```
REM PreProc.bas
    ' Copyright (c) 2018 - 2019 WiseBox.Solutions
    ' All rights reserved.
    ' KKelbert@Wisebox.Solutions; support@wisebox.solutions
    '

REM 'ToDo List
    ' Pedigree
    ' 2019-01-05 Make use of Moxie.cfg file's 'Meta = ' line to modify functioning of this code
    '   - multiple options per line
    '   - Meta = Nightly-Name1: Value1; Nightly-Name2: Value2
    ' 2018-07-14 per: https://trello.com/c/QYZSsd2y/14-ongoing-maintenance-of-preproc-nightly
    '   Nightly log-rotate to limit zip file size by breaking each zip file into many. All less than 700MB.
    ' 2017-04-07 [Slack:Iaas] Kerry - Could you, urgently, add a .zip backup of the Work folder to your nightly backup system?
    ' https://trello.com/c/l6R95UwP/25-s012-automation
    ' 2016-09-28 repair grabbing biz.session.cnd as well as session.cnd
    ' ToDo - Final zPreProc dated.zip file to be placed into same folder as PreProc.conf -> [BackupRootFolder] setting
    ' ToDo - Replace / ReImpliment any and all hard coded links to follow PreProc.conf convention
    ' Test - MoxView path to be found in PreProc.conf -> [MoxViewFolder] setting
    ' PostProc Stats, Exceptions & Logs
    ' 'Fast' search for Countries
    ' Impliment ServiceAudit logic from old MasterSwitch.bas.2014-04-10-0414
    ' Broken: Gather and PreProcess "Exception" file dates and flag them that fall between lsLastStatsSnapshot & StampItNowLogFmt for later processing
    '   Mode: 2014-07-02 Tosses the files that have GPF'd based on time stamp
    '   Require: a command that will cause a GPF (for testing)
    '   repaired 2014-07-04-0504
    ' Done - needs testing sat - Move reports to do move on sat AM run only - very long file names still error
    'next glFinalResult = 9

REM Compiler Directives
    #COMPILER PBWIN 10
    #COMPILE EXE "PreProc"
    #OPTION VERSION5
    #DIM ALL
    #DEBUG ERROR ON
    #TOOLS ON
    #OPTION LARGEMEM32
    #INCLUDE "win32api.inc"

REM 'Macros
    MACRO mCommitLog()
        If IsFalse( glLogQuiet) THEN CommitLog lsHomeFldr, lsResult
    END MACRO

    MACRO mABS_SRC          = lsaFolderPath(%FolderRoot, llFolderPathNum) + "\" + lsaFolderPath(%FolderObject, llFolderPathNum)
    MACRO mABS_DST          = lsaFolderPath(%BackupRoot, llFolderPathNum) + "\" + lsaFolderPath(%FolderObject, llFolderPathNum)

    MACRO mDEST_BIN         = mABS_DST + "\Bin"
    MACRO mDEST_LOG         = mABS_DST + "\Log"
    MACRO mDEST_DBSTORE     = mABS_DST + "\DBStore"
    MACRO mDEST_REPORTS     = mABS_DST + "\Reports"
    MACRO mDEST_WORK        = mABS_DST + "\Work"

    MACRO mSRC__2           = mABS_SRC + "\DBStore\*"
    MACRO mDEST_2           = mABS_DST + "\DBStore\" + StampItDateHM + ".zip"
    MACRO mSRC__3           = mABS_SRC + "\Private\*"
    MACRO mDEST_3           = mABS_DST + "\Private.zip"
    MACRO mSRC__4           = mABS_SRC + "\Public\*"
    MACRO mDEST_4           = mABS_DST + "\Public.zip"
    MACRO mSRC__5           = mABS_SRC + "\Templates\*"
    MACRO mDEST_5           = mABS_DST + "\Templates.zip"
    MACRO mSRC__6           = mABS_SRC + "\Work\*"
    MACRO mDEST_6           = mABS_DST + "\Work\" + StampItDateHM + ".zip"

REM 'Consts
    $ProgramName            = "PreProc"
    $PreProcTempFile        = "PreProc.Ok2del"
    $PreProcListFile        = "PreProc.Include"
    $PreProcCfgName         = "PreProc.conf"

    $SEP_NUL                = CHR$(0)
    $SEP_SOH                = CHR$(1)
    $SEP_CRLF               = "<cnw-$crlf>"
    $SEP_CR                 = "<cnw-$cr>"
    $SEP_LF                 = "<cnw-$lf>"
    $SEP_TAB                = "<cnw-$tab>"

```

```
78      $FlagIfNotEmpty         = "$FlagIfNotEmpty"
79      $FlagIfEmpty            = "$FlagIfEmpty"
80      $FlagIfNotIncludes      = "$FlagIfNotIncludes"
81      $FlagIfIncludes         = "$FlagIfIncludes"
82      $Ignore                 = "$Ignore"
83
84      %dir_DBStore            = &h0001
85      %dir_History            = &h0002
86      %dir_Lib                = &h0004
87      %dir_Private            = &h0008
88      %dir_Public             = &h0010
89      %dir_Recv               = &h0020
90      %dir_Templates          = &h0040
91      %dir_Work               = &h0080
92
93      %file_Replace           = 0
94      %file_Append            = 1
95
96      %statsAlias             = 1
97      %statsStatTime          = 2     :  $statsStatTime       = "StatTime ="
98      %statsMoxieVer          = 3     :  $statsMoxieVer       = "MoxieVer ="
99      %statsHttpState         = 4     :  $statsHttpState      = "HttpState ="
100     %statsHttpsState        = 5     :  $statsHttpsState     = "HttpsState ="
101     %statsBkRecvState       = 6     :  $statsBkRecvState    = "BkRecvState ="
102     %statsSrvHits           = 7     :  $statsSrvHits        = "SrvHits ="
103     %statsSrvCurrent        = 8     :  $statsSrvCurrent     = "SrvCurrent ="
104     %statsSrvPeek           = 9     :  $statsSrvPeek        = "SrvPeek ="
105     %statsSrvCompleted      = 10    :  $statsSrvCompleted   = "SrvCompleted ="
106     %statsSrvUnique         = 11    :  $statsSrvUnique      = "SrvUnique ="
107     %statsSrvUniquePeek     = 12    :  $statsSrvUniquePeek  = "SrvUniquePeek ="
108     %statsClientRead        = 13    :  $statsClientRead     = "ClientRead ="
109     %statsClientWrite       = 14    :  $statsClientWrite    = "ClientWrite ="
110     %statsBackupRead        = 15    :  $statsBackupRead     = "BackupRead ="
111     %statsBackupWrite       = 16    :  $statsBackupWrite    = "BackupWrite ="
112     %MAXstatsFIELDS         = 16
113
114     %FolderRoot             = 1
115     %BackupRoot             = 2
116     %StatsRecv              = 3
117     %FolderObject           = 4
118     %ConfigData             = 5
119     %IP                     = 6     :   $cfgIP              = "IP ="
120     %MaxConnections         = 7     :   $cfgMaxConnections  = "MaxConnections ="
121     %HttpPort               = 8     :   $cfgHttpPort        = "HttpPort ="
122     %HttpsPort              = 9     :   $cfgHttpsPort       = "HttpsPort ="
123     %CertFile               = 10    :   $cfgCertFile        = "CertFile ="
124     %RelayHttpsHost         = 11    :   $cfgRelayHttpsHost  = "RelayHttpsHost ="
125     %Encoding               = 12    :   $cfgEncoding        = "Encoding ="
126     %RAMRecvSize            = 13    :   $cfgRAMRecvSize     = "RAMRecvSize ="
127     %DLLs                   = 14    :   $cfgDLLs            = "DLLs ="
128     %LocalBKPath            = 15    :   $cfgLocalBKPath     = "LocalBKPath ="
129     %SendBKName             = 16    :   $cfgSendBKName      = "SendBKName ="
130     %SendBKPort             = 17    :   $cfgSendBKPort      = "SendBKPort ="
131     %SendBKPWHash           = 18    :   $cfgSendBKPWHash    = "SendBKPWHash ="
132     %RecvBKPort             = 19    :   $cfgRecvBKPort      = "RecvBKPort ="
133     %RecvBKPWHash           = 20    :   $cfgRecvBKPWHash    = "RecvBKPWHash ="
134     %ServiceName            = 21    :   $cfgServiceName     = "ServiceName ="
135     %ServiceUser            = 22    :   $cfgServiceUser     = "ServiceUser ="
136     %DiskFlushTimer         = 23    :   $cfgDiskFlushTimer  = "DiskFlushTimer ="
137     %AltIPInfo              = 24    :   $cfgAltIPInfo       = "AltIPInfo ="
138     %RelayNoAltIP           = 26    :   $cfgRelayNoAltIP    = "RelayNoAltIP ="
139     %ForcedCookies          = 27    :   $cfgForcedCookies   = "ForcedCookies ="
140     %Meta                   = 28    :   $cfgMeta            = "Meta ="
141     %MAXcfgFIELDS           = 28
142
143                             :   %F00eAlias          = 1
144     %F01_FolderName         = 1     :   %F01eFolderName     = 2
145     %F02_LogLocation        = 2     :   %F02eLogLocation    = 3
146     %F03_TimeStamp          = 3     :   %F03eTimeStamp      = 4
147     %F04_Thread             = 4     :   %F04eThread         = 5
148     %F05_Category           = 5     :   %F05eCategory       = 6
149     %F06_Message            = 6     :   %F06eMessage        = 7
150     %F07_Details            = 7     :   %F07eDetails        = 8
151     %F08_Bytes              = 8     :   %F08eBytes          = 9
152     %F09_IPNum              = 9     :   %F09eIPNum          = 10
153     %F10_ThreadNum          = 10    :   %F10eThreadNum      = 11
154     %F11_ThreadTime         = 11    :   %F11eThreadTime     = 12
```

```
155     %F12_PersonUUID         = 12    :   %F12ePersonUUID     = 13
156     %F13_SessionUUID        = 13    :   %F13eSessionUUID    = 14
157     %F14_Datum1             = 14    :   %F14eDatum1         = 15
158     %F15_Datum2             = 15    :   %F15eDatum2         = 16
159     %F16_Datum3             = 16    :   %F16eDatum3         = 17
160     %F17_Datum4             = 17    :   %F17eDatum4         = 18
161     %F18_MessageCode        = 18    :   %F18eMessageCode    = 19
162     %F19_MoxVerMajor        = 19    :   %F19eMoxVerMajor    = 20
163     %F20_MoxVerMinor        = 20    :   %F20eMoxVerMinor    = 21
164     %F21_Country            = 21    :   %F21eCountry        = 22
165
166     %Tdata                  = 21    :   %Tedata             = 22
167
168 Rem 'Main
169     FUNCTION PBMAIN () AS LONG
170         GLOBAL glFinalResult, glLogStats, glLogRotate, glLogQuiet, glAllOff, glAllOn, llIsCritical  AS LONG
171         global gsFinalMessage as string
172         local  lsCommand, lsHomeFldr, lsResult, lsWork, lsHeader, lsFooter, lsTmpLog, lsaFolderPath(), lsaMeta(), lsFile, lsaFiles(), lsaFiles1(), lsaFiles2(), lsTmp, lsaTmp(), lsaSrt(), lsData, lsDa
    ta() as string
173         local  lsDest, lsaDest(), lsLastStatsSnapshot, lsNowStatsSnapshot, lsLookFor, lsPreProcEmail, lsBackupRootFolder, lsStatsRecvFolder, lsMoxViewFolder, lsMailServer, lsMailServerPort, lsMailFrom
     AS STRING
174         local  lsEmailSubject, lsEmailOnError, lsEmailNoError, lsMoxieOnError, lsOrigFldr, lsWorkFldr, lsT1, lsT2, lsForceCnwEmailErr, lsServiceData, lsServiceName AS STRING
175         local  llC1, llC2, llC21, llC22, llC3, llC4, llFlag, llFlag1, llFlag2, llLen, llBrokenFlag, llPerformZip, llFolderPathNum, llLOF as long
176         LOCAL  lnTimer AS SINGLE
177         LOCAL  tddDirData AS DIRDATA
178         LOCAL  tiPowerTime AS IPOWERTIME
179
180         REDIM                                           lsaFolderPath(0), lsaMeta(0), lsaFiles(0), lsaFiles1(0), lsaFiles2(0), lsaTmp(0), lsaData(0), lsaDest(0)
181
182         lnTimer                                 =   TIMER
183         glFinalResult                           =   0
184         StampItInit
185
186         REM 'Preload the email notification with the command line settings
187             gsFinalMessage                      +=  $CRLF + "Command-line arguments used to start the program: " + $CRLF + $TAB + command$
188
189         REM 'Check command line parameters. manually verify with:
190             '   c:\Util\PreProc>%comspec% /z /c PreProc /Logrotate /Stats /Quiet
191             '   c:\Util\PreProc>%comspec% /z /c PreProc /AllOff
192             '   c:\Util\PreProc>%comspec% /z /c PreProc /AllOn
193             '   c:\Util\PreProc>echo %errorlevel%
194
195             lsCommand                           =   lcase$(trim$(command$))
196             If isTrue( INSTR(lsCommand, "/quiet"))          THEN glLogQuiet    = %True
197             If isTrue( INSTR(lsCommand, "/stats"))          THEN glLogStats    = %True
198             If isTrue( INSTR(lsCommand, "/logrotate"))      THEN glLogRotate   = %True
199             If isTrue( INSTR(lsCommand, "/alloff"))         THEN glAllOff      = %True
200             If isTrue( INSTR(lsCommand, "/allon"))          THEN glAllOn       = %True
201
202             If isfalse( glLogStats) _
203             and isfalse(glLogRotate) _
204             and isfalse(glAllOff) _
205             and isfalse(glAllOn) _
206             Then
207                 'NOTE: bacause sCaptureConfiguration has not been called yet, no email will be sent
208                 glFinalResult                   =   1
209                 GoTo                                PrematureExit
210             End If
211
212         REM 'Find out where we are starting from
213             lsCommand                           =   "echo %cd%"
214             ShellExec                               "", "", lsCommand, lsHomeFldr
215             lsHomeFldr                          =   RTRIM$( lsHomeFldr, ANY " \") + "\"
216
217         REM 'Remove any old tmp files from last run (the Moxie program run should have completed at this point!)
218             if glLogRotate OR glLogStats Then
219                 lsCommand                       =   "DEL /Q /F /S " + $DQ + ".\Moxie\Recv\*.*" + $DQ
220                 ResultFlagOnShellExec               "ShellExec", "", "", lsCommand, lsResult, $Ignore, ""
221                 mCommitLog
222             END IF
223
224         REM 'Capture all the config info
225             sCaptureConfiguration                   lsHomeFldr, _
226                                                     lsaFolderPath(), _
227                                                     lsBackupRootFolder, _
228                                                     lsStatsRecvFolder, _
229                                                     lsMoxViewFolder, _
```

```
                                                    lsMailServer, _
                                                    lsMailServerPort, _
                                                    lsMailFrom, _
                                                    lsForceCnwEmailErr, _
                                                    lsEmailOnError, _
                                                    lsEmailNoError, _
                                                    lsMoxieOnError

    lsMoxViewFolder                             =   RTRIM$( lsMoxViewFolder, ANY " \") + "\"
    lsBackupRootFolder                          =   RTRIM$( lsBackupRootFolder, ANY " \") + "\"

    sCaptureConfigurationSave                       lsHomeFldr, _
                                                    lsaFolderPath(), _
                                                    lsResult
    mCommitLog

REM 'Parse out the Meta info, per server
    MetaCapture lsaFolderPath(), lsaMeta()

REM 'Ensure that our various programs are where they are supposed to be
    ' http://www.7-zip.org/faq.html
    ' 7-Zip stores only relative paths of files (without a drive letter prefix).
    ' To impliment, change the current folder to the folder that is common for all files that you want to compress and
    ' then you can use relative paths

    lsCommand                                   =   "7za"
    IF ShellExec7za( lsHomeFldr + $ProgramName, lsHomeFldr, lsCommand, lsTmp) = %True _
    OR INSTR( lcase$( lsTmp), lcase$( "Copyright")) = %False _
    OR DIR$(lsHomeFldr + "PwrMail.exe") <> "PwrMail.exe" _
    OR DIR$(lsHomeFldr + "Moxie\Moxie.exe") <> "Moxie.exe" _
    OR DIR$(lsHomeFldr + "Moxie\Moxie.PreProc.mox") <> "Moxie.PreProc.mox" _
    OR DIR$(lsHomeFldr + "Moxie\Moxie.PreProc.SendGrid.mox") <> "Moxie.PreProc.SendGrid.mox" _
    Then
        lsTmp                                   =   "One or more of these programs is missing or not functioning:"
        lsTmp                                   +=  $CRLF + $TAB + "7za.exe"
        lsTmp                                   +=  $CRLF + $TAB + "PwrMail.exe"
        lsTmp                                   +=  $CRLF + $TAB + "Moxie\Moxie.exe"
        lsTmp                                   +=  $CRLF + $TAB + "Moxie\Moxie.PreProc.mox"
        lsTmp                                   +=  $CRLF + $TAB + "Moxie\Moxie.PreProc.SendGrid.mox"
        lsTmp                                   +=  $CRLF + "Cannot Proceed."
        lsResult                                +=  $CRLF + "Error: " + lsTmp
        gsFinalMessage                          +=  $Crlf + lsTmp
        glFinalResult                           =   3
        mCommitLog
        GoTo                                        PrematureExit
    Else
        lsResult                                +=  $CRLF + "Our zipping program (7za.exe) was found."
        mCommitLog
    End If

REM 'If requested, handle MoxView service start/stop
    REM 'ServiceStop - lsMoxViewFolder
        if glAllOff Then
            IF LEN( DIR$(lsMoxViewFolder + "Moxie.cfg")) THEN
                sFileGet                            lsMoxViewFolder + "Moxie\Moxie.cfg", lsServiceData, llLen
                IF LEN( lsServiceData) THEN
                    lsServiceName               =   TRIM$( EXTRACT$( REMAIN$( lsServiceData,$cfgServiceName     ), $CRLF))
                    lsCommand                   =   "NET STOP " + $DQ + lsServiceName + $DQ
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotIncludes, "successfully")
                    mCommitLog
                End If
            END IF
        End If

    REM 'ServiceStart - lsMoxViewFolder
        if glAllOn Then
            IF LEN( DIR$(lsMoxViewFolder + "Moxie.cfg")) THEN
                sFileGet                            lsMoxViewFolder + "Moxie.cfg", lsServiceData, llLen
                IF LEN( lsServiceData) THEN
                    lsServiceName               =   TRIM$( EXTRACT$( REMAIN$( lsServiceData,$cfgServiceName     ), $CRLF))
                    lsCommand                   =   "NET START " + $DQ + lsServiceName + $DQ
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotIncludes, "successfully")
                    mCommitLog
                End If
            END IF
        End If

```

```
307         REM 'Perform Moxie.PreProc.SendGrid.mox to PreProcess rejected emails
308             IF LEN( DIR$(lsHomeFldr + "Moxie\Moxie.PreProc.SendGrid.mox")) THEN
309                 If glLogRotate THEN
310                     lsCommand                       =   "CD /D " + $DQ + lsHomeFldr + "Moxie" + $DQ + " && Moxie.exe " + $DQ + "Moxie.PreProc.SendGrid.mox"
311                     lsCommand                       +=  $DQ
312                     llBrokenFlag                    =   ResultFlagOnShellExec( "ShellExec", lsWork, lsHomeFldr, lsCommand, lsResult, $Ignore, "")
313                     mCommitLog
314                 END IF
315             END IF
316
317         FOR llFolderPathNum = 1 TO UBOUND( lsaFolderPath(2))
318
319             lsWork                                  =   lsaFolderPath( %FolderRoot, llFolderPathNum) + "\" + lsaFolderPath( %FolderObject, llFolderPathNum) + "\"
320
321             IF  ISTRUE(LEN( lsaFolderPath(%BackupRoot,      llFolderPathNum))) _
322             AND ISTRUE(LEN( lsaFolderPath(%FolderRoot,      llFolderPathNum))) _
323             AND ISTRUE(LEN( lsaFolderPath(%StatsRecv,       llFolderPathNum))) _
324             AND ISTRUE(LEN( lsaFolderPath(%FolderObject,    llFolderPathNum))) _
325             AND ISTRUE(LEN( lsaFolderPath(%ServiceName,     llFolderPathNum))) _
326             AND ISTRUE(LEN( lsaFolderPath(%ServiceUser,     llFolderPathNum))) _
327             THEN
328                 REM 'Folder Init
329                     lsHeader                        =   "<" & $ProgramName & ".Folder " + lsaFolderPath(%FolderObject, llFolderPathNum) + ">"
330                     lsFooter                        =   "</" & $ProgramName & ".Folder>"
331                     lsResult                        +=  $CRLF + lsHeader
332
333                 REM 'Stats Init
334                     If glLogStats then
335                         sFileGet                        mABS_SRC + "\Moxie.stats.Last", lsData, llLen
336                         lsLookFor                   =   "[StatsLast] = "
337                         lsNowStatsSnapshot          =   StampItNowLogFmt
338                         If INSTR( lsData, lsLookFor) THEN
339                             lsLastStatsSnapshot     =   remain$( lsData, lsLookFor)
340                             lsLastStatsSnapshot     =   TRIM$( Extract$( lsLastStatsSnapshot, $CRLF), ANY $WHITESPACE)
341                         ELSE
342                             'Assume a default date of just after midnight; in otherwords, "Today"
343                             lsTmp                   =   DATE$
344                             lsLastStatsSnapshot     =   RIGHT$(lsTmp, 4) + "-" + LEFT$(lsTmp, 5) + " 00:00:00"
345                         END IF
346
347                         REM 'new data for this field
348                             lsNowStatsSnapshot      =   StampItNowLogFmt
349                             Redim                       lsaData( 1 to 3)
350                             lsaData(1)              =   "#Do not edit this file as it will be overwritten"
351                             lsaData(2)              =   ""
352                             lsaData(3)              =   lsLookFor + lsNowStatsSnapshot
353                             sFilePut                    mABS_SRC + "\Moxie.stats.Last", Join$( lsaData(), $CrLf)
354
355                         REM 'Create our stats time stamp file for this run
356                             lsData                  =   "StatsLast" + $TAB + "StatsNow" + $CRLF
357                             lsData                  +=  lsLastStatsSnapshot + $Tab + lsNowStatsSnapshot
358                             sFilePut                    ".\Moxie\Recv\" + format$(llFolderPathNum, "000") + ".snapshot.tab", lsData
359                     End If
360
361                 REM 'ServiceInit - Ensure Folders
362                     if glLogRotate Then
363                         sEnsureFolder                   "" , mDEST_BIN,     lsTmpLog   : lsResult += $CRLF + "sEnsureFolder: " + mDEST_BIN + lsTmpLog
364                         sEnsureFolder                   "" , mDEST_LOG,     lsTmpLog   : lsResult += $CRLF + "sEnsureFolder: " + mDEST_LOG + lsTmpLog
365                         sEnsureFolder                   "" , mDEST_DBSTORE, lsTmpLog   : lsResult += $CRLF + "sEnsureFolder: " + mDEST_DBSTORE + lsTmpLog
366                         sEnsureFolder                   "" , mDEST_REPORTS, lsTmpLog   : lsResult += $CRLF + "sEnsureFolder: " + mDEST_REPORTS + lsTmpLog
367                         sEnsureFolder                   "" , mDEST_WORK,    lsTmpLog   : lsResult += $CRLF + "sEnsureFolder: " + mDEST_WORK + lsTmpLog
368                         mCommitLog
369                     End If
370
371                 REM 'AuxFoldersToZip - Every iteration
372                     if glLogRotate Then
373                         '\Private\
374                         lsCommand                   =   "IF EXIST " + $DQ + mDEST_3 + $DQ + " (DEL " + $DQ + mDEST_3 + $DQ+ ")"
375                         llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
376                         lsCommand                   =   "IF EXIST " + $DQ + mDEST_3 + ".*" + $DQ + " (DEL " + $DQ + mDEST_3 + ".*" + $DQ+ ")"
377                         llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
378                         lsCommand                   =   "7za a -tzip -r -ssw"
379                         lsCommand                   +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
380                         lsCommand                   +=  " " + $DQ + mDEST_3 + $DQ
381                         lsCommand                   +=  " " + $DQ + mSRC__3 + $DQ
382                         llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
383                         mCommitLog
```

```
                    Sleep 500

                    '\Public\
                    lsCommand                   =   "IF EXIST " + $DQ + mDEST_4 + $DQ + " (DEL " + $DQ + mDEST_4 + $DQ+ ")"
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
                    lsCommand                   =   "IF EXIST " + $DQ + mDEST_4 + ".*" + $DQ + " (DEL " + $DQ + mDEST_4 + ".*" + $DQ+ ")"
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
                    lsCommand                   =   "7za a -tzip -v600m -r -ssw"
                    lsCommand                   +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
                    lsCommand                   +=  " " + $DQ + mDEST_4 + $DQ
                    lsCommand                   +=  " " + $DQ + mSRC__4 + $DQ
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
                    mCommitLog
                    Sleep 500

                    '\Templates\
                    lsCommand                   =   "IF EXIST " + $DQ + mDEST_5 + $DQ + " (DEL " + $DQ + mDEST_5 + $DQ+ ")"
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
                    lsCommand                   =   "IF EXIST " + $DQ + mDEST_5 + ".*" + $DQ + " (DEL " + $DQ + mDEST_5 + ".*" + $DQ+ ")"
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
                    lsCommand                   =   "7za a -tzip -r -ssw"
                    lsCommand                   +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
                    lsCommand                   +=  " " + $DQ + mDEST_5 + $DQ
                    lsCommand                   +=  " " + $DQ + mSRC__5 + $DQ
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
                    mCommitLog
                    Sleep 500

                End If

            REM 'ServiceStop
                if glLogRotate OR glAllOff Then
                    lsCommand                   =   "NET STOP " + $DQ + lsaFolderPath(%ServiceName, llFolderPathNum) + $DQ
                    llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotIncludes, "successfully")
                    mCommitLog
                End If

            REM 'Perform Zip of Moxie Service and data folders
                if glLogRotate Then
                    llPerformZip                =   %False
                    llBrokenFlag                =   %False

                    REM 'Perform Daily on SSL enabled servers
                        IF VAL(lsaFolderPath(%HttpsPort, llFolderPathNum)) = 443  THEN
                            llPerformZip        =   %True
                        END IF

                    REM 'Perform Saturday on ALL servers
                        IF LCASE$(GetDow(StampItDate)) = LCASE$("Sat") THEN
                            llPerformZip        =   %True
                        END IF

                    IF llPerformZip THEN
                        REM 'Gather and zip all "Moxie" files
                            sGetFileNamesOnly       lsaFiles(), lsWork, %dir_Lib OR %dir_History
                            llC2                =   0
                            For llC1 = 1 to UBOUND(lsaFiles)
                                llFlag          =   %False
                                If lcase$( Right$( lsaFiles(llC1),  4)) = "lib\"    Then Iterate for
                                If lcase$( Right$( lsaFiles(llC1),  8)) = "history\" Then Iterate for
                                If lcase$( Right$( lsaFiles(llC1),  4)) = ".dll"    Then llFlag = %True
                                If lcase$( Right$( lsaFiles(llC1),  4)) = ".dat"    Then llFlag = %True
                                If lcase$( Right$( lsaFiles(llC1),  4)) = ".cfg"    Then llFlag = %True
                                If lcase$( Right$( lsaFiles(llC1),  4)) = ".exe"    Then llFlag = %True
                                If lcase$( Right$( lsaFiles(llC1),  4)) = ".mox"    Then llFlag = %True
                                If lcase$( Right$( lsaFiles(llC1),  4)) = ".bat"    Then llFlag = %True
                                IF llFlag = %False THEN iterate for
                                Incr                llC2
                                lsaFiles(llC2)      =   lsaFiles(llC1)
                            Next
                            REDIM PRESERVE lsaFiles(1 TO llC2)
                            If llC2 THEN
                                sFilePut                lsWork + $PreProcListFile, Join$( lsaFiles(), $Crlf)
                                lsCommand           =   "7za a -tzip -ssw"
                                lsCommand           +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
                                lsCommand           +=  " -i@" + $DQ + lsWork + $PreProcListFile + $DQ
                                lsCommand           +=  " " + $DQ + mDEST_BIN + "\" + StampItDateHM + $DQ
```

```
                                    llBrokenFlag        =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
                                    KILL                    lsWork + $PreProcListFile
                                    If llBrokenFlag Then
                                        glFinalResult   =   6
                                        gsFinalMessage  +=  $CRLF + "The following command failed. Proceeding..."
                                        gsFinalMessage  +=  $CRLF + lsCommand
                                    End If
                                ELSE
                                    lsResult            +=  $CRLF + "Error: sGetFileNamesOnly returned 0 files. 'Moxie' files were not zipped"
                                End If

                                mCommitLog
                            END IF
                        END IF

                    REM 'Gather and zip all "DBStore" files and sub folders
                        if glLogRotate Then
                            lsCommand                   =   "7za a -tzip -r -ssw"
                            lsCommand                   +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
                            lsCommand                   +=  " " + $DQ + mDEST_2 + $DQ
                            lsCommand                   +=  " " + $DQ + mSRC__2 + $DQ
                            llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
                            If llBrokenFlag Then
                                glFinalResult           =   7
                                gsFinalMessage          +=  $CRLF + "The following command failed. Proceeding..."
                                gsFinalMessage          +=  $CRLF + lsCommand
                            End If

                            mCommitLog
                        End If

                    REM 'Gather and zip all "Work" files and sub folders but only if the "Work" folder exists
                        if glLogRotate Then
                            if LEN( DIR$(lsWork + "Work\")) THEN
                                'lsCommand                =   "7za a -tzip -v600m -r -ssw"
                                lsCommand               =   "7za a -tzip -v600m -r -ssw -x!*.pdf"
                                lsCommand               +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
                                lsCommand               +=  " " + $DQ + mDEST_6 + $DQ
                                lsCommand               +=  " " + $DQ + mSRC__6 + $DQ
                                llBrokenFlag            =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
                                If llBrokenFlag Then
                                    glFinalResult       =   7
                                    gsFinalMessage      +=  $CRLF + "The following command failed. Proceeding..."
                                    gsFinalMessage      +=  $CRLF + lsCommand
                                End If
                            End If

                            mCommitLog
                        End If

                    REM 'Gather and PreProcess "log" files that fall between lsLastStatsSnapshot & StampItNowLogFmt for later processing for later processing
                        If glLogStats then
                            sGetFileNamesOnly               lsaFiles(), lsWork, %dir_History
                            llC2                    =   0
                            For llC1 = 1 to UBOUND(lsaFiles)
                                If lcase$( Right$( lsaFiles(llC1), 4)) <> ".log" Then Iterate for
                                Incr                        llC2
                                lsaFiles(llC2)          =   lsWork + lsaFiles(llC1)
                            Next
                            REDIM                           PRESERVE lsaFiles(1 TO llC2)
                            If llC2 THEN
                                GatherLogFilesAndPreProcess     lsaFolderPath(), llFolderPathNum, lsaFiles(), lsResult
                            ELSE
                                lsResult                +=  $CrLf + "Error: No log files seen for " + lsWork
                            End If

                            mCommitLog
                        End If

                    REM 'Gather and PreProcess "stats" files that fall between lsLastStatsSnapshot & StampItNowLogFmt for later processing for later processing
                        If glLogStats then
                            sGetFileNamesOnly               lsaFiles(), lsWork, %dir_History
                            llC2                    =   0
                            For llC1 = 1 to UBOUND(lsaFiles)
                                If                          lcase$( Right$( lsaFiles(llC1), 6)) <> ".stats" Then Iterate for
                                Incr                        llC2
                                lsaFiles(llC2)          =   lsWork + lsaFiles(llC1)
```

```
538                         Next
539                         REDIM                               PRESERVE lsaFiles(1 TO llC2)
540                         If llC2 THEN
541                             Redim                           lsaTmp( 1 to llC2), lsaSrt( 1 to llC2)
542                             Redim                           lsaData( 1 to %MAXstatsFIELDS, 1 to llC2)
543
544                             For llC1 = 1 to UBOUND(lsaFiles)
545
546                                 sFileGet                    lsaFiles(llC1), lsTmp, llLOF
547
548                                 lsaData( %statsStatTime, llC1)      =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsStatTime         ), $CRLF))
549                                 lsaData( %statsMoxieVer, llC1)      =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsMoxieVer         ), $CRLF))
550                                 lsaData( %statsHttpState, llC1)     =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsHttpState        ), $CRLF))
551                                 lsaData( %statsHttpsState, llC1)    =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsHttpsState       ), $CRLF))
552                                 lsaData( %statsBkRecvState, llC1)   =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsBkRecvState      ), $CRLF))
553                                 lsaData( %statsSrvHits, llC1)       =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsSrvHits          ), $CRLF))
554                                 lsaData( %statsSrvCurrent, llC1)    =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsSrvCurrent       ), $CRLF))
555                                 lsaData( %statsSrvPeek, llC1)       =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsSrvPeek          ), $CRLF))
556                                 lsaData( %statsSrvCompleted, llC1)  =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsSrvCompleted     ), $CRLF))
557                                 lsaData( %statsSrvUnique, llC1)     =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsSrvUnique        ), $CRLF))
558                                 lsaData( %statsSrvUniquePeek, llC1) =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsSrvUniquePeek    ), $CRLF))
559                                 lsaData( %statsClientRead, llC1)    =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsClientRead       ), $CRLF))
560                                 lsaData( %statsClientWrite, llC1)   =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsClientWrite      ), $CRLF))
561                                 lsaData( %statsBackupRead, llC1)    =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsBackupRead       ), $CRLF))
562                                 lsaData( %statsBackupWrite, llC1)   =   TRIM$( EXTRACT$( REMAIN$( lsTmp, $statsBackupWrite      ), $CRLF))
563
564                                 For llC3 = 1 to %MAXstatsFIELDS
565                                     lsaTmp( llC1)       +=  $Tab + lsaData( llC3, llC1)
566                                 Next
567                                 lsaTmp( llC1)           =   TRIM$( lsaTmp( llC1), $Tab)
568
569                                 lsaSrt( llC1)           =   lsaData( %statsStatTime, llC1)  + $Tab + _
570                                                             format$(llFolderPathNum, "000") + _
571                                                             Format$(llC2, "0000000")
572
573                             Next
574
575                             ARRAY SORT lsaSrt(), COLLATE UCASE, TAGARRAY lsaTmp()
576                             For llC2 = 1 to llC1
577                                 lsaTmp( llC2)           =   format$(llFolderPathNum, "000") + Format$(llC2, "0000000") + $Tab + lsaTmp( llC2)
578                             Next
579
580                             lsData                      =   JOIN$( lsaTmp(), $CRLF)
581                             REPLACE                         $CRLF WITH chr$(0) IN lsData
582                             REPLACE                         $Tab WITH chr$(1) IN lsData
583                             sFilePut                        ".\Moxie\Recv\" + format$(llFolderPathNum, "000") + ".stats.cnd", lsData
584                             Reset                           lsData
585
586                             lsTmp                       =  "Alias" + chr$(1) + "Alias"              + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
587                             lsTmp                       += "F01"   + chr$(1) + "F01StatTime"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
588                             lsTmp                       += "F02"   + chr$(1) + "F02MoxieVer"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
589                             lsTmp                       += "F03"   + chr$(1) + "F03HttpState"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
590                             lsTmp                       += "F04"   + chr$(1) + "F04HttpsState"      + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
591                             lsTmp                       += "F05"   + chr$(1) + "F05BkRecvState"     + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
592                             lsTmp                       += "F06"   + chr$(1) + "F06SrvHits"         + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
593                             lsTmp                       += "F07"   + chr$(1) + "F07SrvCurrent"      + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
594                             lsTmp                       += "F08"   + chr$(1) + "F08SrvPeek"         + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
595                             lsTmp                       += "F09"   + chr$(1) + "F09SrvCompleted"    + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
596                             lsTmp                       += "F10"   + chr$(1) + "F10SrvUnique"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
597                             lsTmp                       += "F11"   + chr$(1) + "F11SrvUniquePeek"   + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
598                             lsTmp                       += "F12"   + chr$(1) + "F12ClientRead"      + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
599                             lsTmp                       += "F13"   + chr$(1) + "F13ClientWrite"     + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
600                             lsTmp                       += "F14"   + chr$(1) + "F14BackupRead"      + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
601                             lsTmp                       += "F15"   + chr$(1) + "F15BackupWrite"     + chr$(1) + "" + chr$(1) + "[R]" + chr$(1)
602
603                             sFilePut                        ".\Moxie\Recv\stats.cnt", TRIM$( lsTmp, chr$(0))
604                             Reset                           lsTmp
605
606                         ELSE
607                             lsResult                    +=   $CrLf + "Error: No stats files seen for " + lsWork
608                         End If
609
610                         mCommitLog
611                     End If
612
613                 REM 'Gather and PreProcess "Exception" file dates and flag them that fall between lsLastStatsSnapshot & StampItNowLogFmt for later processing
614                     ' Filename format
```

```
615                     '  Exception-2014-05-05-011925-Default.dll.dat
616                     '  Exception-2014-05-05-011925-Moxie.dll.dat
617                     '  Exception-2014-05-05-011925-Moxie.exe.dat
618                     If glLogStats then
619                         sGetFileNamesOnly               lsaFiles(), lsWork, %dir_History
620                         llC2                        =   0
621                         For llC1 = 1 to UBOUND(lsaFiles)
622                             If lcase$( Right$( lsaFiles(llC1), 4)) <> ".dat" Then Iterate for
623                             If lcase$( LEFT$( lsaFiles(llC1), 9)) <> "exception" Then Iterate for
624                             Incr                        llC2
625                             lsaFiles(llC2)              =   lsWork + lsaFiles(llC1)
626                         Next
627                         REDIM                           PRESERVE lsaFiles(1 TO llC2)
628                         If llC2 THEN
629                             REM 'Parse and convert filename time stamp to find out when the exception occured
630                             For llC3 = 1 to llC2
631                                 lsT2                =   LEFT$( lsaFiles(llC3), (INSTR( -1, lsaFiles(llC3), "-") - 1))
632                                 lsT2                =   remain$( lsT2, "-")
633                                 '(convert 2014-06-09-150400 -> 2014-06-09 15:04:00)
634                                 lsT1                =   left$( lsT2, 10)
635                                 lsT1                +=  " " + MID$( lsT2, 12, 2)
636                                 lsT1                +=  ":" + MID$( lsT2, 14, 2)
637                                 lsT1                +=  ":" + MID$( lsT2, 16, 2)
638
639                                 REM 'If the fimestamp is in the past, assume that it has been reported already
640                                     'If  (DateTimeToQuad(lsT1) <= DateTimeToQuad( lsNowStatsSnapshot)) _
641                                     'AND (DateTimeToQuad(lsT1) => DateTimeToQuad( lsLastStatsSnapshot)) _
642                                     'THEN
643                                     If  (DateTimeToQuad(lsT1) => DateTimeToQuad( lsLastStatsSnapshot)) THEN
644                                         lsResult        +=  $Crlf + "Error: Exceptions found " + lsaFiles(llC3)
645                                         glFinalResult   =   8
646                                         gsFinalMessage  +=  $Crlf + "Error: Exceptions found " + lsaFiles(llC3)
647                                     End If
648                             Next
649                         ELSE
650                             ' Fall through as this is a good thing
651                         End If
652
653                         mCommitLog
654                     End If
655
656                 REM 'Move old reports to the backup area
657                     if glLogRotate Then
658                         IF LCASE$(GetDow(StampItDate)) = LCASE$("Sat") THEN
659                             if LEN( DIR$(lsWork + "Private\Report Results\")) THEN
660                                 sGetFileNamesOnly           lsaFiles(), lsWork + "Private\Report Results\", 0
661                                 llC2                    =   0
662                                 REDIM                       lsaDest( 1 to UBOUND(lsaFiles))
663                                 For llC1 = 1 to UBOUND(lsaFiles)
664                                     lsTmp               =   DIR$(lsWork + "Private\Report Results\" + lsaFiles(llC1) TO tddDirData)
665                                     tiPowerTime         =   CLASS "PowerTime"
666                                     tiPowerTime.FileTime =  tddDirData.CreationTime   ' tiPowerTime contains the file creation time in a localized format.
667                                     lsT1                =   tiPowerTime.DateString
668                                     tiPowerTime.AddMonths       2
669                                     lsT2                =   tiPowerTime.DateString
670                                     If lsT2 > StampItDate Then Iterate for
671                                     Incr                    llC2
672                                     lsaDest(llC2)       =   $DQ + mDEST_REPORTS + "\" + $DQ
673                                     lsaFiles(llC2)      =   $DQ +  lsaFiles(llC1) + $DQ
674                                 Next
675                                 REDIM                       PRESERVE lsaFiles(1 TO llC2)
676                                 REDIM                       PRESERVE lsaDest(1 TO llC2)
677                                 If llC2 THEN
678                                     REM 'Prep the 'local' folder
679                                         lsOrigFldr      =   RTRIM$( CURDIR$, ANY " \") + "\"
680                                         lsWorkFldr      =   lsWork + "Private\Report Results\"
681
682                                         If lsOrigFldr <> lsWorkFldr THEN
683                                             CHDRIVE         Extract$(lsWorkFldr, ":")
684                                             chdir           REMAIN$(lsWorkFldr, ":")
685                                         End If
686
687                                     For llC1 = 1 to llC2
688                                         lsCommand       =   "MOVE /Y " + lsaFiles(llC1) + " " + lsaDest(llC1)
689                                         llBrokenFlag    =   ResultFlagOnShellExec( "ShellExec", lsWork, "", lsCommand, lsResult, $FlagIfNotIncludes, "1 file(s) moved")
690                                     NEXT
691
```

```
692                                 REM ' Restore the 'local' folder
693                                     If lsOrigFldr <> lsWorkFldr THEN
694                                         CHDRIVE         Extract$(lsOrigFldr, ":")
695                                         chdir           REMAIN$(lsOrigFldr, ":")
696                                     End If
697
698                                 End If
699                             End If
700                         end if
701                     End If
702
703                 REM 'Zip/Move/Save logs: If success, then delete them from the lsaFolderPath()
704                     if glLogRotate Then
705                         sGetFileNamesOnly               lsaFiles(), lsWork, %dir_History OR %dir_DBStore
706                         'Require two versions of lsaFiles(),
707                         ' 1st lsaFiles1() is what gets zipped/saved
708                         ' 2nd lsaFiles2() is what gets deleted
709
710                         llC21                           =   0
711                         llC22                           =   0
712
713                         REDIM lsaFiles1(1 TO UBOUND(lsaFiles))
714                         REDIM lsaFiles2(1 TO UBOUND(lsaFiles))
715
716                         For llC1 = 1 to UBOUND(lsaFiles)
717                             llFlag1                     =   %False
718                             llFlag2                     =   %False
719                             If lcase$( Right$( lsaFiles(llC1), 4))  = ".log" Then
720                                 llFlag1                 =   %True
721                                 llFlag2                 =   %True
722                             End If
723                             If lcase$( Right$( lsaFiles(llC1), 4))  = ".dat" Then
724                                 llFlag1                 =   %True
725                                 llFlag2                 =   %True
726                             End If
727                             If lcase$( Right$( lsaFiles(llC1), 6))  = ".stats" Then
728                                 llFlag1                 =   %True
729                                 llFlag2                 =   %True
730                             End If
731                             If ((lcase$( Right$( lsaFiles(llC1), 11)) = "session.cnd") AND (lcase$( Right$( lsaFiles(llC1), 15)) <> "biz.session.cnd")) Then
732                                 llFlag1                 =   %True
733                                 If MetaSearch( lsaMeta(), llFolderPathNum, "Session") <> "NoReset" Then
734                                     llFlag2             =   %True
735                                 End If
736                             End If
737                             If lcase$( Right$( lsaFiles(llC1), 10)) = ".TwoFactor" Then
738                                 llFlag1                 =   %True
739                                 llFlag2                 =   %True
740                             End If
741                             lsTmp                           =   lsWork + lsaFiles(llC1)
742                             IF llFlag1 = %True THEN
743                                 Incr                        llC21
744                                 lsaFiles1(llC21)          =   lsTmp
745                             End If
746                             IF llFlag2 = %True THEN
747                                 Incr                        llC22
748                                 lsaFiles2(llC22)          =   lsTmp
749                             End If
750                         Next
751                         lsResult                    +=  $CrLf + "counts " + format$(UBOUND(lsaFiles)) + " " + format$(llC21) + " " + format$(llc22)
752
753                         REDIM PRESERVE lsaFiles1(1 TO llC21)
754                         REDIM PRESERVE lsaFiles2(1 TO llC22)
755
756                         If llC21 THEN
757                             sFilePut                        lsWork + $PreProcListFile, Join$( lsaFiles1(), $Crlf)
758                             lsResult                    +=  $CrLf + "<PreProcListFile Archiving>" + Join$( lsaFiles1(), ", ") + "</PreProcListFile>"
759                             lsCommand                   =   "7za a -tzip -ssw"
760                             lsCommand                   +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
761                             lsCommand                   +=  " -i@" + $DQ + lsWork + $PreProcListFile + $DQ
762                             lsCommand                   +=  " " + $DQ + mDEST_LOG + "\" + StampItDateHM + ".zip"+ $DQ
763                             llBrokenFlag                =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
764                             KILL                            lsWork + $PreProcListFile
765
766                             REM 'Test to ensure "Everything is Ok" prior to deleting original (unZipped) file(s)
767                                 If llBrokenFlag = %False Then
768                                     'The zipping occurrs into a tmp file which is then copied into the final file
```

```
                                    'The test occurs on the final file
                                    'Therefore, we need a 1 sec timer to ensure that the final file is completely written
                                    '
                                    SLEEP 1000
                                    lsCommand           =   "7za t -tzip"
                                    lsCommand           +=  " " + $DQ + mDEST_LOG + "\" + StampItDateHM + ".zip"+ $DQ
                                    llBrokenFlag        =   ResultFlagOnShellExec( "ShellExec7za", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotIncludes, "Everything is Ok")
                                    IF llBrokenFlag = 0 THEN
                                        'Only if everything is good can we delete the orig files
                                        'lsResult        +=  $CrLf + "<PreProcListFile Deleting>" + Join$( lsaFiles(), ", ") + "</PreProcListFile>"
                                        'sFilePut            lsWork + $PreProcListFile, Join$( lsaFiles(), $Crlf)
                                        'lsCommand       =   "FOR /F %I IN (" + lsWork + $PreProcListFile + ") DO DEL " + $DQ + "%I" + $DQ
                                        'llBrokenFlag    =   ResultFlagOnShellExec( "ShellExec", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotEmpty, "")
                                        'KILL                lsWork + $PreProcListFile

                                        lsResult        +=  $CrLf + "<PreProcListFile Deleting>" + Join$( lsaFiles2(), ", ") + "</PreProcListFile>"
                                        sFilePut            lsWork + $PreProcListFile, Join$( lsaFiles2(), $Crlf)
                                        lsCommand       =   "FOR /F %I IN (" + lsWork + $PreProcListFile + ") DO DEL " + $DQ + "%I" + $DQ
                                        llBrokenFlag    =   ResultFlagOnShellExec( "ShellExec", lsWork, lsHomeFldr, lsCommand, lsResult, $FlagIfNotEmpty, "")
                                        KILL                lsWork + $PreProcListFile

                                    ELSE
                                        'We have a broken zip file
                                        llBrokenFlag    =   %True
                                    END IF
                                ELSE
                                    'We have a broken zip file
                                    llBrokenFlag        =   %True
                                END IF

                                IF llBrokenFlag = %True THEN
                                    'If we have a broken zip file
                                    lsResult            +=  $CRLF + "Error: 7za Failure: log file did not compress properly. Original log file intact."
                                END IF
                            ELSE
                                lsResult                +=  $CRLF + "Error: sGetFileNamesOnly returned 0 files. 'Zip/Move/Save logs' files were not zipped"
                            End If

                            mCommitLog
                        End If

                    REM 'Remove old TwoFactor file
                        if glLogRotate Then
                            KILL                            lsWork + "Moxie.TwoFactor"
                        End If

                    REM 'Perform RunNightly.mox if found
                        if glLogRotate Then
                            IF LEN( DIR$(lsWork + "RunNightly.mox")) THEN
                                REM 'Perform Daily on SSL enabled servers or not
                                IF VAL(lsaFolderPath(%HttpsPort, llFolderPathNum)) = 443  THEN
                                    lsCommand           =   "CD /D " + $DQ + lsWork + $DQ + " && Moxie.exe " + $DQ + "RunNightly.mox?TwoFactorAudit=y" + $DQ
                                ELSE
                                    lsCommand           =   "CD /D " + $DQ + lsWork + $DQ + " && Moxie.exe " + $DQ + "RunNightly.mox" + $DQ
                                END IF

                                llBrokenFlag            =   ResultFlagOnShellExec( "ShellExec", lsWork, lsHomeFldr, lsCommand, lsResult, $Ignore, "")

                                mCommitLog
                            END IF
                        End If

                    REM 'Gather and PreProcess all "TwoFactor" files for later processing
                        if glLogRotate Then
                            IF LEN( DIR$(lsWork + "Moxie.TwoFactor")) THEN
                                sFileGet                    lsWork + "Moxie.TwoFactor", lsData, llLof
                                lsData                  =   REMAIN$( lsData, $Crlf)

                                llC1                    =   parsecount( lsData, $CRLF)
                                Redim                       lsaTmp( 1 to llC1)
                                Parse                       lsData, lsaTmp(), $CRLF
                                Reset                       lsData

                                For llC2 = 1 to llC1        :   ' insert an Alias field
                                    lsaTmp( llC2)           =   format$(llFolderPathNum, "000") + Format$(llC2, "0000000") + $Tab + lsaTmp( llC2)
                                NEXT
```

```
                    lsData                      =   Join$( lsaTmp(), $CrLf)

                    REPLACE                         $CRLF WITH chr$(0) IN lsData
                    REPLACE                         $Tab WITH chr$(1) IN lsData
                    sFilePut                        ".\Moxie\Recv\" + format$(llFolderPathNum, "000") + ".twofactor.cnd", lsData
                    Reset                           lsData


                    lsTmp                       =   "Alias" + chr$(1) + "Alias"         + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
                    lsTmp                       +=  "F01"   + chr$(1) + "F01Site"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
                    lsTmp                       +=  "F02"   + chr$(1) + "F02Type"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
                    lsTmp                       +=  "F03"   + chr$(1) + "F03ID"         + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
                    lsTmp                       +=  "F04"   + chr$(1) + "F04Handle"     + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
                    lsTmp                       +=  "F05"   + chr$(1) + "F05Name"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
                    lsTmp                       +=  "F06"   + chr$(1) + "F06TwoFactor"  + chr$(1) + "" + chr$(1) + "[R]" + chr$(1)

                    sFilePut                        ".\Moxie\Recv\twofactor.cnt", TRIM$( lsTmp, chr$(0))
                    Reset                           lsTmp

                    lsResult                    +=  $CrLf + "Moxie.TwoFactor PreProcessed"
                END IF
            End If

        REM 'Remove DBStore files that are marked for deletion (DBStore\*.del)
            if glLogRotate Then
                lsCommand                       =   "IF EXIST " + $DQ + lsWork + "DBStore\*.del"+ $DQ + " (DEL " + $DQ + lsWork + "DBStore\*.del"+ $DQ + ")"
                llBrokenFlag                    =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotEmpty, "")
                mCommitLog
            End If

        REM 'Force Email-if-serious-error address
            if glLogRotate Then
                lsFile                          =   lsWork + "DBStore\Setup.cnd"
                sFileGet                            lsFile, lsData, llLOF

                llC1                            =   parsecount( lsData, $SEP_NUL)
                Redim                               lsaTmp( 1 to llC1)
                Redim                               lsaData( 1 to llC1)
                Parse                               lsData, lsaTmp(), $SEP_NUL

                Rem 'Remove old Setup values
                    llC2                        =   0
                    For llC3 = 1 to llC1
                        SELECT CASE AS CONST$ Lcase$( Extract$( lsaTmp(llC3), $SEP_SOH))
                            Case "cnw.emailerrfrom", "cnw.emailerrto", "cnw.emailipadminfrom", "cnw.emailipadminto"
                                'punt these values in favour of below values
                            Case else
                                Incr llC2
                                lsaData(llC2)   =   lsaTmp(llC3)
                        End Select
                    Next
                    REDIM PRESERVE                  lsaData( 1 TO llC2)
                    lsData                      =   join$( lsaData(), $SEP_NUL)

                Rem 'Append New Setup Values or default values if no Meta flag given
                    lsTmp                       =   MetaSearch( lsaMeta(), llFolderPathNum, "Cnw.EmailErrFrom")
                    If Len( lsTmp) Then
                        lsData                  +=  $SEP_NUL + "Cnw.EmailErrFrom"       + $SEP_SOH + lsTmp
                    Else
                        lsData                  +=  $SEP_NUL + "Cnw.EmailErrFrom"       + $SEP_SOH + lsForceCnwEmailErr
                    End If

                    lsTmp                       =   MetaSearch( lsaMeta(), llFolderPathNum, "Cnw.EmailErrTo")
                    If Len( lsTmp) Then
                        lsData                  +=  $SEP_NUL + "Cnw.EmailErrTo"         + $SEP_SOH + lsTmp
                    Else
                        lsData                  +=  $SEP_NUL + "Cnw.EmailErrTo"         + $SEP_SOH + lsForceCnwEmailErr
                    End If

                    lsTmp                       =   MetaSearch( lsaMeta(), llFolderPathNum, "Cnw.EmailIPAdminFrom")
                    If Len( lsTmp) Then
                        lsData                  +=  $SEP_NUL + "Cnw.EmailIPAdminFrom"   + $SEP_SOH + lsTmp
                    Else
                        lsData                  +=  $SEP_NUL + "Cnw.EmailIPAdminFrom"   + $SEP_SOH + lsForceCnwEmailErr
                    End If

                    lsTmp                       =   MetaSearch( lsaMeta(), llFolderPathNum, "Cnw.EmailIPAdminTo")
```

```
                            If Len( lsTmp) Then
                                lsData                  +=  $SEP_NUL + "Cnw.EmailIPAdminTo"     + $SEP_SOH + lsTmp
                            Else
                                lsData                  +=  $SEP_NUL + "Cnw.EmailIPAdminTo"     + $SEP_SOH + lsForceCnwEmailErr
                            End If

                        Rem 'Maintain Moxie Sort
                            llC1                        =   parsecount( lsData, $SEP_NUL)
                            Redim                           lsaData( 1 to llC1)
                            Parse                           lsData, lsaData(), $SEP_NUL
                            ARRAY SORT                      lsaData(), COLLATE UCASE
                            lsData                      =   join$( lsaData(), $SEP_NUL)

                        sFilePut                            lsFile, lsData

                        lsResult                        +=  $CrLf + "EmailForced: " + lsForceCnwEmailErr + " " + lsFile
                        mCommitLog
                    End If

                REM 'ServiceStart
                    if glLogRotate OR glAllOn Then
                        lsCommand                       =   "NET START " + $DQ + lsaFolderPath(%ServiceName, llFolderPathNum) + $DQ
                        llBrokenFlag                    =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $FlagIfNotIncludes, "successfully")
                        mCommitLog
                    End If

                    lsResult += $CRLF + lsFooter
                    mCommitLog

            ELSE
                IF LEN( lsaFolderPath(%FolderRoot, llFolderPathNum)) THEN
                    lsResult                            +=  $CRLF + "ShellExecResult: " + lsWork + " might not be configured properly, check the ServiceName and UserName"
                    glFinalResult                       =   4
                    gsFinalMessage                      +=  $CRLF + "ShellExecResult: " + lsWork + " might not be configured properly, check the ServiceName and UserName. Proceeding..."
                    CommitLog lsHomeFldr, lsResult      :   '** NOTE ** Do not exit on error!!
                END IF
            END IF
        NEXT

        REM 'Post processing of earlier Stats snapshots
            If glLogStats then
                FinalizeLog                                 UBOUND( lsaFolderPath(2))
                FinalizeStats                               UBOUND( lsaFolderPath(2))
                'ConductAudit                       :   'Done with MoxLogsRecv.mox
                'CommitAudit                        :   'Done with MoxLogsRecv.mox
            End If

        REM 'Post processing of earlier Log snapshots
            If glLogRotate then
                FinalizeTwoFactor                           UBOUND( lsaFolderPath(2))
            End If

        REM 'Perform MoxLogsRecv.mox to do final processing and push stats into server
            IF LEN( DIR$(lsHomeFldr + "Moxie\Moxie.PreProc.mox")) THEN
                If glLogRotate or glLogStats THEN
                    'NOTE: This is a batch file, it has to be executed as:
                    ' Moxie.exe "Moxie.PreProc.mox?Stamp=2014-07-04-1800&Stats=y&Rotate=y&MoxieOnError=Kerry.Kelbert@KerrTek.net"
                    lsCommand                       =   "CD /D " + $DQ + lsHomeFldr + "Moxie" + $DQ + " && Moxie.exe " + $DQ + "Moxie.PreProc.mox"
                    lsCommand                       +=  "?Stamp=" & StampItDateHM
                    lsCommand                       +=  IIF$( glLogStats,  "&Stats=y", "&Stats=")
                    lsCommand                       +=  IIF$( glLogRotate, "&Rotate=y", "&Rotate=")
                    lsCommand                       +=  IIF$( glLogRotate, "&MoxieOnError=" & lsMoxieOnError, "&MoxieOnError=")
                    lsCommand                       +=  $DQ
                    llBrokenFlag                    =   ResultFlagOnShellExec( "ShellExec", lsWork, lsHomeFldr, lsCommand, lsResult, $Ignore, "")
                    mCommitLog
                END IF
            END IF

        REM 'Create and send email detailing failure/success codes then return Final Result code
           PrematureExit:

            Select Case as const glFinalResult
                case 0      : llIsCritical = %False     :
                CASE 1      : llIsCritical = %True      :   gsFinalMessage  +=  $CRLF + "Invalid command line arguments. "
                                                            gsFinalMessage  +=  $CRLF + " " + lsCommand
                                                            gsFinalMessage  +=  $CRLF + "Command line syntax is:"
                                                            gsFinalMessage  +=  $CRLF + " c:\Util\PreProc>%comspec% /z /c PreProc /logrotate /stats /quiet"
```

```
1000                                                gsFinalMessage  +=  $CRLF + "Or if run as a task:"
1001                                                gsFinalMessage  +=  $CRLF + " PreProc /logrotate /stats /quiet [ with path set to: c:\Util\PreProc ]"
1002
1003                                                'As CASE 1 can be triggered by improper command line options, allow a printed final result
1004                                                sFilePut        "PreProc.FinalResult.log", TRIM$(gsFinalMessage, $CRLF)
1005
1006                case 2      : llIsCritical = %False     :   'text has been handled already
1007                case 3      : llIsCritical = %True      :   'text has been handled already
1008                case 4      : llIsCritical = %True      :   'text has been handled already
1009                case 5      : llIsCritical = %True      :   'text has been handled already
1010                case 6      : llIsCritical = %True      :   'text has been handled already
1011                case 7      : llIsCritical = %True      :   'text has been handled already
1012                case 8      : llIsCritical = %True      :   'text has been handled already
1013                case else   : llIsCritical = %True      :   gsFinalMessage  +=  $CRLF + "An un-handled message (gsFinalMessage = " + format$(glFinalResult) + ") has occured; hope thats ok."
1014            End Select
1015
1016            If glLogQuiet THEN
1017                CommitLog                               lsHomeFldr, TRIM$( lsResult, $CRLF) + $CRLF + TRIM$( gsFinalMessage, $CRLF)
1018            Else
1019                REM 'Make Email Header, Message, etc
1020                    lsEmailSubject                  =   "[PreProc] " + ENVIRON$("COMPUTERNAME") + " " + IIF$(llIsCritical, "Error", "Success")
1021                    lsPreProcEmail                  =   "[MAIL SERVER]"                 + $CRLF + Trim$( lsMailServer, $CRLF)      + $CRLF + $CRLF
1022                    lsPreProcEmail                  +=  "[MAIL SERVER PORT]"            + $CRLF + Trim$( lsMailServerPort, $CRLF)  + $CRLF + $CRLF
1023                    lsPreProcEmail                  +=  "[FROM]"                        + $CRLF + Trim$( lsMailFrom, $CRLF)        + $CRLF + $CRLF
1024                    lsPreProcEmail                  +=  "[SUBJECT]"                     + $CRLF + Trim$( lsEmailSubject, $CRLF)    + $CRLF + $CRLF
1025                    lsPreProcEmail                  +=  "[EMAIL MESSAGE]"               + $CRLF + Trim$( gsFinalMessage, $CRLF)    + $CRLF + "[END]" + $CRLF + $CRLF
1026                    IF llIsCritical = %False Then
1027                        lsData                      =   lsEmailNoError
1028                    ELSE
1029                        lsData                      =   lsEmailOnError
1030                    END IF
1031
1032                REM 'Send email. One per address. Remove empty addresses.
1033                    llC1                            =   parsecount( lsData, $CRLF)
1034                    llC3                            =   0
1035                    Redim                               lsaData( 1 to llC1)
1036                    Parse                               lsData, lsaData(), $CRLF
1037                    for llC2 = 1 to llC1
1038                        lsTmp                       =   trim$( lsaData( llC2), ANY $WHITESPACE)
1039                        if len( lsTmp) THEN
1040                            incr                        llC3
1041                            lsaData( llC3)          =   lsTmp
1042                        End If
1043                    Next
1044                    If llC3 Then
1045                        for llC2 = 1 to llC3
1046                            sFilePut                    "PreProc.Email", lsPreProcEmail + "[SEND TO (ONE ADDRESS PER LINE)]"  + $CRLF + lsaData( llC2) + $CRLF + "[END]"
1047                            SHELL                       ENVIRON$("COMSPEC") + " /C  PwrMail.exe PreProc.email 2>&1" , 0
1048                            KILL                        "PreProc.Email"
1049                        nEXT
1050                    End If
1051            End If
1052
1053            mCommitLog
1054
1055        REM 'Move/Zip PreProc, MoxView & MoxLogsRecv.mox logs: 1x/day
1056            if glLogRotate Then
1057                'Clear LOCAL memory space
1058                    lsResult                        =   ""
1059                    lsTmp                           =   ""
1060                    lsDest                          =   lsBackupRootFolder + "zPreProc\" + StampItDateHM + ".zip"
1061
1062                'Amalgamate all xxx.log.tab files into one (use MergeTabFiles.bas code)
1063                    For llFolderPathNum = 1 to UBOUND( lsaFolderPath(2))
1064                        sFileGet                        ".\Moxie\Recv\" + format$(llFolderPathNum, "000") + ".log.tab", lsData, llLof
1065                        if llFolderPathNum = 1 then     lsHeader = Extract$( lsData, $CRLF)
1066                        lsTmp                       +=  $CRLF + REMAIN$( lsData, $CRLF)
1067                    Next
1068                    lsData                          =   ""
1069                    sFilePut                            ".\Moxie\Recv\log.tab", lsHeader + $crlf + Trim$( lsTmp, $CRLF)
1070                    lsTmp                           =   ""
1071
1072                'move all PreProc logs into Moxie\Recv\PreProc folder
1073                    lsCommand                       =   "MKDIR " + lsHomeFldr + "Moxie\Recv\PreProc"
1074                    llBrokenFlag                    =   ResultFlagOnShellExec( "ShellExec", "",  "", lsCommand, lsResult, $Ignore, "")
1075
1076                    lsCommand                       =   "MOVE /Y " + ".\PreProc\*.*" + " " + ".\Moxie\Recv\PreProc"
```

```
1077                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $Ignore, "")
1078
1079                'Copy SendGrid results into Moxie\Recv folder folder
1080                    lsCommand                           =   "COPY /Y " + ".\Moxie\Work\SendGrid\*.tab" + " " + ".\Moxie\Recv"
1081                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $Ignore, "")
1082
1083                'Move all Moxie logs into Moxie\Recv\Moxie folder folder
1084                    lsCommand                           =   "MKDIR " + lsHomeFldr + "Moxie\Recv\Moxie"
1085                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", "",  "", lsCommand, lsResult, $Ignore, "")
1086
1087                    lsCommand                           =   "MOVE /Y " + ".\Moxie\History\*.*" + " " + ".\Moxie\Recv\Moxie"
1088                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", "", "", lsCommand, lsResult, $Ignore, "")
1089            End If
1090
1091            if glLogRotate OR glAllOff OR glAllOn Then
1092                'Parse lsMoxViewFolder - Moxie.cfg to find the ServiceName
1093                    sFileGet                            lsMoxViewFolder + "Moxie.cfg", lsData, llLen
1094                    lsFile                              =   TRIM$( EXTRACT$( REMAIN$( lsData, $cfgServiceName ), $CRLF))
1095            End If
1096
1097            if glLogRotate OR glAllOff Then
1098                'Stop MoxView
1099                    lsCommand                           =   "NET STOP " + $DQ + lsFile + $DQ
1100                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", lsMoxViewFolder, "", lsCommand, lsResult, $Ignore, "")
1101            End If
1102
1103            if glLogRotate Then
1104                'Move all MoxView\History into Moxie\Recv\MoxView folder
1105                    lsCommand                           =   "MKDIR " + lsHomeFldr + "Moxie\Recv\MoxView"
1106                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", "",  "", lsCommand, lsResult, $Ignore, "")
1107
1108                    lsCommand                           =   "MOVE /Y " + lsMoxViewFolder + "History\*.*" + " " + lsHomeFldr + "Moxie\Recv\MoxView"
1109                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", lsMoxViewFolder,  "", lsCommand, lsResult, $Ignore, "")
1110            End If
1111
1112            if glLogRotate OR glAllOn Then
1113                'Start MoxView
1114                    lsCommand                           =   "NET START " + $DQ + lsFile + $DQ
1115                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec", lsMoxViewFolder, "", lsCommand, lsResult, $Ignore, "")
1116            End If
1117
1118            if glLogRotate Then
1119                'Perform Zip of Recv and move into C:\Backup\PreProc
1120                    lsCommand                           =   "7za a -tzip -r -ssw"
1121                    lsCommand                           +=  " -w" + $DQ + lsHomeFldr + $ProgramName + $DQ
1122                    lsCommand                           +=  " " + $DQ + lsDest + $DQ
1123                    lsCommand                           +=  " " + $DQ + lsHomeFldr + "Moxie\Recv" + $DQ
1124                    llBrokenFlag                        =   ResultFlagOnShellExec( "ShellExec7za", "", lsHomeFldr, lsCommand, lsResult, $Ignore, "")
1125            End If
1126
1127            if glLogRotate OR glAllOff OR glAllOn Then
1128                'leave everything in Recv as it will be removed on the next run
1129                    sFilePut                                "PreProc.FinalResult.log", TRIM$(lsResult, $CRLF)
1130            End If
1131
1132            FUNCTION                                    =   glFinalResult
1133    END FUNCTION
1134
1135 REM SubRoutines
1136
1137    SUB FinalizeTwoFactor( plFolderCount AS LONG)
1138        local lsData, lsHeader, lsaHeader() as string
1139        local llC1, llC2, llSite, llLof as long
1140
1141        Rem 'create header for log.tab
1142            sFileGet                                    ".\Moxie\Recv\twofactor.cnt", lsHeader, llLof
1143            KILL                                        ".\Moxie\Recv\twofactor.cnt"
1144
1145            llC1                                    =   parsecount( lsHeader, $SEP_NUL)
1146            Redim                                       lsaHeader( 1 to llC1)
1147            parse                                       lsHeader, lsaHeader(), $SEP_NUL
1148            for llC2 = 1 to llC1
1149                lsaHeader( llC2)                    =   parse$( lsaHeader( llC2), $SEP_SOH, 2)
1150            Next
1151
1152        REM 'Per Site
1153            For llSite = 1 to plFolderCount
```

```
1154                if DIR$(".\Moxie\Recv\" + format$(llSite, "000") + ".twofactor.cnd") = format$(llSite, "000") + ".twofactor.cnd" THEN
1155                    REM 'This iterations data
1156                        sFileGet                            ".\Moxie\Recv\" + format$(llSite, "000") + ".twofactor.cnd", lsData, llLof
1157                        KILL                                ".\Moxie\Recv\" + format$(llSite, "000") + ".twofactor.cnd"
1158
1159                    Rem 'Pack and save all data for the Moxie program & posterity
1160                        Replace                             $SEP_NUL with $CRLF in lsData
1161                        Replace                             $SEP_SOH with $TAB in lsData
1162                        sFilePut                            ".\Moxie\Recv\" + format$(llSite, "000") + ".twofactor.tab", join$( lsaHeader(), $Tab) + $crlf + lsData
1163                End If
1164            Next
1165    end sub
1166
1167    SUB FinalizeStats( plFolderCount AS LONG)
1168        local lsData, lsHeader, lsaHeader() as string
1169        local llC1, llC2, llSite, llLof as long
1170
1171        Rem 'create header for log.tab
1172            sFileGet                                        ".\Moxie\Recv\stats.cnt", lsHeader, llLof
1173            KILL                                            ".\Moxie\Recv\stats.cnt"
1174
1175            llC1                                    =       parsecount( lsHeader, $SEP_NUL)
1176            Redim                                           lsaHeader( 1 to llC1)
1177            parse                                           lsHeader, lsaHeader(), $SEP_NUL
1178            for llC2 = 1 to llC1
1179                lsaHeader( llC2)                    =       parse$( lsaHeader( llC2), $SEP_SOH, 2)
1180            Next
1181
1182        REM 'Per Site
1183            For llSite = 1 to plFolderCount
1184                REM 'This iterations data
1185                    sFileGet                                ".\Moxie\Recv\" + format$(llSite, "000") + ".stats.cnd", lsData, llLof
1186                    KILL                                    ".\Moxie\Recv\" + format$(llSite, "000") + ".stats.cnd"
1187
1188                Rem 'Pack and save all data for the Moxie program & posterity
1189                    Replace                                 $SEP_NUL with $CRLF in lsData
1190                    Replace                                 $SEP_SOH with $TAB in lsData
1191                    sFilePut                                ".\Moxie\Recv\" + format$(llSite, "000") + ".stats.tab", join$( lsaHeader(), $Tab) + $crlf + lsData
1192            Next
1193    end sub
1194
1195    SUB FinalizeLog( plFolderCount AS LONG)
1196        local lsFolderPath, lsIP1, lsIP2, lsIP3, lsIP4, lsData, lsaData(), lsCtry, lsaCtry(), lsaCountry(), lsCtryHeader, lsTmp, lsaTmp(), lsFinal, lsHeader, lsaHeader(), lsLastStatsSnapshot as string
1197        local llC1, llC2, llC3, llLof, llSite, llCtryCount as long
1198        local ldData, ldaCtry(), ldaCtry2() as dword
1199
1200        Rem 'create header for log.tab
1201            'note: the default sort order is 'most recent entry last'
1202            sFileGet                                        ".\Moxie\Recv\log.cnt", lsHeader, llLof
1203            KILL                                            ".\Moxie\Recv\log.cnt"
1204
1205            llC1                                    =       parsecount( lsHeader, $SEP_NUL)
1206            Redim                                           lsaHeader( 1 to llC1)
1207            parse                                           lsHeader, lsaHeader(), $SEP_NUL
1208            for llC2 = 1 to llC1
1209                lsaHeader( llC2)                    =       parse$( lsaHeader( llC2), $SEP_SOH, 2)
1210            Next
1211
1212        REM 'IPv4ToCountry: Load and prep for searches
1213            sFileGet                                        ".\IpToCountry\Ip2Country.tab", lsCtry, llLof
1214            lsCtryHeader                            =       Extract$( lsCtry, $CRLF)
1215            lsCtry                                  =       remain$( lsCtry, $CRLF)
1216            llCtryCount                             =       parsecount( lsCtry, $CRLF)
1217
1218            Redim                                           lsaCtry( 1 to llCtryCount)
1219            Redim                                           lsaCountry( 1 to llCtryCount)
1220            Redim                                           ldaCtry( 1 to llCtryCount)
1221            Redim                                           ldaCtry2( 1 to llCtryCount)
1222            Parse                                           lsCtry, lsaCtry(), $CRLF
1223
1224            For llC1 = 1 to llCtryCount
1225                ldaCtry(llC1)                       =       val( parse$( lsaCtry(llC1), $Tab, 1))
1226            Next
1227            ARRAY SORT ldaCtry(), TAGARRAY lsaCtry()
1228            For llC1 = 1 to llCtryCount
1229                ldaCtry2(llC1)                      =       val( parse$( lsaCtry(llC1), $Tab, 2))
1230                lsaCountry(llC1)                    =            parse$( lsaCtry(llC1), $Tab, 7)
```

```
1231            Next
1232
1233        REM 'Per Site
1234            For llSite = 1 to plFolderCount
1235
1236                REM 'This iterations snapshot datestamp
1237                    sFileGet                                ".\Moxie\Recv\" + format$(llSite, "000") + ".snapshot.tab", lsData, llLof
1238                    lsLastStatsSnapshot             =   Remain$( lsData, $CRLF)
1239                    lsLastStatsSnapshot             =   EXTRACT$( lsLastStatsSnapshot, $TAB)
1240
1241                REM 'This iterations data
1242                    sFileGet                                ".\Moxie\Recv\" + format$(llSite, "000") + ".log.cnd", lsData, llLof
1243                    KILL                                    ".\Moxie\Recv\" + format$(llSite, "000") + ".log.cnd"
1244                    llC1                            =   parsecount( lsData, $SEP_NUL)
1245                    Redim                                   lsaData( 1 to %Tedata + 1, 1 to llC1)
1246                    Redim                                   lsaTmp( 1 to llC1)
1247
1248                    Parse                                   lsData, lsaTmp(), $SEP_NUL
1249
1250                    For llC2 = 1 to llC1
1251                        For llC3 = 1 to %Tedata
1252                            lsaData( llC3, llC2)        =   parse$( lsaTmp( llC2), $SEP_SOH, llC3)
1253                        Next
1254                    Next
1255
1256                REM 'IP address lookup is done by using IP10 values
1257                    'http://stackoverflow.com/questions/10668831/how-to-efficiently-index-and-search-ipv4-address-ranges-in-c-sharp
1258                    'http://en.wikipedia.org/wiki/Binary_search_algorithm#Recursive
1259                    For llC2 = 1 to llC1
1260                        If LEN( lsaData( %F09eIPNum, llC2))  _
1261                        AND "127.0.0.1"         <> lsaData( %F09eIPNum, llC2) _
1262                        AND "ECom"              <> lsaData( %F05eCategory, llC2) _
1263                        AND lsLastStatsSnapshot <= lsaData( %F03eTimeStamp, llC2) _
1264                        Then
1265                            lsIP1                   =   parse$( lsaData( %F09eIPNum, llC2), ".", 1)
1266                            lsIP2                   =   parse$( lsaData( %F09eIPNum, llC2), ".", 2)
1267                            lsIP3                   =   parse$( lsaData( %F09eIPNum, llC2), ".", 3)
1268                            lsIP4                   =   parse$( lsaData( %F09eIPNum, llC2), ".", 4)
1269                            ldData                  =   (val(lsIP1) * 16777216) + (val(lsIP2) * 65536) + (val(lsIP3) * 256) + val(lsIP4)
1270
1271                            Rem 'for a given data record, find where it fits between in an Ctry record
1272                                For llC3 = 1 to llCtryCount
1273                                    If ldData => ldaCtry(llC3) _
1274                                    AND ldData < ldaCtry2(llC3) _
1275                                    THEN
1276                                        lsaData( %F21eCountry, llC2) = lsaCountry(llC3)
1277                                        exit
1278                                    END IF
1279                                NEXT
1280                        End If
1281                    Next
1282
1283                REM 'Pack and save this iterations data
1284                    For llC2 = 1 to llC1
1285                        lsaTmp( llC2)               =   lsaData( %F00eAlias             , llC2) + $Tab + _
1286                                                        lsaData( %F01eFolderName        , llC2) + $Tab + _
1287                                                        lsaData( %F02eLogLocation       , llC2) + $Tab + _
1288                                                        lsaData( %F03eTimeStamp         , llC2) + $Tab + _
1289                                                        lsaData( %F04eThread            , llC2) + $Tab + _
1290                                                        lsaData( %F05eCategory          , llC2) + $Tab + _
1291                                                        lsaData( %F06eMessage           , llC2) + $Tab + _
1292                                                        lsaData( %F07eDetails           , llC2) + $Tab + _
1293                                                        lsaData( %F08eBytes             , llC2) + $Tab + _
1294                                                        lsaData( %F09eIPNum             , llC2) + $Tab + _
1295                                                        lsaData( %F10eThreadNum         , llC2) + $Tab + _
1296                                                        lsaData( %F11eThreadTime        , llC2) + $Tab + _
1297                                                        lsaData( %F12ePersonUUID        , llC2) + $Tab + _
1298                                                        lsaData( %F13eSessionUUID       , llC2) + $Tab + _
1299                                                        lsaData( %F14eDatum1            , llC2) + $Tab + _
1300                                                        lsaData( %F15eDatum2            , llC2) + $Tab + _
1301                                                        lsaData( %F16eDatum3            , llC2) + $Tab + _
1302                                                        lsaData( %F17eDatum4            , llC2) + $Tab + _
1303                                                        lsaData( %F18eMessageCode       , llC2) + $Tab + _
1304                                                        lsaData( %F19eMoxVerMajor       , llC2) + $Tab + _
1305                                                        lsaData( %F20eMoxVerMinor       , llC2) + $Tab + _
1306                                                        lsaData( %F21eCountry           , llC2)
1307                    Next
```

```
1308
1309                Rem 'Pack and save all data for the Moxie program & posterity
1310                    sFilePut                                ".\Moxie\Recv\" + format$(llSite, "000") + ".log.tab", join$( lsaHeader(), $Tab) + $crlf + Join$( lsaTmp(), $CrLf)
1311            Next
1312    END SUB
1313
1314    FUNCTION UpdateTally( psIndex AS STRING, psaIndex() AS STRING, psaTally() AS STRING) AS LONG
1315        LOCAL lsIndex AS STRING
1316        LOCAL llIndex, llC1 AS LONG
1317
1318        llIndex                                     =   UBOUND(psaIndex(1))
1319        lsIndex                                     =   TRIM$(psIndex)
1320
1321        IF LEN(lsIndex) THEN
1322            IF llIndex THEN
1323                FOR llC1 = 1 TO llIndex
1324                    IF  psaIndex(llC1) = lsIndex THEN       EXIT FOR
1325                NEXT
1326                IF llC1 > llIndex THEN
1327                    INCR                                    llIndex
1328                    REDIM                                   PRESERVE psaIndex(1 TO llIndex), psaTally(1 TO llIndex)
1329                    psaIndex(llIndex)               =   lsIndex
1330                    psaTally(llIndex)               =   FORMAT$(1)
1331                    ARRAY SORT                              psaIndex(), COLLATE UCASE, TAGARRAY psaTally()
1332                ELSEIF psaIndex(llC1) = lsIndex THEN
1333                    psaTally(llC1)                  =   FORMAT$(VAL(psaTally(llC1)) + 1)
1334                END IF
1335            ELSE
1336                INCR llIndex
1337                REDIM psaIndex(1 TO llIndex), psaTally(1 TO llIndex)
1338                psaIndex(1)                         =   lsIndex
1339                psaTally(1)                         =   FORMAT$(1)
1340            END IF
1341        END IF
1342    END FUNCTION
1343
1344    SUB GatherLogFilesAndPreProcess( psaFolderPath() as string, plFolderPathNum as long, psaFiles() as string, psResult as string)
1345        LOCAL lsaFiles() AS STRING
1346        LOCAL lsVerMajor, lsVerMinor, lsTmp, lsaTmp(), lsaSrt(), lsTemp, lsData, lsaData() as string
1347        LOCAL llC1, llC2, llC3, llC4, llLOF AS LONG
1348
1349        REM 'seperate all the log files found into lines
1350            For llC1 = 1 to UBOUND(psaFiles)
1351                sFileGet                                psaFiles(llC1), lsTmp, llLOF
1352                lsTmp                               =   Trim$( lsTmp, $CRLF)
1353                llC4                                =   parsecount( lsTmp, $CRLF)
1354                Redim                                   lsaTmp( 1 to llC4)
1355                Parse                                   lsTmp, lsaTmp(), $CRLF
1356                For llC3 = 1 to llC4
1357                    lsaTmp(llC3)                    =   psaFolderPath( %FolderObject, plFolderPathNum) + $Tab + psaFiles(llC1) + $Tab + lsaTmp(llC3)
1358                Next
1359                lsData                              +=  Join$( lsaTmp(), $CRLF) + $CRLF
1360            Next
1361
1362        lsData                                      =   Trim$( lsData, $Crlf)
1363
1364        REM 'and now start to make a single tab delimited table
1365            llC1                                    =   parsecount( lsData, $CRLF)
1366
1367            Redim                                       lsaTmp( 1 to llC1)
1368            Parse                                       lsData, lsaTmp(), $CRLF
1369            Reset                                       lsData
1370
1371            Redim                                       lsaData( 1 to %Tdata, 1 to llC1)
1372            For llC2 = 1 to llC1
1373                For llC3 = 1 to %Tdata
1374                    lsaData( llC3, llC2)            =   PARSE$(lsaTmp(llC2), $TAB, llC3)
1375                Next
1376            NEXT
1377            RESET                                       lsaTmp()
1378
1379        REM 'While everything is in sequence, save the Moxie Version decryption key
1380            For llC2 = 1 to llC1
1381                Select Case as const$ LCASE$( lsaData( %F05_Category , llC2))
1382                    case    "-server started-", "-server stopped-", "-cmdmox started-", "-cmdmox stopped-"
1383                            lsVerMajor = ""
1384                            lsVerMinor = ""
```

```
1385
1386                    case    "moxie version"
1387                            lsVerMajor = Extract$( lsaData( %F06_Message, llC2), " ")
1388                            lsVerMinor = remain$(  lsaData( %F06_Message, llC2), " ")
1389
1390                    CASE    "http", "https"
1391                            llC4       = Val( Extract$( lsaData( %F06_Message, llC2), " "))
1392                            If llC4 Then lsaData( %F18_MessageCode, llC2) = format$( llC4)
1393
1394                    CASE    "run time error"    :   'Deconstruct any run time error
1395                            lsaData( %F06_Message, llC2) = _
1396                                lsaData( %F06_Message       , llC2) + $SEP_TAB + _
1397                                lsaData( %F07_Details       , llC2) + $SEP_TAB + _
1398                                lsaData( %F08_Bytes         , llC2) + $SEP_TAB + _
1399                                lsaData( %F09_IPNum         , llC2) + $SEP_TAB + _
1400                                lsaData( %F10_ThreadNum     , llC2) + $SEP_TAB + _
1401                                lsaData( %F11_ThreadTime    , llC2) + $SEP_TAB + _
1402                                lsaData( %F12_PersonUUID    , llC2) + $SEP_TAB + _
1403                                lsaData( %F13_SessionUUID   , llC2) + $SEP_TAB + _
1404                                lsaData( %F14_Datum1        , llC2) + $SEP_TAB + _
1405                                lsaData( %F15_Datum2        , llC2) + $SEP_TAB + _
1406                                lsaData( %F16_Datum3        , llC2) + $SEP_TAB + _
1407                                lsaData( %F17_Datum4        , llC2) + $SEP_TAB + _
1408                                lsaData( %F18_MessageCode   , llC2) + $SEP_TAB
1409
1410                            lsaData( %F06_Message, llC2) = TRIM$( lsaData( %F06_Message, llC2), $SEP_TAB)
1411                                lsaData( %F07_Details       , llC2) = ""
1412                                lsaData( %F08_Bytes         , llC2) = ""
1413                                lsaData( %F09_IPNum         , llC2) = ""
1414                                lsaData( %F10_ThreadNum     , llC2) = ""
1415                                lsaData( %F11_ThreadTime    , llC2) = ""
1416                                lsaData( %F12_PersonUUID    , llC2) = ""
1417                                lsaData( %F13_SessionUUID   , llC2) = ""
1418                                lsaData( %F14_Datum1        , llC2) = ""
1419                                lsaData( %F15_Datum2        , llC2) = ""
1420                                lsaData( %F16_Datum3        , llC2) = ""
1421                                lsaData( %F17_Datum4        , llC2) = ""
1422                                lsaData( %F18_MessageCode   , llC2) = ""
1423
1424                    CASE    "login"        :   'Reconstruct Login
1425                            lsaData( %F12_PersonUUID, llC2) = lsaData( %F09_IPNum,   llC2)        :   lsaData( %F09_IPNum,   llC2) = ""
1426                            lsaData( %F15_Datum2,     llC2) = lsaData( %F08_Bytes,   llC2)        :   lsaData( %F08_Bytes,   llC2) = ""
1427                            lsaData( %F14_Datum1,     llC2) = lsaData( %F07_Details, llC2)        :   lsaData( %F07_Details, llC2) = ""
1428
1429                    CASE    "bademail"     :   'Reconstruct BadEmail
1430                            lsaData( %F16_Datum3,     llC2) = lsaData( %F09_IPNum,   llC2)        :   lsaData( %F09_IPNum,   llC2) = ""
1431                            lsaData( %F15_Datum2,     llC2) = lsaData( %F08_Bytes,   llC2)        :   lsaData( %F08_Bytes,   llC2) = ""
1432                            lsaData( %F14_Datum1,     llC2) = lsaData( %F07_Details, llC2)        :   lsaData( %F07_Details, llC2) = ""
1433
1434                        Case else
1435                    End Select
1436
1437                    lsaData( %F19_MoxVerMajor  , llC2)      =   lsVerMajor
1438                    lsaData( %F20_MoxVerMinor  , llC2)      =   lsVerMinor
1439                Next
1440
1441            REM 'F07 breakdown details into datum with error flag
1442                For llC2 = 1 to llC1
1443                    If Val( lsaData( %F18_MessageCode, llC2)) Then
1444                        lsaData( %F14_Datum1, llC2)         =   lcase$( Extract$( lsaData( %F07_Details, llC2), " "))
1445                        lsTmp                               =   lcase$( REMAIN$( lsaData( %F07_Details, llC2), " "))
1446
1447                        lsaData( %F17_Datum4, llC2)         =   REMAIN$( lsTmp, " ")
1448                        lsTmp                               =   Extract$( lsTmp, " ")
1449
1450                        lsaData( %F15_Datum2, llC2)         =   Extract$( lsTmp, "?")
1451                        lsaData( %F16_Datum3, llC2)         =   REMAIN$( lsTmp, "?")
1452                    End If
1453
1454                    Select Case as const$ LCASE$( lsaData( %F17_Datum4, llC2))
1455                        case    "http/1.1", "http/1.0"
1456                                'Fall Through
1457                        Case else
1458                                If len( lsaData( %F18_MessageCode, llC2)) Then lsaData( %F17_Datum4, llC2)    =   "*"
1459                    End Select
1460
1461                    Select Case as const$ LCASE$( lsaData( %F14_Datum1, llC2))
```

```
1462                    case    "connect", "trace", "delete", "put", "post", "head", "get", "options"
1463                    'Fall Through
1464                    Case else
1465                            If len( lsaData( %F18_MessageCode, llC2)) Then lsaData( %F17_Datum4, llC2)    =   "*"
1466                End Select
1467            Next
1468
1469        REM 'Prep, sort and apply an Alias
1470            Redim                                       lsaTmp( 1 to llC1), lsaSrt( 1 to llC1)
1471            For llC2 = 1 to llC1
1472                lsaSrt( llC2)                       =   lsaData( %F03_TimeStamp, llC2)  + $Tab + _
1473                                                        format$(plFolderPathNum, "000") + _
1474                                                        Format$(llC2, "0000000")
1475
1476                lsaTmp( llC2)                       =   lsaData( %F01_FolderName       , llC2) + $Tab + _
1477                                                        lsaData( %F02_LogLocation      , llC2) + $Tab + _
1478                                                        lsaData( %F03_TimeStamp        , llC2) + $Tab + _
1479                                                        lsaData( %F04_Thread           , llC2) + $Tab + _
1480                                                        lsaData( %F05_Category         , llC2) + $Tab + _
1481                                                        lsaData( %F06_Message          , llC2) + $Tab + _
1482                                                        lsaData( %F07_Details          , llC2) + $Tab + _
1483                                                        lsaData( %F08_Bytes            , llC2) + $Tab + _
1484                                                        lsaData( %F09_IPNum            , llC2) + $Tab + _
1485                                                        lsaData( %F10_ThreadNum        , llC2) + $Tab + _
1486                                                        lsaData( %F11_ThreadTime       , llC2) + $Tab + _
1487                                                        lsaData( %F12_PersonUUID       , llC2) + $Tab + _
1488                                                        lsaData( %F13_SessionUUID      , llC2) + $Tab + _
1489                                                        lsaData( %F14_Datum1           , llC2) + $Tab + _
1490                                                        lsaData( %F15_Datum2           , llC2) + $Tab + _
1491                                                        lsaData( %F16_Datum3           , llC2) + $Tab + _
1492                                                        lsaData( %F17_Datum4           , llC2) + $Tab + _
1493                                                        lsaData( %F18_MessageCode      , llC2) + $Tab + _
1494                                                        lsaData( %F19_MoxVerMajor      , llC2) + $Tab + _
1495                                                        lsaData( %F20_MoxVerMinor      , llC2) + $Tab + _
1496                                                        lsaData( %F21_Country          , llC2)
1497
1498            NEXT
1499
1500            ARRAY SORT                                  lsaSrt(), COLLATE UCASE, TAGARRAY lsaTmp()
1501
1502            For llC2 = 1 to llC1
1503                lsaTmp( llC2)                       =   format$(plFolderPathNum, "000") + Format$(llC2, "0000000") + $Tab + lsaTmp( llC2)
1504            Next
1505
1506            Reset lsaData()
1507
1508            lsData                                  =   Join$( lsaTmp(), $CRLF)
1509            Reset                                       lsaTmp()
1510
1511            REPLACE                                     $CRLF WITH chr$(0) IN lsData
1512            REPLACE                                     $Tab WITH chr$(1) IN lsData
1513            sFilePut                                    ".\Moxie\Recv\" + format$(plFolderPathNum, "000") + ".log.cnd", lsData
1514
1515            lsTmp  = "Alias" + chr$(1) + "Alias"            + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1516            lsTmp += "F01"   + chr$(1) + "F01FolderName"    + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1517            lsTmp += "F02"   + chr$(1) + "F02LogLocation"   + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1518            lsTmp += "F03"   + chr$(1) + "F03TimeStamp"     + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1519            lsTmp += "F04"   + chr$(1) + "F04Thread"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1520            lsTmp += "F05"   + chr$(1) + "F05Category"      + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1521            lsTmp += "F06"   + chr$(1) + "F06Message"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1522            lsTmp += "F07"   + chr$(1) + "F07Details"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1523            lsTmp += "F08"   + chr$(1) + "F08Bytes"         + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1524            lsTmp += "F09"   + chr$(1) + "F09IPNum"         + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1525            lsTmp += "F10"   + chr$(1) + "F10ThreadNum"     + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1526            lsTmp += "F11"   + chr$(1) + "F11ThreadTime"    + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1527            lsTmp += "F12"   + chr$(1) + "F12PersonUUID"    + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1528            lsTmp += "F13"   + chr$(1) + "F13SessionUUID"   + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1529            lsTmp += "F14"   + chr$(1) + "F14Datum1"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1530            lsTmp += "F15"   + chr$(1) + "F15Datum2"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1531            lsTmp += "F16"   + chr$(1) + "F16Datum3"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1532            lsTmp += "F17"   + chr$(1) + "F17Datum4"        + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1533            lsTmp += "F18"   + chr$(1) + "F18MessageCode"   + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1534            lsTmp += "F19"   + chr$(1) + "F19MoxVerMajor"   + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1535            lsTmp += "F20"   + chr$(1) + "F20MoxVerMinor"   + chr$(1) + "" + chr$(1) + "[R]" + chr$(1) + chr$(0)
1536            lsTmp += "F21"   + chr$(1) + "F21Country"       + chr$(1) + "" + chr$(1) + "[R]" + chr$(1)
1537
1538            sFilePut                                    ".\Moxie\Recv\log.cnt", TRIM$( lsTmp, chr$(0))
```

```
1539            Reset                                       lsTmp
1540        END SUB
1541
1542        SUB sGetFileNamesOnly( psaFiles() AS STRING, psFolder AS STRING, plSubFolders as long)
1543            LOCAL llC1 AS LONG
1544            Local lsFolder, lsThisFile as string
1545            REDIM                                       psaFiles(1 TO 10) : llC1 = 1
1546            'Start with the current (default) directory
1547                lsFolder = ""                       :   Gosub FillAndLoop
1548
1549            'Check each possable directory using bitwise AND
1550                If ( plSubFolders AND %dir_DBStore  ) Then lsFolder = "DBStore"     :   Gosub FillAndLoop
1551                If ( plSubFolders AND %dir_History  ) Then lsFolder = "History"     :   Gosub FillAndLoop
1552                If ( plSubFolders AND %dir_Lib      ) Then lsFolder = "Lib"         :   Gosub FillAndLoop
1553                If ( plSubFolders AND %dir_Private  ) Then lsFolder = "Private"     :   Gosub FillAndLoop
1554                If ( plSubFolders AND %dir_Public   ) Then lsFolder = "Public"      :   Gosub FillAndLoop
1555                If ( plSubFolders AND %dir_Recv     ) Then lsFolder = "Recv"        :   Gosub FillAndLoop
1556                If ( plSubFolders AND %dir_Templates) Then lsFolder = "Templates"   :   Gosub FillAndLoop
1557                If ( plSubFolders AND %dir_Work     ) Then lsFolder = "Work"        :   Gosub FillAndLoop
1558
1559            REDIM                                       PRESERVE psaFiles(1 TO llC1 - 1)
1560            Exit                                        Sub
1561
1562            FillAndLoop:
1563                If Len(lsFolder) Then
1564                    psaFiles(llC1)                  =   lsFolder + "\" + DIR$(TRIM$(psFolder, "\") + "\" + lsFolder + "\*.*")
1565                Else
1566                    psaFiles(llC1)                  =   DIR$(TRIM$(psFolder, "\") + "\*.*")
1567                End If
1568                DO
1569                    INCR llC1
1570                    IF llC1 > UBOUND(psaFiles) THEN         REDIM PRESERVE psaFiles(1 TO llC1 * 1.5)
1571                    lsThisFile                      =   DIR$
1572                    If Len(lsFolder) Then
1573                        psaFiles(llC1)              =   lsFolder + "\" + lsThisFile
1574                    Else
1575                        psaFiles(llC1)              =   lsThisFile
1576                    End If
1577                LOOP WHILE LEN(lsThisFile)
1578            Return
1579        END SUB
1580
1581        Function ShellExec( psWorkFldr AS STRING, psHomeFldr as string, psShellCommand AS STRING, psShellCommandResult AS STRING) as long
1582            LOCAL llLen AS LONG
1583            LOCAL lsPreProcFilePath AS STRING
1584
1585            IF psWorkFldr = "" THEN
1586                lsPreProcFilePath               =   ".\" + $PreProcTempFile
1587            ELSE
1588                lsPreProcFilePath               =   RTRIM$( TRIM$(psWorkFldr), ANY " \") + "\" + $PreProcTempFile
1589            END IF
1590
1591            IF INSTR(lsPreProcFilePath, $SPC) THEN
1592                lsPreProcFilePath               =   $DQ + lsPreProcFilePath + $DQ
1593            END IF
1594
1595            KILL lsPreProcFilePath  :   'Remove any prevously failed temp file
1596            SHELL                                       ENVIRON$("COMSPEC") + " /C " + psShellCommand + " > " & lsPreProcFilePath & " 2>&1", 0
1597
1598            sFileGet                                    lsPreProcFilePath, psShellCommandResult, llLen
1599            sFilePut                                    lsPreProcFilePath, ""
1600            KILL                                        lsPreProcFilePath
1601
1602            REM ' Some programs do not play nice with DOS Error messages
1603                REPLACE                                 $CR + $CR WITH $CR IN psShellCommandResult
1604                psShellCommandResult            =   RTRIM$( psShellCommandResult, $CRLF)
1605        END FUNCTION
1606
1607        FUNCTION ShellExec7za( _                        :   ' Set's up a "local" environment so that the 7za command can use relative file paths
1608                    psWorkFldr AS STRING, _             :   ' Set this to be the 'current folder' for the 7za command to execute
1609                    _                                       ' it cannot be empty and must be an absolute path
1610                    _                                       ' The original folder is restored automatically on return
1611                    psHomeFldr as string, _             :   ' Where our 7za.exe file is. Cannot be empty.
1612                    psShellCommand AS STRING, _         :   ' The DOS command to execute
1613                    psShellCommandResult AS STRING _    :   ' The results of the command
1614                    ) as long
1615            LOCAL llLen, llRet, llErr AS LONG
```

```
1616        local lsWorkFldr, lsHomeFldr, lsOrigFldr, lsData as string
1617
1618        llErr                                       =   %False
1619
1620        lsWorkFldr                                  =   RTRIM$( psWorkFldr, ANY " \") + "\"
1621        lsHomeFldr                                  =   RTRIM$( psHomeFldr, ANY " \") + "\"
1622        lsOrigFldr                                  =   RTRIM$( CURDIR$, ANY " \") + "\"
1623
1624        REM ' Basic error checking and init
1625            If psWorkFldr = "\" _
1626            OR psHomeFldr = "\" _
1627            OR LEN( DIR$(psHomeFldr + "7za.exe")) = 0 THEN
1628                llErr                               =   %True
1629                GOTO                                    ShellExec7zaDone
1630            END IF
1631
1632            If lsHomeFldr = lsWorkFldr THEN
1633                llErr                               =   %True
1634                GOTO                                    ShellExec7zaDone
1635            END IF
1636
1637        REM 'Prep the 'local' folder
1638            If lsOrigFldr <> lsWorkFldr THEN
1639                CHDRIVE                                 Extract$(psWorkFldr, ":")
1640                chdir                                   REMAIN$(psWorkFldr, ":")
1641            End If
1642
1643        REM 'Prep the 7za command to execute as a 'local' command with our current privelige level
1644            sFileGet                                    lsHomeFldr + "7za.exe", lsData, llLen
1645            If llLen < 2 THEN
1646                llErr                               =   %True
1647                GOTO                                    ShellExec7zaDone
1648            End If
1649            sFilePut                                    lsWorkFldr + "7za.exe", lsData
1650            KILL                                        "7za.tmp"  :   'Remove any prevously failed zip tmp file
1651
1652            SHELL                                       ENVIRON$("COMSPEC") + " /C " + psShellCommand + " > 7za.tmp 2>&1" , 0
1653            sFileGet                                    "7za.tmp", psShellCommandResult, llLen
1654            sFilePut                                    "7za.tmp", ""
1655            KILL                                        "7za.tmp"
1656            KILL                                        lsWorkFldr + "7za.exe"
1657
1658        REM ' Restore
1659            If lsOrigFldr <> lsWorkFldr THEN
1660                CHDRIVE                                 Extract$(lsOrigFldr, ":")
1661                chdir                                   REMAIN$(lsOrigFldr, ":")
1662            End If
1663
1664        REM ' Some programs do not play nice with DOS Error messages
1665            REPLACE                                     $CR + $CR WITH $CR IN psShellCommandResult
1666            psShellCommandResult                    =   RTRIM$( psShellCommandResult, $CRLF)
1667
1668        REM ' Done
1669            ShellExec7zaDone:
1670            FUNCTION                                =   llErr
1671    END FUNCTION
1672
1673    FUNCTION ResultFlagOnShellExec( _
1674                    psCommand as string, _              :   '
1675                    psWorkFldr AS STRING, _             :   '
1676                    psHomeFldr as string, _             :   '
1677                    psShellCommand AS STRING, _         :   '
1678                    psResultLog AS STRING, _            :   '
1679                    psFailFlagCriteria AS STRING, _     :   '
1680                    psFailFlagTestString AS STRING _    :   '
1681                    ) AS LONG
1682        GLOBAL glFinalResult AS LONG
1683        global gsFinalMessage as string
1684        LOCAL llFailFlag, llErrorFlag AS LONG
1685        LOCAL lsTmpLog, lsResult, lsResultMessage, lsFailFlagTestString AS STRING
1686
1687        llErrorFlag                                 =  %False
1688
1689        If      LCASE$( psCommand) = LCASE$( "ShellExec") Then
1690            psResultLog                             +=  $CRLF + "ShellExec: " + psShellCommand
1691            llErrorFlag                             =  ShellExec( psWorkFldr, psHomeFldr, psShellCommand, lsTmpLog)
1692
```

```
1693        Elseif LCASE$( psCommand) = LCASE$( "ShellExec7za") Then
1694            psResultLog                                 +=  $CRLF + "ShellExec7za: " + psShellCommand
1695            llErrorFlag                                 =  ShellExec7za( psWorkFldr, psHomeFldr, psShellCommand, lsTmpLog)
1696
1697        ELSE
1698            psResultLog                                 +=  $CRLF + "ResultFlagOnShellExec: Error: invalid command " + psShellCommand
1699            llErrorFlag                                 =  %True
1700            goto                                           ResultFlagOnShellExecDone
1701        End If
1702
1703        lsResult                                        =   LCASE$(lsTmpLog)
1704        lsResult                                        =   TRIM$( lsResult)
1705        lsResult                                        =   TRIM$( lsResult, $TAB)
1706        lsResult                                        =   TRIM$( lsResult, $CRLF)
1707        lsResult                                        =   TRIM$( lsResult, $CR)
1708        lsResult                                        =   TRIM$( lsResult, $LF)
1709
1710        lsFailFlagTestString                            =   LCASE$( psFailFlagTestString)
1711        llFailFlag                                      =   INSTR( lsResult, lsFailFlagTestString)
1712
1713        SELECT CASE psFailFlagCriteria
1714            CASE $Ignore
1715                ' Fall Through
1716
1717            CASE $FlagIfEmpty
1718                IF LEN( lsResult) = 0 THEN
1719                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command " & psFailFlagCriteria
1720                END IF
1721
1722            CASE $FlagIfNotEmpty
1723                IF LEN( lsResult) <> 0 THEN
1724                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command " & psFailFlagCriteria
1725                END IF
1726
1727            CASE $FlagIfIncludes
1728                IF LEN(lsFailFlagTestString) = 0 THEN
1729                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command " & psFailFlagCriteria & " Invalid Test"
1730                END IF
1731                IF llFailFlag <> 0 THEN
1732                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command " & psFailFlagCriteria
1733                ELSE
1734                    ' Fall Through
1735                END IF
1736
1737            CASE $FlagIfNotIncludes
1738                IF LEN(lsFailFlagTestString) = 0 THEN
1739                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command " & psFailFlagCriteria & " Invalid Test"
1740                END IF
1741                IF llFailFlag = 0 THEN
1742                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command " & psFailFlagCriteria
1743                ELSE
1744                    ' Fall Through
1745                END IF
1746
1747            CASE ELSE
1748                    llErrorFlag   = %True               :   lsResultMessage = "Error: " + psCommand + " Command $UnHandled"
1749        END SELECT
1750
1751        REPLACE                                             $CRLF WITH  $SEP_CRLF IN lsTmpLog
1752        REPLACE                                             $CR WITH    $SEP_CR   IN lsTmpLog
1753        REPLACE                                             $LF WITH    $SEP_LF   IN lsTmpLog
1754
1755        IF psFailFlagCriteria = $Ignore THEN
1756            psResultLog                                 +=  $CRLF + "ResultFlagOnShellExec: " + "$Ignore" + "<ResultFlagOnShellExec>" + lsTmpLog + "</ResultFlagOnShellExec>"
1757        ELSE
1758            psResultLog                                 +=  $CRLF + "ResultFlagOnShellExec: " + lsResultMessage + "<ResultFlagOnShellExec>" + lsTmpLog + "</ResultFlagOnShellExec>"
1759        END IF
1760
1761        ResultFlagOnShellExecDone:
1762        IF llErrorFlag THEN
1763            glFinalResult                               =   5
1764            gsFinalMessage                              +=  $CRLF + "Non fatal command failure: " + lsResultMessage + ". See logs; proceeding..."
1765        End If
1766        FUNCTION = llErrorFlag
1767    END FUNCTION
1768
1769    SUB sCaptureConfiguration( psWorkFldr AS STRING, _  :   '
```

```
1770                    psaFolderPath() AS STRING, _        :   ’
1771                    psBackupRootFolder as string, _     :   ’
1772                    psStatsRecvFolder as string, _      :   ’
1773                    psMoxViewFolder as string, _        :   ’
1774                    psMailServer as string, _           :   ’
1775                    psMailServerPort as string, _       :   ’
1776                    psMailFrom as string, _             :   ’
1777                    psForceCnwEmailErr as string, _     :   ’
1778                    psEmailOnError as string, _         :   ’
1779                    psEmailNoError as string, _         :   ’
1780                    psMoxieOnError as string)
1781            GLOBAL glFinalResult AS LONG
1782            LOCAL llLen, llFlag, llC1, llC2, llC3, llC4, llC5, llC6 AS LONG
1783            LOCAL lsPreProcCfgPath, lsPreProcData, lsPreProcCfgData, lsData, lsaData(), lstmp1, lstmp2 AS STRING
1784            LOCAL lsFolderPath, lsaFolderPath(), lsLookFor, lsLookForResult, lsObjectRootFolder, lsBackupRootFolder AS STRING
1785
1786            REDIM                                       lsaFolderPath(0), lsaData(0)
1787
1788            REM ’parse config file for object directories
1789                sFileGet                                psWorkFldr + $PreProcCfgName, lsPreProcCfgData, llLen
1790                IF llLen < 2 THEN
1791                    lsPreProcCfgData                =   ""
1792                    lsPreProcCfgData                +=  $CRLF + "# ------------------------------- PreProc.conf -------------------------"
1793                    lsPreProcCfgData                +=  $CRLF + "# Emails below are one email per line; as many lines as required"
1794                    lsPreProcCfgData                +=  $CRLF + "# no more comments below this line"
1795                    lsPreProcCfgData                +=  $CRLF + "# ----------------------------------------------------------------------"
1796                    lsPreProcCfgData                +=  $CRLF + ""
1797                    lsPreProcCfgData                +=  $CRLF + "[ObjectRootFolder]"
1798                    lsPreProcCfgData                +=  $CRLF + "U:\Work"
1799                    lsPreProcCfgData                +=  $CRLF + ""
1800                    lsPreProcCfgData                +=  $CRLF + "[BackupRootFolder]"
1801                    lsPreProcCfgData                +=  $CRLF + "C:\Backup3"
1802                    lsPreProcCfgData                +=  $CRLF + ""
1803                    lsPreProcCfgData                +=  $CRLF + "[StatsRecvFolder]"
1804                    lsPreProcCfgData                +=  $CRLF + "C:\Work\Logs.WebNnow.cc\Recv"
1805                    lsPreProcCfgData                +=  $CRLF + ""
1806                    lsPreProcCfgData                +=  $CRLF + "[MoxViewFolder]"
1807                    lsPreProcCfgData                +=  $CRLF + "D:\Util\MoxView"
1808                    lsPreProcCfgData                +=  $CRLF + ""
1809                    lsPreProcCfgData                +=  $CRLF + "[MailServer]"
1810                    lsPreProcCfgData                +=  $CRLF + "mail.topdomain.top"
1811                    lsPreProcCfgData                +=  $CRLF + ""
1812                    lsPreProcCfgData                +=  $CRLF + "[MailServerPort]"
1813                    lsPreProcCfgData                +=  $CRLF + "65536"
1814                    lsPreProcCfgData                +=  $CRLF + ""
1815                    lsPreProcCfgData                +=  $CRLF + "[MailFrom]"
1816                    lsPreProcCfgData                +=  $CRLF + "server@topdomain.top"
1817                    lsPreProcCfgData                +=  $CRLF + ""
1818                    lsPreProcCfgData                +=  $CRLF + "[Force.Cnw.EmailErr]"
1819                    lsPreProcCfgData                +=  $CRLF + "Server.Admin@topdomain.top"
1820                    lsPreProcCfgData                +=  $CRLF + ""
1821                    lsPreProcCfgData                +=  $CRLF + "[EmailOnError]"
1822                    lsPreProcCfgData                +=  $CRLF + "Server.Admin@topdomain.top"
1823                    lsPreProcCfgData                +=  $CRLF + "The.Boss@topdomain.top"
1824                    lsPreProcCfgData                +=  $CRLF + ""
1825                    lsPreProcCfgData                +=  $CRLF + "[EmailNoError]"
1826                    lsPreProcCfgData                +=  $CRLF + "Server.Admin@topdomain.top"
1827                    lsPreProcCfgData                +=  $CRLF + ""
1828                    lsPreProcCfgData                +=  $CRLF + "[MoxieOnError]"
1829                    lsPreProcCfgData                +=  $CRLF + "Server.Admin@topdomain.top"
1830                    lsPreProcCfgData                +=  $CRLF + ""
1831                    sFilePut                            psWorkFldr + $PreProcCfgName, Trim$( lsPreProcCfgData, $CRLF)
1832                    lsObjectRootFolder              =   ""
1833                    glFinalResult                   =   2
1834                    gsFinalMessage                  +=  $CRLF + "A config file was not found; the following suggested configuration file was created. Please edit it before running again."
1835                    gsFinalMessage                  +=  $CRLF + lsPreProcCfgData
1836                ELSE
1837                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[ObjectRootFolder]"   : GOSUB ParseConfigParam   : lsObjectRootFolder = lsLookForResult
1838                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[BackupRootFolder]"   : GOSUB ParseConfigParam   : lsBackupRootFolder = lsLookForResult
1839                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[StatsRecvFolder]"    : GOSUB ParseConfigParam   : psStatsRecvFolder  = lsLookForResult
1840                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[MoxViewFolder]"      : GOSUB ParseConfigParam   : psMoxViewFolder    = lsLookForResult
1841                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[MailServer]"         : GOSUB ParseConfigParam   : psMailServer       = lsLookForResult
1842                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[MailServerPort]"     : GOSUB ParseConfigParam   : psMailServerPort   = lsLookForResult
1843                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[MailFrom]"           : GOSUB ParseConfigParam   : psMailFrom         = lsLookForResult
1844                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[Force.Cnw.EmailErr]" : GOSUB ParseConfigParam   : psForceCnwEmailErr = lsLookForResult
1845                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[EmailOnError]"       : GOSUB ParseConfigParam   : psEmailOnError     = lsLookForResult
1846                    lsLookForResult                 =   lsPreProcCfgData : lsLookFor = "[EmailNoError]"       : GOSUB ParseConfigParam   : psEmailNoError     = lsLookForResult
```

```
1847                lsLookForResult                         =   lsPreProcCfgData : lsLookFor = "[MoxieOnError]"      : GOSUB ParseConfigParam   : psMoxieOnError     = lsLookForResult
1848
1849                psBackupRootFolder                      =   lsBackupRootFolder
1850            END IF
1851
1852        REM 'Capture Configuration Info
1853            IF lsObjectRootFolder = "" THEN
1854                REDIM                                       psaFolderPath(0)
1855                EXIT                                        SUB
1856            ELSE
1857                ShellExec                                   "", "", "DIR /AD /B " + lsObjectRootFolder, lsPreProcData
1858                llC1                                    =   PARSECOUNT(lsPreProcData, $CRLF)
1859                REDIM                                       psaFolderPath(1 TO %MAXcfgFIELDS, 1 TO llC1)
1860                REDIM                                       lsaFolderPath(1 TO llC1)
1861                PARSE                                       lsPreProcData, lsaFolderPath(), $CRLF
1862                llC3                                    =   0
1863                FOR llC2 = 1 TO llC1
1864                    IF DIR$(lsObjectRootFolder + "\" + lsaFolderPath(llC2) + "\Moxie.exe") = "Moxie.exe" THEN
1865                        INCR                                llC3
1866                        psaFolderPath(%BackupRoot,   llC3) = lsBackupRootFolder
1867                        psaFolderPath(%FolderRoot,   llC3) = lsObjectRootFolder
1868                        psaFolderPath(%StatsRecv,    llC3) = psStatsRecvFolder
1869                        psaFolderPath(%FolderObject, llC3) = lsaFolderPath(llC2)
1870                        lsFolderPath                    =   psaFolderPath(%FolderRoot, llC3) + "\" + psaFolderPath(%FolderObject, llC3)
1871                        IF LEN( DIR$(lsFolderPath + "\Moxie.cfg")) THEN
1872                            sFileGet                        lsFolderPath + "\Moxie.cfg", psaFolderPath(%ConfigData, llC3), llLen
1873                        END IF
1874                        IF LEN( psaFolderPath(%ConfigData, llC3)) THEN
1875                            psaFolderPath(%IP               , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgIP             ), $CRLF))
1876                            psaFolderPath(%MaxConnections   , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgMaxConnections  ), $CRLF))
1877                            psaFolderPath(%HttpPort         , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgHttpPort       ), $CRLF))
1878                            psaFolderPath(%HttpsPort        , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgHttpsPort      ), $CRLF))
1879                            psaFolderPath(%CertFile         , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgCertFile        ), $CRLF))
1880                            psaFolderPath(%RelayHttpsHost   , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgRelayHttpsHost  ), $CRLF))
1881                            psaFolderPath(%Encoding         , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgEncoding        ), $CRLF))
1882                            psaFolderPath(%RAMRecvSize      , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgRAMRecvSize     ), $CRLF))
1883                            psaFolderPath(%DLLs             , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgDLLs            ), $CRLF))
1884                            psaFolderPath(%LocalBKPath      , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgLocalBKPath     ), $CRLF))
1885                            psaFolderPath(%SendBKName       , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgSendBKName      ), $CRLF))
1886                            psaFolderPath(%SendBKPort       , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgSendBKPort      ), $CRLF))
1887                            psaFolderPath(%SendBKPWHash     , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgSendBKPWHash    ), $CRLF))
1888                            psaFolderPath(%RecvBKPort       , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgRecvBKPort      ), $CRLF))
1889                            psaFolderPath(%RecvBKPWHash     , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgRecvBKPWHash    ), $CRLF))
1890                            psaFolderPath(%ServiceName      , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgServiceName     ), $CRLF))
1891                            psaFolderPath(%ServiceUser      , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgServiceUser     ), $CRLF))
1892                            psaFolderPath(%DiskFlushTimer   , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgDiskFlushTimer  ), $CRLF))
1893                            psaFolderPath(%AltIPInfo        , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgAltIPInfo       ), $CRLF))
1894                            psaFolderPath(%RelayNoAltIP     , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgRelayNoAltIP    ), $CRLF))
1895                            psaFolderPath(%ForcedCookies    , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgForcedCookies   ), $CRLF))
1896                            psaFolderPath(%Meta             , llC3) = TRIM$( EXTRACT$( REMAIN$( psaFolderPath(%ConfigData, llC3),$cfgMeta            ), $CRLF))
1897
1898                            psaFolderPath(%HttpPort, llC3)  = IIF$( VAL(psaFolderPath(%HttpPort,  llC3)) = 0, "", FORMAT$(VAL(psaFolderPath(%HttpPort,  llC3))))
1899                            psaFolderPath(%HttpsPort, llC3) = IIF$( VAL(psaFolderPath(%HttpsPort, llC3)) = 0, "", FORMAT$(VAL(psaFolderPath(%HttpsPort, llC3))))
1900
1901                            Replace                         $TAB  with $SEP_TAB  in psaFolderPath(%ConfigData, llC3)
1902                            Replace                         $CRLF with $SEP_CRLF in psaFolderPath(%ConfigData, llC3)
1903
1904                        END IF
1905                    END IF
1906                NEXT
1907                REDIM                                       PRESERVE psaFolderPath(1 TO %MAXcfgFIELDS, 1 TO llC3)
1908            END IF
1909
1910            EXIT                                            SUB
1911
1912        ParseConfigParam:
1913                lsData                                  =   REMAIN$( lsLookForResult, lsLookFor)
1914                lsData                                  =   EXTRACT$( lsData, "[")
1915                llC4                                    =   PARSECOUNT(lsData, $CRLF)
1916                REDIM                                       lsaData(1 TO llC4)
1917                PARSE                                       lsData, lsaData(), $CRLF
1918                llC6                                    =   0
1919                FOR llC5 = 1 to llC4
1920                    llFlag                              =   %True
1921                    If Len( Trim$(lsaData(llC5), ANY $WHITESPACE )) = 0 THEN llFlag = %False
1922                    If left$( Trim$(lsaData(llC5), ANY $WHITESPACE ), 1) = "#" then llFlag = %False
1923                    IF llFlag = %False THEN iterate for
```

```
                    INCR                                llC6
                    lsaData(llC6)                   =  Trim$(lsaData(llC5), ANY $WHITESPACE )
                Next
                REDIM Preserve lsaData(1 to llC6)
                If llC6 then
                    lsLookForResult                 =   join$( lsaData(), $CRLF)
                else
                    lsLookForResult                 =   ""
                End If
            RETURN
    END SUB

    sub MetaCapture( psaFolderPath() as string, psaMeta() as string)
        'make the Meta information more suitable for lookup
        LOCAL lsData, lsaData(), lsaTmp(), lsaTmp2() as string
        LOCAL llC1, llC2, llC3, llC4, llC5 AS LONG

        llC2 = UBOUND( psaFolderPath(2))

        'redim psaMeta() to same size as psaFolderPath() and process
        REDIM psaMeta(1 to llC2)
        REDIM lsaData (1 to llC2)
        For llC1 = 1 to llC2
            lsaData(llC1)                           =   psaFolderPath(%Meta, llC1)
        Next
        For llC1 = 1 to llC2
            lsData                                  =   lsaData(llC1)
            llC3                                    =   parsecount( lsData, ";")
            Redim                                       lsaTmp( 1 to llC3)
            Parse                                       lsData, lsaTmp(), ";"
            Reset                                       lsData
            If Len( lsaTmp(1)) THEN
                llC5                                =   0
                FOR llC4 = 1 to llC3
                    If INSTR( LCASE$( lsaTmp(llC4)), "nightly-") = 0 THEN iterate for
                    INCR                                llC5
                    lsaTmp(llC5)                    =   lsaTmp(llC4)
                Next
                llC3                                =   llC5
                REDIM Preserve lsaTmp(1 to llC3)
                REDIM lsaTmp2(1 to llC3, 2)
                FOR llC4 = 1 to llC3
                    'perform a single instance case-insensitive REPLACE "nightly-" WITH "" IN lsaTmp(llC4)
                    lsaTmp(llC4)                    =   ReplaceOnce( lsaTmp(llC4), "nightly-", "")
                    'and trim
                    lsaTmp2( llC4, 1)               =   Extract$( lsaTmp(llC4), ":")
                    lsaTmp2( llC4, 2)               =   Remain$( lsaTmp(llC4), ":")
                    lsaTmp2( llC4, 1)               =   trim$( lsaTmp2( llC4, 1), ANY $WHITESPACE)
                    lsaTmp2( llC4, 2)               =   trim$( lsaTmp2( llC4, 2), ANY $WHITESPACE)
                    lsaTmp(llC4)                    =   lsaTmp2( llC4, 1) & $SEP_SOH & lsaTmp2( llC4, 2)
                Next

                psaMeta(llC1)                       =   Join$( lsaTmp(), $SEP_NUL)
            End If
        Next
    end sub

    FUNCTION ReplaceOnce( psString as string, psSearchFor as string, psReplaceWith as string) as string
        ' PB Thread https://forum.powerbasic.com/forum/user-to-user-discussions/
        ' ... source-code/60412-a-time-efficient-case-insensitive-replace-function
        local llC1, llC2 as long
        llC2                =   LEN(psSearchFor)
        llC1                =   INSTR(LCASE$(psString), LCASE$(psSearchFor))
        If llC1 = 0 then
            FUNCTION = psString
        ELSE
            FUNCTION = LEFT$(psString, llC1 - 1) + psReplaceWith + MID$(psString,llC1 + llC2)
        End If
    End Function

    FUNCTION MetaSearch( psaMeta() AS STRING, plFolderPathNum AS LONG, psLookForThisMetaKey AS STRING) AS STRING
        LOCAL llC1, llC2 as long
        LOCAL lsResult, lsaTmp() as string

        llC2                =   parsecount( psaMeta( plFolderPathNum), CHR$(0))
        Redim                   lsaTmp( 1 to llC2)
        Parse                   psaMeta( plFolderPathNum), lsaTmp(), CHR$(0)
```

```
2001          For llC1 = 1 to llC2
2002              If LCASE$( Extract$( lsaTmp( llC1), CHR$(1))) = LCASE$(psLookForThisMetaKey) THEN
2003                  lsResult = Remain$( lsaTmp( llC1), CHR$(1))
2004              End If
2005          Next
2006          FUNCTION = lsResult
2007      END FUNCTION
2008
2009      sub sCaptureConfigurationSave( psWorkFldr as string, psaFolderPath() as string, psResult as string)
2010          LOCAL lsData as string
2011          LOCAL llC1, llC2 AS LONG
2012
2013          lsData                                          =   ""
2014          lsData                                          +=  "FolderRoot"          + $TAB
2015          lsData                                          +=  "BackupRoot"          + $TAB
2016          lsData                                          +=  "StatsRecv"           + $TAB
2017          lsData                                          +=  "FolderObject"        + $TAB
2018          lsData                                          +=  "ConfigData"          + $TAB
2019          lsData                                          +=  $cfgIP                + $TAB
2020          lsData                                          +=  $cfgMaxConnections    + $TAB
2021          lsData                                          +=  $cfgHttpPort          + $TAB
2022          lsData                                          +=  $cfgHttpsPort         + $TAB
2023          lsData                                          +=  $cfgCertFile          + $TAB
2024          lsData                                          +=  $cfgRelayHttpsHost    + $TAB
2025          lsData                                          +=  $cfgEncoding          + $TAB
2026          lsData                                          +=  $cfgRAMRecvSize       + $TAB
2027          lsData                                          +=  $cfgDLLs              + $TAB
2028          lsData                                          +=  $cfgLocalBKPath       + $TAB
2029          lsData                                          +=  $cfgSendBKName        + $TAB
2030          lsData                                          +=  $cfgSendBKPort        + $TAB
2031          lsData                                          +=  $cfgSendBKPWHash      + $TAB
2032          lsData                                          +=  $cfgRecvBKPort        + $TAB
2033          lsData                                          +=  $cfgRecvBKPWHash      + $TAB
2034          lsData                                          +=  $cfgServiceName       + $TAB
2035          lsData                                          +=  $cfgServiceUser       + $TAB
2036          lsData                                          +=  $cfgDiskFlushTimer    + $TAB
2037          lsData                                          +=  $cfgAltIPInfo         + $TAB
2038          lsData                                          +=  $cfgRelayNoAltIP      + $TAB
2039          lsData                                          +=  $cfgForcedCookies     + $TAB
2040          lsData                                          +=  $cfgMeta
2041          lsData                                          +=  $CRLF
2042
2043          replace                                             " =" with "" in lsData
2044
2045          FOR llC2 = 1 TO UBOUND( psaFolderPath(2))
2046              For llC1 = 1 to UBOUND( psaFolderPath(1))
2047                  lsData                                  +=  psaFolderPath(llC1, llC2)   + $TAB
2048              Next
2049              lsData                                      +=  $CRLF
2050          NEXT
2051          psResult                                        +=  $CRLF + Trim$(lsData, $CRLF)
2052          sFilePut                                            psWorkFldr + $ProgramName + ".conf.dump", Trim$(lsData, $CRLF)
2053          sFilePut                                            psWorkFldr + "Moxie\Recv\conf.dump", Trim$(lsData, $CRLF)
2054      End Sub
2055
2056      SUB CommitLog( psWorkFldr AS STRING, psResultLog AS STRING)
2057          sFilePut                                            psWorkFldr + $ProgramName + "\" + $ProgramName + "." + StampItDateHM + ".LogFile.log", TRIM$( psResultLog, $CRLF)
2058      END SUB
2059
2060      SUB sFileGet( psCompletePath AS STRING, psData AS STRING, plLOF AS LONG)
2061          LOCAL llFileNum AS LONG
2062          llFileNum                                       =   FREEFILE
2063          psData                                          =   ""
2064          OPEN                                                psCompletePath FOR BINARY LOCK SHARED AS llFileNum
2065              plLOF                                       =   LOF(llFileNum)
2066              GET$                                            llFileNum, LOF(llFileNum), psData
2067          CLOSE                                               llFileNum
2068      END SUB
2069
2070      SUB sFilePut( psCompletePath AS STRING, psData AS STRING)
2071          LOCAL llFileNum AS LONG
2072          llFileNum                                       =   FREEFILE
2073          OPEN                                                psCompletePath FOR BINARY LOCK SHARED AS llFileNum
2074              PUT$                                            llFileNum, psData : SETEOF(llFileNum)
2075          CLOSE                                               llFileNum
2076      END SUB
2077
```

```
2078    SUB sFilePutSeq( psCompletePath AS STRING, plMode as long, psData AS STRING)
2079        LOCAL llFileNum AS LONG
2080        llFileNum                                   =   FREEFILE
2081        OPEN                                            psCompletePath FOR BINARY LOCK write AS llFileNum
2082            If %file_Replace = plMode THEN              seek llFileNum, 0
2083            If %file_Append  = plMode THEN              seek llFileNum, lof(llFileNum)
2084            PUT$                                        llFileNum, psData : SETEOF(llFileNum)
2085        CLOSE                                           llFileNum
2086    END SUB
2087
2088    SUB sEnsureFolder( psWorkFldr AS STRING, psCompletePath AS STRING, psShellCommandResult AS STRING)
2089        LOCAL lsCommand AS STRING
2090        lsCommand                                   =   "IF NOT EXIST " + $DQ + TRIM$(psCompletePath) + $DQ + " (MKDIR " + $DQ + TRIM$(psCompletePath) + $DQ + ")"
2091        ShellExec                                       psWorkFldr, "", lsCommand, psShellCommandResult
2092    END SUB
2093
2094    SUB StampItInit
2095        GLOBAL gsDate, gsTime AS STRING
2096        gsDate                                      =   DATE$
2097        gsTime                                      =   TIME$
2098    END SUB
2099
2100    FUNCTION StampItDate AS STRING
2101        GLOBAL gsDate, gsTime AS STRING
2102        FUNCTION                                    =   RIGHT$(gsDate, 4) + "-" + LEFT$(gsDate, 5)
2103    END FUNCTION
2104
2105    FUNCTION StampItNowLogFmt AS STRING
2106        local lsDate, lsTime AS STRING
2107        lsDate                                      =   DATE$
2108        lsTime                                      =   TIME$
2109        FUNCTION                                    =   RIGHT$(lsDate, 4) + "-" + LEFT$(lsDate, 5) + " " + lsTime
2110    END FUNCTION
2111
2112    FUNCTION StampItDateHM AS STRING
2113        GLOBAL gsDate, gsTime AS STRING
2114        FUNCTION                                    =   RIGHT$(gsDate, 4) + "-" + LEFT$(gsDate, 5) + "-" + LEFT$(REMOVE$(gsTime, ":"),4)
2115    END FUNCTION
2116
2117    FUNCTION StampItCurrentSeconds AS STRING
2118        LOCAL lsDate, lsTime AS STRING
2119        lsDate                                      =   DATE$
2120        lsTime                                      =   TIME$
2121        FUNCTION                                    =   RIGHT$(lsDate, 4) + "-" + LEFT$(lsDate, 5) + "-" + REMOVE$(lsTime, ":")
2122    END FUNCTION
2123
2124    FUNCTION GetDow( psDate AS STRING) AS STRING
2125        'Clay Clear, Member PB Forums
2126        'posted September 29, 2005 03:49 AM
2127
2128        LOCAL x AS SYSTEMTIME, q AS QUAD
2129
2130        x.wyear                                     =   VAL(PARSE$(psDate, "-", 1))
2131        x.wmonth                                    =   VAL(PARSE$(psDate, "-", 2))
2132        x.wday                                      =   VAL(PARSE$(psDate, "-", 3))
2133
2134        systemtimetofiletime                            x, BYVAL VARPTR(q)
2135        filetimetosystemtime                            BYVAL VARPTR(q), x
2136
2137        FUNCTION                                    =   READ$((x.wdayofweek)+1)
2138
2139        DATA                                            "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
2140
2141    END FUNCTION
2142
2143    Function QuadToStatTime (pqDateTime as quad) as string
2144        local lqTmp as quad
2145        local llDay, llH, llM, llS as long
2146
2147        lqTmp                                       =   pqDateTime
2148        llDay                                       =   lqTmp \ 86400   :   lqTmp   =   lqTmp - (llDay * 86400)
2149        llH                                         =   lqTmp \ 3600    :   lqTmp   =   lqTmp - (llH * 3600)
2150        llM                                         =   lqTmp \ 60      :   lqTmp   =   lqTmp - (llM * 60)
2151        llS                                         =   lqTmp
2152
2153        function                                    =   format$( llDay) + " " + Format$( llH, "00") + ":" + Format$( llM, "00") + ":" + Format$( llS, "00")
2154    End Function
```

```
2155
2156    Function DateTimeToQuad (psDateTime as string) as QuaD
2157        'expected format
2158        ' 0        1         2
2159        ' 12345678901234567890
2160        ' 2014-02-29 15:34:25
2161        local llYmd, llH, llM, llS as long
2162        local lsYmd, lsHms as string
2163
2164        lsYmd                                       =   Extract$( psDateTime, " ")
2165        lsHms                                       =   Remain$(  psDateTime, " ")
2166
2167        llYmd                                       =   String2Julian( lsYmd)
2168        llH                                         =   VAL(LEFT$(    lsHms, 2))
2169        llM                                         =   VAL(MID$(     lsHms, 4, 2))
2170        llS                                         =   VAL(RIGHT$(   lsHms, 2))
2171
2172        function                                    =   llS + (llM * 60) + (llH * 3600) + (llYmd * 86400)
2173    End Function
2174
2175    FUNCTION Julian2String( plJulianDate AS LONG) AS STRING
2176        'Ref: Fliegel and van Flandern (1968)
2177        'From http://www.hermetic.ch/cal_stud/jdn.htm#comp
2178        LOCAL D, I, J, L, M, N, Y AS LONG
2179
2180        l                                           =   plJulianDate + 68569
2181        n                                           =   ( 4 * l ) \ 146097
2182        l                                           =   l - ( 146097 * n + 3 ) \ 4
2183        i                                           =   ( 4000 * ( l + 1 ) ) \ 1461001
2184        l                                           =   l - ( 1461 * i ) \ 4 + 31
2185        j                                           =   ( 80 * l ) \ 2447
2186        d                                           =   l - ( 2447 * j ) \ 80
2187        l                                           =   j \ 11
2188        m                                           =   j + 2 - ( 12 * l )
2189        y                                           =   100 * ( n - 49 ) + i + l
2190
2191        FUNCTION                                    =   FORMAT$(Y, "0000") + "-" + FORMAT$(M, "00") + "-" + FORMAT$(D, "00")
2192    END FUNCTION
2193
2194    FUNCTION String2Julian( psDate AS STRING) AS LONG
2195        'Ref: Fliegel and van Flandern (1968)
2196        'From http://www.hermetic.ch/cal_stud/jdn.htm#comp
2197        LOCAL Y, M, D AS LONG
2198
2199        IF IsDateValid(psDate) = %False THEN
2200            FUNCTION                                =   0
2201            EXIT                                        FUNCTION
2202        END IF
2203
2204        Y                                           =   VAL(LEFT$(  psDate, 4))
2205        M                                           =   VAL(MID$(   psDate, 6, 2))
2206        D                                           =   VAL(RIGHT$( psDate, 2))
2207        FUNCTION                                    =   ( 1461 * ( y + 4800 + ( m - 14 ) \ 12 ) ) \ 4 + _
2208                                                        ( 367 * ( m - 2 - 12 * ( ( m - 14 ) \ 12 ) ) ) \ 12 - _
2209                                                        ( 3 * ( ( y + 4900 + ( m - 14 ) \ 12 ) \ 100 ) ) \ 4 + _
2210                                                          d - 32075
2211    END FUNCTION
2212
2213    FUNCTION IsDateValid( psDate AS STRING) AS LONG
2214        'Adapted From http://www.hermetic.ch/cfunlib/dateval.htm
2215        LOCAL llYear, llMonth, llDay AS LONG
2216
2217        llYear                                      =   VAL(LEFT$(   psDate, 4))
2218        llMonth                                     =   VAL(MID$(    psDate, 6, 2))
2219        llDay                                       =   VAL(RIGHT$(  psDate, 2))
2220
2221        FUNCTION                                    =   %True
2222        IF llDay < 1 OR llYear < 1800 OR llYear > 2199 THEN
2223            FUNCTION                                =   %False
2224        ELSE
2225            SELECT CASE AS LONG llMonth
2226                CASE 1  :       IF llDay > 31 THEN FUNCTION = %False
2227                CASE 2  :   IF IsLeapYear( llYear) THEN
2228                                IF llDay > 29 THEN FUNCTION = %False
2229                            ELSE
2230                                IF llDay > 28 THEN FUNCTION = %False
2231                            END IF
```

```
2232                CASE 3  :       IF llDay > 31 THEN FUNCTION = %False
2233                CASE 4  :       IF llDay > 30 THEN FUNCTION = %False
2234                CASE 5  :       IF llDay > 31 THEN FUNCTION = %False
2235                CASE 6  :       IF llDay > 30 THEN FUNCTION = %False
2236                CASE 7  :       IF llDay > 31 THEN FUNCTION = %False
2237                CASE 8  :       IF llDay > 31 THEN FUNCTION = %False
2238                CASE 9  :       IF llDay > 30 THEN FUNCTION = %False
2239                CASE 10 :       IF llDay > 31 THEN FUNCTION = %False
2240                CASE 11 :       IF llDay > 30 THEN FUNCTION = %False
2241                CASE 12 :       IF llDay > 31 THEN FUNCTION = %False
2242                CASE ELSE
2243                    FUNCTION                            =   %False
2244            END SELECT
2245        END IF
2246    END FUNCTION
2247
2248    FUNCTION IsLeapYear( plYear AS LONG) AS LONG
2249        'Adapted From http://www.hermetic.ch/cfunlib/dateval.htm
2250        IF ( (plYear MOD 4) <> 0 ) THEN
2251            FUNCTION = %False
2252        ELSEIF ( (plYear MOD 400) = 0 ) THEN
2253            FUNCTION = %True
2254        ELSEIF ( (plYear MOD 100) = 0 ) THEN
2255            FUNCTION = %False
2256        ELSE
2257            FUNCTION = %True
2258        END IF
2259    END FUNCTION
2260
```