will contain all the elements that the subroutines will use. Since each QBC is self contained, you may have multiple QBC's (multiple SQL queries) in use at the same time if needed.

Last Updated 04/06/2016

QBC Layout (Query Control Block)

QCB(1)=F.FCB
QCB(2)=F.TMP
QCB(3)=TAB
QCB(4)=FILEOPEN
QCB(5)=NUMLINES
QCB(6)=ERR
QCB(7)=CHANNEL
QCB(8)=USRNO
QCB(9)=NUMOFFIELDS
QCB(10)=FIRSTFIELD
QCB(11)=DELETEFILES
QCB(12)=CNV
QCB(13)=DATAROWS

F.FCB - FCB of the SQL output file
F.TMP - FCB of path /tmp
TAB - Character used by MSSQL for field delimiting
FILEOPEN - Contains File Status (1=open, 0=closed)
NUMLINES - NUMLINES Created - # of lines in MSSQL Output File (not # rows)
ERR - 0 if no errors detected, 1 otherwise
CHANNEL - Current CHANNEL being used for this query – alpha or numeric
USRNO - Current @USERNO:"-":CHANNEL
NUMOFFIELDS - number of fields+1 in query
FIRSTFIELD - First columnname in query output
DELETEFILES - Flag to delete files when completed
CNV - Convert "NULL" to "" in Column Values
DATAROWS - Estimated # of data rows - adjusted as rows are read

Last Updated 04/06/2016

Sample Universe Base Program

```
   OPEN "","SOMEFILE" TO F.SOMEFILE ELSE STOP "NO SOMEFILE"
   DIM QCB1(20)
   PROGNAME="SAMPLE-PROGRAM" ; DELYN=0; CONVNULL=1; BADF=""
   GOSUB 1300 ; HEAD=DATALIN
   PRINT QCB1(5):" ROWS.":
   CONVERT CHAR(253) TO CHAR(254) IN HEAD
   WRITE HEAD ON F.SOMEFILE,"HEAD"
   NUMB=0
*
   LOOP
      GOSUB 2100 ; LIN=DATALIN
      IF QCB1(4)=0 THEN EXIT  ; * File is now flagged as NOT OPEN
      NUMB=NUMB+1
      CONVERT CHAR(253) TO CHAR(254) IN LIN
      CID=LIN<1>
      WRITE LIN ON F.SOMEFILE,CID
   REPEAT
*
   PRINT "Done!"
   IF DCOUNT(BADF,CHAR(254)) > 0 THEN
      PRINT "# OF BAD FIELDS = ":DCOUNT(BADF,CHAR(254))
      CALL *EMAILFILE(BADF,"EMAILADDRESS","Bad Lines","*")
   END
   PRINT QCB1(13):" AND ":NUMB  ; * display calculated vs actual row counts
   STOP
*
1300:
   SQLFIELDS="ContactId, username, address"
   SQLFILE="Contact c inner join Addresses a on (c.ContactId = a.ContactId)"
   SQLWHERE="WHERE c.StatusCode = 1"
   SQLCMD="select ":SQLFIELDS:" from ":SQLFILE:" ":SQLWHERE
   FIRSTFIELD="ContactId"
   PRINT "Getting CONTACts..":
   GOSUB 2000
   PRINT "Read...Processing...":
   RETURN
*
2000:
   CALL *MSSQL.EXECUTE.COMMAND(SQLCMD,FIRSTFIELD,"1",PROGNAME,DELYN,CONVNULL,MAT QCB1)
   CALL *MSSQL.RETURNHEADER(DATALIN,MAT QCB1)
   RETURN
*
2100:
   CALL *MSSQL.GETNEXTLINE(DATALIN,BADF,MAT QCB1)
   RETURN
```

Last Updated 04/06/2016

MSSQL.EXECUTE.COMMAND()

```
SUBROUTINE MSSQL.EXECUTE.COMMAND(SQLCMD, FIRSTFIELD, CHANNEL, PROGNAME, DELETEFILES, CNV, QCB)
*
* MSSQL.EXECUTE.COMMAND - This routine when passed SQLCMD, CHANNEL and PROGNAME
*          will return an Array of elements for the Query Control Block. other
*          routines can use these elements as well. By keeping these elments
*          encapsulated into one Array - mulitple Queries can be in use at
*          the same time.
************************************************************************
*
   DIM QCB(20)
   MAT QCB=""
*
   USERNAME="sql-login-name"  ; * If you want to make routine more flexible
   PASSWORD="sql-login-password" ; * you can add USERNAME, PASSWORD and DATABASE
   DATABASENAME="sql-databasename" ; * to the list of Parameters sent to the routine
   SERVNAME="SERVERNAME"  ; * This name is from the /etc/freetds.conf file
*
   DIRECTORY="/tmp"  ; * unix directory where files will be written to
   TOEMAIL="contactemail"
   QCB(6)=1 ; * DEFAULT TO ERROR FOUND
    QCB(4)=0 ; * DEFAULT TO FILE NOT OPEN
*
   OPENPATH DIRECTORY TO F.TMP ELSE RETURN
*
   TAB=CHAR(9)
   QCB(3)=TAB
   AM=CHAR(254); VM=CHAR(253)
*
   BLANKLINE=TAB:TAB
   DEBUG=0
*
   NUMLINES=0
   USRNO=@USERNO:"-":CHANNEL
   QCB(10)=FIRSTFIELD
   QCB(11)=DELETEFILES
   QCB(12)=CNV
*
   SQLLINE="tsql -S ":SERVNAME:" -U ":USERNAME:" -P ":PASSWORD
   SQLLINE=SQLLINE:" 2>&1 > ":DIRECTORY:"/output":USRNO:" <<EOF"
*
   SQLLINE<-1>="use ":DATABASENAME:";"
   SQLLINE<-1>=SQLCMD
   SQLLINE<-1>="go"
   SQLLINE<-1>="bye"
   SQLLINE<-1>="EOF"
   SQLLINE<-1>="/bin/cp ":DIRECTORY:"/output":USRNO:" ":DIRECTORY:"/holding":USRNO
   SQLLINE<-1>="perl -pe 's/\r\n/ /g' <":DIRECTORY:"/holding":USRNO:" >
":DIRECTORY:"/output":USRNO
   SQLLINE<-1>="/bin/rm ":DIRECTORY:"/holding":USRNO
*
   WRITE SQLLINE ON F.TMP,"TSQLCMD":USRNO
   CMD1=DIRECTORY:"/TSQLCMD":USRNO
   ERR=0 ; FILEOPEN=1
   EXECUTE \SH -c "\:CMD1:\"\ CAPTURING ERRORS
   IF ERRORS#"" THEN
      EMAILBODY= SQLLINE:AM:AM:ERRORS
      EMAILSUBJ="[":SERVNAME:"] (":PROGNAME:") Errors Executing"
      CALL *EMAILFILE(EMAILBODY,TOEMAIL,EMAILSUBJ,"*")
      ERR=1; FILEOPEN=0; F.FCB=""
   END
   IF QCB(11)=1 THEN
```

```
      DELETE F.TMP,"TSQLCMD":USRNO
   END
*
   CMD2="/bin/wc -l ":DIRECTORY:"/output":USRNO
   EXECUTE \SH -c "\:CMD2:\"\ CAPTURING LINECNT
   NUMLINES=FIELD(LINECNT<1>," ",1)
*
   IF ERR=0 THEN
      OPENSEQ DIRECTORY:"/output":USRNO TO F.FCB ELSE
         F.FCB=""; ERR=1; FILEOPEN=0
      END
   END
*
   QCB(1)=F.FCB
   QCB(2)=F.TMP
   QCB(3)=TAB
   QCB(4)=FILEOPEN
   QCB(5)=NUMLINES
   QCB(6)=ERR
   QCB(7)=CHANNEL
   QCB(8)=USRNO
   QCB(9)=0
   QCB(13)=""
*
   RETURN
*
STOP
END
```

MSSQL.RETURNHEADER()

```
SUBROUTINE MSSQL.RETURNHEADER(DATALIN, QCB)
*
* MSSQL.RETURNHEADER - Passed the Queue Control Block and is returned DATALIN
*                      which will contain the Header Row - separated by @VM.
*                      NUMOFFIELDS is updated in the QCB based on the header row
*                      DATAROWS is updated in QCB - although not always accurate
*                        if any of the datarows contain embedded CR this will
*                        cause the DATAROW count to be higher by the amount of
*                        CRs embedded. This value will be reduced by the routine
*                        CRM.GETNEXTLINE() if it has to join any data rows.
*************************************************************************
*
   DIM QCB(20)
   AM=CHAR(254); VM=CHAR(253)
*
   STARTCHECK=0;
   F.FCB=QCB(1)
   TAB=QCB(3)
   FILEOPEN=QCB(4)
   ERR=QCB(6)
   FIRSTFIELD=QCB(10)
*
   BLANKLINE=TAB:TAB
*
   YDATA=""
   DATALIN=""
   NUMOFFIELDS=0
   ROWCNT=0
*
   IF ERR=0 THEN
      LOOP
         READSEQ PLINE FROM F.FCB ELSE EXIT
         ROWCNT=ROWCNT+1
         IF PLINE=BLANKLINE THEN CONTINUE
         IF STARTCHECK=0 THEN
            IF PLINE[1,LEN(FIRSTFIELD)]=FIRSTFIELD THEN
               STARTCHECK=1
               YDATA=CHANGE(PLINE,TAB,VM)
               DATALIN=YDATA
               NUMOFFIELDS=DCOUNT(DATALIN,VM)
               EXIT
            END
            CONTINUE
         END
      REPEAT
   END
*
   IF DATALIN#"" THEN FILEOPEN=1 ; ERR=0; DATAROWS=QCB(5)-ROWCNT-1 ELSE DATAROWS=0
*
   QCB(1)=F.FCB ; * update the QCB with updated file control block
   QCB(6)=ERR
   QCB(4)=FILEOPEN
   QCB(9)=NUMOFFIELDS
   QCB(13)=DATAROWS ; * Accurate if there are no embedded <CR> in data fields in the data rows
*
   RETURN
*
STOP
END
```

MSSQL.GETNEXTLINE()

```
SUBROUTINE MSSQL.GETNEXTLINE(DATALIN,BADF,QCB)
*
* MSSQL.GETNEXTLINE - Passed BADF dynamic array, QCB queue control block
*                     returned in DATALIN is the next row of data.
*                     Rows are concatenated together until there are the correct
*                     number of Columns (based on NUMOFFIELDS) to combat the
*                     is the issue of broken output data from embedded CR in
*                     column data. If the number of columns exceeds NUMOFFIELDS
*                     the entire line is added to the BADF dynamic array and
*                     skipped. If any lines are joined the value of DATAROWS
*                     will be reduced for a more accurate row count. When two
*                     data rows are joined because of an embedded CR the CR is
*                     changed to a space.
*************************************************************************
*
   DIM QCB(20)
   AM=CHAR(254); VM=CHAR(253)
*
   F.FCB=QCB(1)
   TAB=QCB(3)
   FILEOPEN=QCB(4)
   ERR=QCB(6)
   NUMOFFIELDS=QCB(9)
   CNV=QCB(12)
*
   DATALIN=""
   KEEPIT=""
*
   IF FILEOPEN=0 THEN RETURN
*
   LOOP
      READSEQ PLINE FROM F.FCB ELSE
         DATALIN=""
         CALL *MSSQL.CLOSEFILE(MAT QCB)
         RETURN
      END
      IF PLINE[1,2]="1>" THEN CONTINUE
      PTEMP=CHANGE(PLINE,TAB,VM)
      LINF=DCOUNT(PTEMP,VM)
      IF CNV=1 THEN
         FOR T=1 TO LINF
            IF PTEMP<1,T>="NULL" THEN PTEMP<1,T>=""
         NEXT T
      END
      DATALIN=PTEMP
*
      IF KEEPIT#"" THEN
         DATALIN=KEEPIT:" ":DATALIN
          QCB(13)=QCB(13)-1
      END
      LINF=DCOUNT(DATALIN,VM)
      IF LINF>NUMOFFIELDS THEN
         BADF<-1>=DATALIN
          KEEPIT=""
          QCB(13)=QCB(13)-1
          CONTINUE
      END
      IF LINF#NUMOFFIELDS THEN
         KEEPIT=DATALIN
          CONTINUE
      END
      EXIT
   REPEAT
*
   QCB(1)=F.FCB  ; * write updated file control block to the QCB
   RETURN
STOP
END
```

Last Updated 04/06/2016

MSSQL.CLOSEFILE() - Generally only called by MSSQL.GETNEXTLINE()

```
SUBROUTINE MSSQL.CLOSEFILE(QCB)
*
* MSSQL.CLOSEFILE Will execute a CLOSESEQ on the F.FCB that is passed via QCB
*                 will also change the FILEOPEN status to 0 (Closed) and then
*                 proceed to delete the temporary file in /tmp based on USRNO
*                 as the unique identifier of the filename. Will also execute
*                 a CLOSE on the F.TMP control block to close the /tmp file
*************************************************************************
*
   DIM QCB(20)
*
   F.FCB=QCB(1)
   F.TMP=QCB(2)
   FILEOPEN=QCB(4)
   USRNO=QCB(8)
   CHANNEL=QCB(7)
*
   CLOSESEQ F.FCB
   FILEOPEN=0
   IF QCB(11)=1 THEN
      DELETE F.TMP,"output":USRNO
   END
   CLOSE F.TMP
*
   QCB(1)=""
   QCB(2)=""
   QCB(4)=FILEOPEN
*
   RETURN
STOP
END
```

Universe/MySQL Integration

Requirements:

1. **Universe (Linux)**
2. **PERL**
3. **Linux ‘wc’ command**

This is a programmatic solution to query and import data from a MySQL Server, it does not give you a direct ODBC connection to the SQL server from the command line. There is no expense (aside from time) to implement this solution.

This solution is based on Universe on Linux. It then uses a PERL statement to eliminate cr/lf combinations, this can be accomplished by many means, I just chose to use a PERL statement. The unix command ‘wc’ is also used to determine how many lines are in the output data file – again this was the method I chose.

I have not attempted to implement this solution using Universe on a windows platform.

Feel free to modify these to suit your needs.

Last Updated 04/06/2016

Universe Subroutines

MYSQL.EXECUTE.COMMAND() – Will Execute the SQL command, setup an output file from MYSQL and return a Query Control Block (QCB).

MYSQL.RETURNHEADER() – When passed the QCB, will return a delimited string of the column names.

MYSQL.GETNEXTLINE() – When passed the QCB, will return a delimited string of the next row of data.

MYSQL.CLOSEFILE() – When passed the QCB, will close the files used to read the SQL output data.

The QBC is a dimensioned Array which contains elements related to the specific SQL Query. Each QBC will contain all the elements that the subroutines will use. Since each QBC is self contained, you may have multiple QBC's (multiple SQL queries) in use at the same time if needed.

Last Updated 04/06/2016

QBC Layout (Query Control Block)

QCB(1)=F.FCB
QCB(2)=F.TMP
QCB(3)=TAB
QCB(4)=FILEOPEN
QCB(5)=NUMLINES
QCB(6)=ERR
QCB(7)=CHANNEL
QCB(8)=USRNO
QCB(9)=NUMOFFIELDS
QCB(10)=FIRSTFIELD
QCB(11)=DELETEFILES
QCB(12)=CNV
QCB(13)=DATAROWS

F.FCB - FCB of the SQL output file
F.TMP - FCB of path /tmp
TAB - Character used by MSSQL for field delimiting
FILEOPEN - Contains File Status (1=open, 0=closed)
NUMLINES - NUMLINES Created - # of lines in MSSQL Output File (not # rows)
ERR - 0 if no errors detected, 1 otherwise
CHANNEL - Current CHANNEL being used for this query – alpha or numeric
USRNO - Current @USERNO:"-":CHANNEL
NUMOFFIELDS - number of fields+1 in query
FIRSTFIELD - First columnname in query output
DELETEFILES - Flag to delete files when completed
CNV - Convert "NULL" to "" in Column Values
DATAROWS - Estimated # of data rows - adjusted as rows are read

Last Updated 04/06/2016

Sample Universe Base Program

```
   OPEN "","SOMEFILE" TO F.SOMEFILE ELSE STOP "NO SOMEFILE"
   DIM QCB1(20)
   PROGNAME="SAMPLE-PROGRAM" ; DELYN=0; CONVNULL=1; BADF=""
   GOSUB 1300 ; HEAD=DATALIN
   PRINT QCB1(5):" ROWS.":
   CONVERT CHAR(253) TO CHAR(254) IN HEAD
   WRITE HEAD ON F.SOMEFILE,"HEAD"
   NUMB=0
*
   LOOP
      GOSUB 2100 ; LIN=DATALIN
      IF QCB1(4)=0 THEN EXIT  ; * File is now flagged as NOT OPEN
      NUMB=NUMB+1
      CONVERT CHAR(253) TO CHAR(254) IN LIN
      CID=LIN<1>
      WRITE LIN ON F.SOMEFILE,CID
   REPEAT
*
   PRINT "Done!"
   IF DCOUNT(BADF,CHAR(254)) > 0 THEN
      PRINT "# OF BAD FIELDS = ":DCOUNT(BADF,CHAR(254))
      CALL *EMAILFILE(BADF,"EMAILADDRESS","Bad Lines","*")
   END
   PRINT QCB1(13):" AND ":NUMB  ; * display calculated vs actual row counts
   STOP
*
1300:
   SQLFIELDS="ContactId, username, address"
   SQLFILE="Contact c inner join Addresses a on (c.ContactId = a.ContactId)"
   SQLWHERE="WHERE c.StatusCode = 1"
   SQLCMD="select ":SQLFIELDS:" from ":SQLFILE:" ":SQLWHERE
   FIRSTFIELD="ContactId"
   PRINT "Getting CONTACts..":
   GOSUB 2000
   PRINT "Read...Processing...":
   RETURN
*
2000:
   CALL *MYSQL.EXECUTE.COMMAND(SQLCMD,FIRSTFIELD,"1",PROGNAME,DELYN,CONVNULL,MAT QCB1)
   CALL *MYSQL.RETURNHEADER(DATALIN,MAT QCB1)
   RETURN
*
2100:
   CALL *MYSQL.GETNEXTLINE(DATALIN,BADF,MAT QCB1)
   RETURN
```

Last Updated 04/06/2016

MYSQL.EXECUTE.COMMAND()

```
SUBROUTINE MYSQL.EXECUTE.COMMAND(SQLCMD, FIRSTFIELD, CHANNEL, PROGNAME, DELETEFILES, CNV, QCB)
*
* MYSQL.EXECUTE.COMMAND - This routine when passed SQLCMD, CHANNEL and PROGNAME
*          will return an Array of elements for the Query Control Block. other
*          routines can use these elements as well. By keeping these elments
*          encapsulated into one Array - mulitple Queries can be in use at
*          the same time.
*************************************************************************
*
   DIM QCB(20)
   MAT QCB=""
*
   USERNAME="sql-login-name"  ; * If you want to make routine more flexible
   PASSWORD="sql-login-password" ; * you can add USERNAME, PASSWORD and DATABASE
   DATABASENAME="sql-databasename" ; * to the list of Parameters sent to the routine
   SERVNAME="MYSQL"
*
   DIRECTORY="/tmp"  ; * unix directory where files will be written to
   TOEMAIL="contactemail"
   QCB(6)=1 ; * DEFAULT TO ERROR FOUND
    QCB(4)=0 ; * DEFAULT TO FILE NOT OPEN
*
   OPENPATH DIRECTORY TO F.TMP ELSE RETURN
*
   TAB=CHAR(9)
   QCB(3)=TAB
   AM=CHAR(254); VM=CHAR(253)
*
   BLANKLINE=TAB:TAB
   DEBUG=0
*
   NUMLINES=0
   USRNO=@USERNO:"-":CHANNEL
   QCB(10)=FIRSTFIELD
   QCB(11)=DELETEFILES
   QCB(12)=CNV
*
   SQLLINE="mysql --user=":USERNAME:" --password=":PASSWORD:" -e ":'"':SQLCMD:'"':" ":DATABASE
   SQLLINE=SQLLINE:" > ":DIRECTORY:"/output":USRNO
*
   SQLLINE<-1>="/bin/cp ":DIRECTORY:"/output":USRNO:" ":DIRECTORY:"/holding":USRNO
   SQLLINE<-1>="perl -pe 's/\r\n/ /g' <":DIRECTORY:"/holding":USRNO:" >
":DIRECTORY:"/output":USRNO
   SQLLINE<-1>="/bin/rm ":DIRECTORY:"/holding":USRNO
*
   WRITE SQLLINE ON F.TMP,"MYSQLCMD":USRNO
   CMD1=DIRECTORY:"/MYSQLCMD":USRNO
   ERR=0 ; FILEOPEN=1
   EXECUTE \SH -c "\:CMD1:\"\ CAPTURING ERRORS
   IF ERRORS#"" THEN
      EMAILBODY= SQLLINE:AM:AM:ERRORS
      EMAILSUBJ="[":SERVNAME:"] (":PROGNAME:") Errors Executing"
      CALL *EMAILFILE(EMAILBODY,TOEMAIL,EMAILSUBJ,"*")
      ERR=1; FILEOPEN=0; F.FCB=""
   END
   IF QCB(11)=1 THEN
      DELETE F.TMP,"MYSQLCMD":USRNO
   END
*
   CMD2="/bin/wc -l ":DIRECTORY:"/output":USRNO
   EXECUTE \SH -c "\:CMD2:\"\ CAPTURING LINECNT
```

```
   NUMLINES=FIELD(LINECNT<1>," ",1)
*
   IF ERR=0 THEN
      OPENSEQ DIRECTORY:"/output":USRNO TO F.FCB ELSE
         F.FCB=""; ERR=1; FILEOPEN=0
      END
   END
*
   QCB(1)=F.FCB
   QCB(2)=F.TMP
   QCB(3)=TAB
   QCB(4)=FILEOPEN
   QCB(5)=NUMLINES
   QCB(6)=ERR
   QCB(7)=CHANNEL
   QCB(8)=USRNO
   QCB(9)=0
   QCB(13)=""
*
   RETURN
*
STOP
END
```

MYSQL.RETURNHEADER()

```
SUBROUTINE MYSQL.RETURNHEADER(DATALIN, QCB)
*
* MYSQL.RETURNHEADER - Passed the Queue Control Block and is returned DATALIN
*                      which will contain the Header Row - separated by @VM.
*                      NUMOFFIELDS is updated in the QCB based on the header row
*                      DATAROWS is updated in QCB - although not always accurate
*                        if any of the datarows contain embedded CR this will
*                        cause the DATAROW count to be higher by the amount of
*                        CRs embedded. This value will be reduced by the routine
*                        CRM.GETNEXTLINE() if it has to join any data rows.
*************************************************************************
*
   DIM QCB(20)
   AM=CHAR(254); VM=CHAR(253)
*
   STARTCHECK=0;
   F.FCB=QCB(1)
   TAB=QCB(3)
   FILEOPEN=QCB(4)
   ERR=QCB(6)
   FIRSTFIELD=QCB(10)
*
   BLANKLINE=TAB:TAB
*
   YDATA=""
   DATALIN=""
   NUMOFFIELDS=0
   ROWCNT=0
*
   IF ERR=0 THEN
      LOOP
         READSEQ PLINE FROM F.FCB ELSE EXIT
         ROWCNT=ROWCNT+1
         IF PLINE=BLANKLINE THEN CONTINUE
         IF STARTCHECK=0 THEN
            IF PLINE[1,LEN(FIRSTFIELD)]=FIRSTFIELD THEN
               STARTCHECK=1
               YDATA=CHANGE(PLINE,TAB,VM)
               DATALIN=YDATA
               NUMOFFIELDS=DCOUNT(DATALIN,VM)
               EXIT
            END
            CONTINUE
         END
      REPEAT
   END
*
   IF DATALIN#"" THEN FILEOPEN=1 ; ERR=0; DATAROWS=QCB(5)-ROWCNT ELSE DATAROWS=0
*
   QCB(1)=F.FCB ; * update the QCB with updated file control block
   QCB(6)=ERR
   QCB(4)=FILEOPEN
   QCB(9)=NUMOFFIELDS
   QCB(13)=DATAROWS ; * Accurate if there are no embedded <CR> in data fields in the data rows
*
   RETURN
*
STOP
END
```

MYSQL.GETNEXTLINE()

```
SUBROUTINE MYSQL.GETNEXTLINE(DATALIN,BADF,QCB)
*
* MYSQL.GETNEXTLINE - Passed BADF dynamic array, QCB queue control block
*                     returned in DATALIN is the next row of data.
*                     Rows are concatenated together until there are the correct
*                     number of Columns (based on NUMOFFIELDS) to combat the
*                     is the issue of broken output data from embedded CR in
*                     column data. If the number of columns exceeds NUMOFFIELDS
*                     the entire line is added to the BADF dynamic array and
*                     skipped. If any lines are joined the value of DATAROWS
*                     will be reduced for a more accurate row count. When two
*                     data rows are joined because of an embedded CR the CR is
*                     changed to a space.
*************************************************************************
*
   DIM QCB(20)
   AM=CHAR(254); VM=CHAR(253)
*
   F.FCB=QCB(1)
   TAB=QCB(3)
   FILEOPEN=QCB(4)
   ERR=QCB(6)
   NUMOFFIELDS=QCB(9)
   CNV=QCB(12)
*
   DATALIN=""
   KEEPIT=""
*
   IF FILEOPEN=0 THEN RETURN
*
   LOOP
      READSEQ PLINE FROM F.FCB ELSE
         DATALIN=""
         CALL *MYSQL.CLOSEFILE(MAT QCB)
         RETURN
      END
      IF PLINE[1,2]="1>" THEN CONTINUE
      PTEMP=CHANGE(PLINE,TAB,VM)
      LINF=DCOUNT(PTEMP,VM)
      IF CNV=1 THEN
         FOR T=1 TO LINF
            IF PTEMP<1,T>="NULL" THEN PTEMP<1,T>=""
         NEXT T
      END
      DATALIN=PTEMP
*
      IF KEEPIT#"" THEN
         DATALIN=KEEPIT:" ":DATALIN
          QCB(13)=QCB(13)-1
      END
      LINF=DCOUNT(DATALIN,VM)
      IF LINF>NUMOFFIELDS THEN
         BADF<-1>=DATALIN
          KEEPIT=""
          QCB(13)=QCB(13)-1
          CONTINUE
      END
      IF LINF#NUMOFFIELDS THEN
         KEEPIT=DATALIN
          CONTINUE
      END
      EXIT
   REPEAT
*
   QCB(1)=F.FCB  ; * write updated file control block to the QCB
   RETURN
STOP
END
```

Last Updated 04/06/2016

MYSQL.CLOSEFILE() - Generally only called by MYSQL.GETNEXTLINE()

```
SUBROUTINE MYSQL.CLOSEFILE(QCB)
*
* MYSQL.CLOSEFILE Will execute a CLOSESEQ on the F.FCB that is passed via QCB
*                 will also change the FILEOPEN status to 0 (Closed) and then
*                 proceed to delete the temporary file in /tmp based on USRNO
*                 as the unique identifier of the filename. Will also execute
*                 a CLOSE on the F.TMP control block to close the /tmp file
*************************************************************************
*
   DIM QCB(20)
*
   F.FCB=QCB(1)
   F.TMP=QCB(2)
   FILEOPEN=QCB(4)
   USRNO=QCB(8)
   CHANNEL=QCB(7)
*
   CLOSESEQ F.FCB
   FILEOPEN=0
   IF QCB(11)=1 THEN
      DELETE F.TMP,"output":USRNO
   END
   CLOSE F.TMP
*
   QCB(1)=""
   QCB(2)=""
   QCB(4)=FILEOPEN
*
   RETURN
STOP
END
```