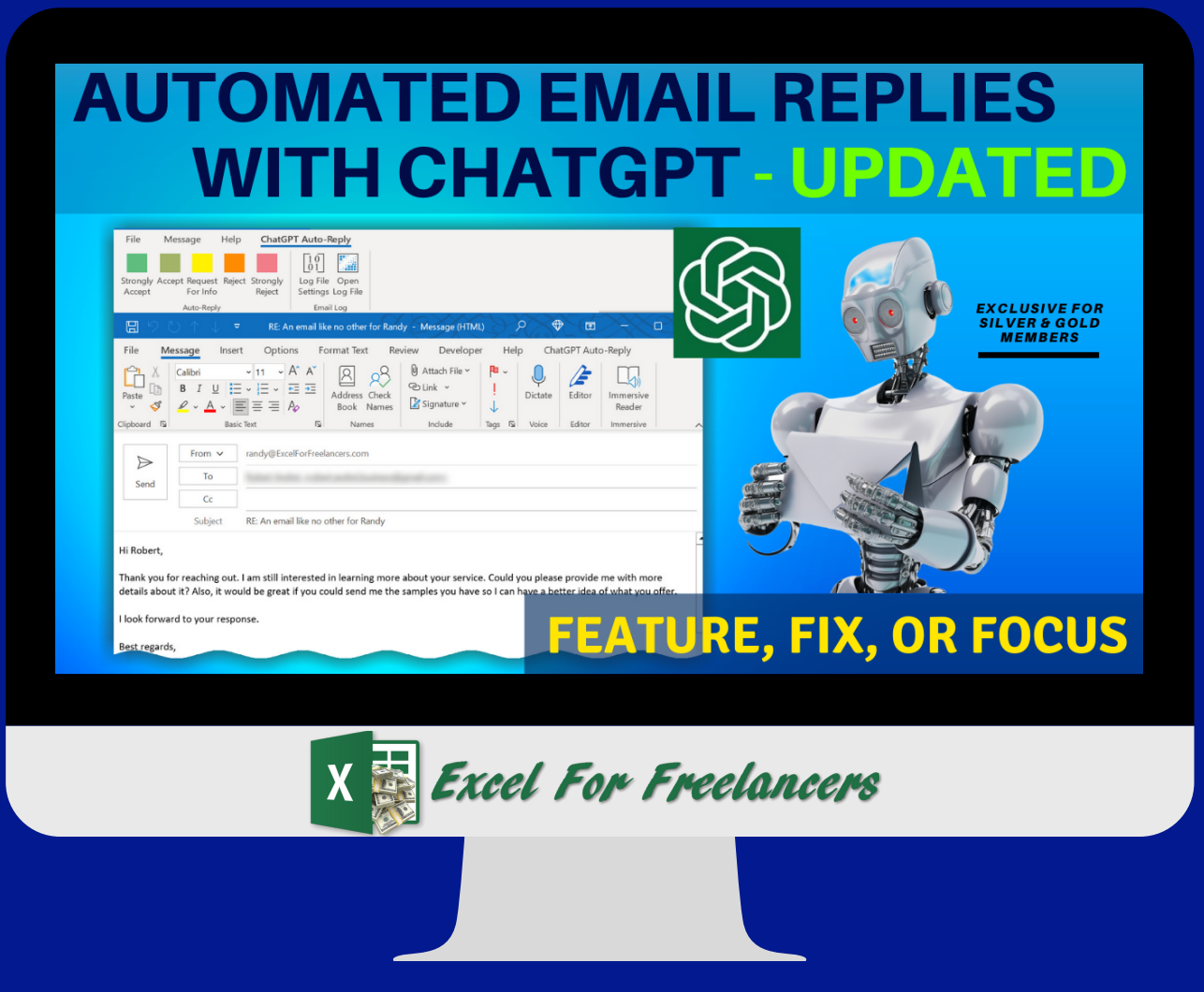
VBA SOURCE CODE BOOK



Excel For Freelancers

Maximize Your Productivity Using ChatGPT With 1-Click Email Replies + Excel Logging

DOWNLOAD APPLICATION

VIEW TRAINING

by: Randy Austin

ABOUT THE AUTHOR

A two-time Microsoft MVP & lifetime Excel enthusiast, Randy Austin founded Excel For Freelancers in 2017. Excel For Freelancers quickly became the most prominent resource Excel for developers to learn how to turn their passion for Excel into profits by building & selling their own excel-based applications for passive & recurring income.

With over 442,000 YouTube subscribers, 33,000,000 video views, 250+ comprehensive training videos, and a thriving 60,000 member Facebook community, Excel For Freelancers has positioned itself as the #1 Excel developers resource in the world.

Get free content, training, and downloads just by clicking any of the free resources below:

WEBSITE

YOUTUBE

FACEBOOK

TWITTER

DISCORD

INSTAGRAM

TELEGRAM

RUMBLE

Microsoft® MVP Most Valuable Professional

OUR COURSES & PRODUCTS

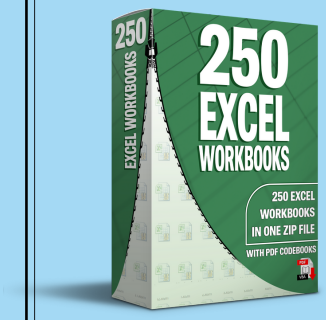
MENTORSHIP EXPRESS

This comprehensive program will take you through a 12-phase process that will turn your enthusiasm for Excel into passive income.

Click here to learn more

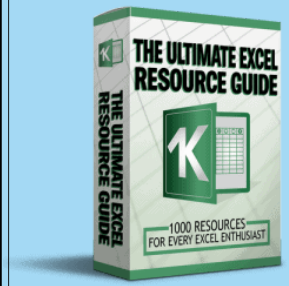
16 hour masterclass that will teach you the tips, tricks and techniques on how to create a dynamic single-click dashboard, and a ton more

Click here to learn more



Incredible Package of 250 of my BEST Applications now with PDF VBA Codebooks packed into a SINGLE ZIP File which also includes the "250 Workbook Library".

Click here to learn more



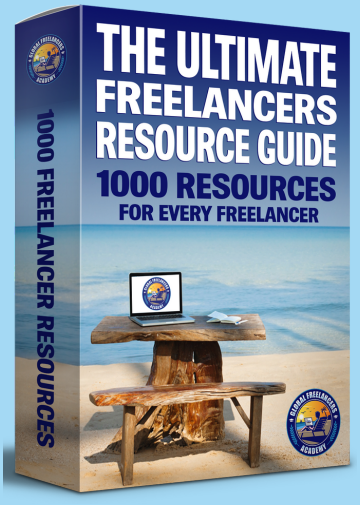
With 1000 live links, continuously updating content, sort-able and filterable items, you will always have exactly what you need, when you need it.

Click here to learn more

This incredible 13-hour freelancer masterclass will teach anyone how to be a successful freelancer with my proven 9-Phase 'Financial Freedom Roadmap' and includes 30+ downloads and exercises along with a ton of resources for anyone to reach their goals through freelancing.

Click here to learn more

-



I've put together all the Freelancing essentials, from freelance tools to freelance templates, in this Ultimate Freelancer's Resource Guide for freelancers at any stage in their career. With 1,000 Live Links, and a single click-to-update application, you will always have the most current and up-to-date information at your fingertips.

Click here to learn more

-

Table of Contents

```
Option Explicit
''''UPDATES
''''Same Language Response - Anni Birgit Sommer
''''Custom Response - Randy
''''No Selected Email Fix - Randy


Dim APIKey As String, LangReply As Boolean
Sub CheckForAPIKey()
   'Check For Existing API Key on Mac & Windows PC
   #If Mac Then 'Mac Computers
   Dim Script As String, Domain As String, Key As String
   Domain = "com.ChatGPT.OutlookKey"  'Set the name of the preferences domain
   APIKey = "defaults read "  & Domain & " "  & Key ' Set the name of the preferences key
   LangReply = "defaults read "  & Domain & " "  & LangReply 'Same Language Reply Option
   ' Set the preferences value to the API key
   'MacScript "defaults write " & Domain & " " & Key & " " & APIKey
   #Else 'Windows/PC
   APIKey = GetSetting("ChatGPT" , "Outlook" , "APIKey" ) 'Get API Key from registry (If Any)
   LangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" ) 'Same Language Reply Option
   #End If
   If APIKey = ""  Then
      APIKey = InputBox("Please enter your OpenAI API Key"  & vbCrLf & "You can obtain your key from here: https://platform.openai.com/account/api-keys" , "OpenAI API Key" )
   End If
   If APIKey = ""  Then End
   #If Mac Then
   MacScript "defaults write "  & Domain & " "  & Key & " "  & APIKey 'Write Script to save to user defaults system
   #Else
   SaveSetting "ChatGPT" , "Outlook" , "APIKey" , APIKey 'Save To Register
   #End If
End Sub

Sub EmailReply_Accept()
   ReplyWithChatGPT ("Accept" )
End Sub

Sub EmailReply_CustomReply()
   Dim CustomReply As String
   CustomReply = InputBox("Please enter the custom reply text for this email" , "Custom Reply" )
   If CustomReply = ""  Then Exit Sub
   ReplyWithChatGPT (CustomReply)
End Sub

Sub EmailReply_Reject()
   ReplyWithChatGPT ("Reject" )
End Sub

Sub EmailReply_RequestForInfo()
   ReplyWithChatGPT ("Request for more information" )
End Sub

Sub EmailReply_StronglyAccept()
   ReplyWithChatGPT ("Strongly Accept" )
1
```

```
End Sub

Sub EmailReply_StronglyReject()
    ReplyWithChatGPT ("Strongly Reject" )
End Sub

Sub ReplyWithChatGPT(ReplyType As String)
    Dim selectedEmail As MailItem, ReplyEmail As MailItem
    Dim EmailBody As String, AIBody As String, URL As String, Response As String, CompletedText As String
    Dim Prompt As String, ReplyText As String, JsonString As String, EmailFrom As String, SaveOpt As String, LogPath As String
    Dim oLink As Object, oXMLHTTP As Object, jsonObject As Object, choices As Object
    Dim olApp As Object, olMail As Object, olInspector As Object, olDocument As Object
    Dim olRange As Object, olLink As Object, Json As Object, objExcel As Object, objWorkbook As Object 'Late Binding Objects
    Dim LogRow As Long
    CheckForAPIKey
    #If Mac Then 'Mac Computers
    LangReply = "defaults read "  & Domain & " "  & LangReply 'Same Language Reply Option
    #Else 'Windows/PC
    LangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" ) 'Same Language Reply Option
    #End If

    Set olApp = CreateObject("Outlook.Application" )
    On Error GoTo SelectEmail
    Set selectedEmail = ActiveExplorer.Selection(1) ' Get the selected email
    On Error GoTo 0
    'Check if there is an active inspector window
    If olApp.ActiveInspector Is Nothing Then
        MsgBox "No active inspector window found." , vbExclamation
        Exit Sub
    End If

    EmailBody = selectedEmail.Body    'Get the visible text of the email body
    EmailFrom = selectedEmail.SenderEmailAddress 'From Email Address
    Prompt = "Please create an email response that will "  & ReplyType & " the following email\nn"  & EmailBody
    If LangReply = True Then Prompt = Prompt & " Make sure the reply is in the same language as the original email"
    'Create a JSON object
    Set jsonObject = CreateObject("Scripting.Dictionary" )
    jsonObject.Add "model" , "text-davinci-003"
    jsonObject.Add "prompt" , Prompt
    jsonObject.Add "temperature" , 0.5
    jsonObject.Add "max_tokens" , 3500

    'Convert the JSON object to a string
    AIBody = JsonConverter.ConvertToJson(jsonObject)
    Debug.Print AIBody

    'CheckForAPIKey 'Run macro to check and get API Key
    Set oXMLHTTP = CreateObject("MSXML2.ServerXMLHTTP" )
    oXMLHTTP.Open "POST" , "https://api.openai.com/v1/completions" , False 'Set Open Posts For Text
    oXMLHTTP.setRequestHeader "Content-Type" , "application/json"  'Set Header Type
    oXMLHTTP.setRequestHeader "Authorization" , "Bearer "  & APIKey 'Set Header Authorization
    On Error GoTo TimeOut
```

```
106     oXMLHTTP.Send AIBody
107     On Error GoTo 0
108     Response = oXMLHTTP.responseText
109     Debug.Print Response
110     If InStr(Response, "maximum context length" ) > 0 Then
111       MsgBox "The original email or the response has reached the current maximum # of
          words for the API. Please try with a shorter email"
112       Exit Sub
113     End If
114     If InStr(Response, "You didn't provide an API key" ) > 0 Then
115       MsgBox "No API Key was added. You can find your API key at "  & vbCrLf &
          "https://platform.openai.com/account/api-keys."
116       CheckForAPIKey 'Run Macro to place API Key
117       Exit Sub
118     End If
119     If InStr(Response, "Incorrect API key provided" ) > 0 Then
120       MsgBox "Incorrect API key provided. You can find your API key at "  & vbCrLf &
          "https://platform.openai.com/account/api-keys."
121       APIKey = InputBox("Please enter your OpenAI API Key"  & vbCrLf & "You can obtain
          your key from here: https://platform.openai.com/account/api-keys" , "OpenAI API
          Key" )
122       If APIKey = ""  Then End
123       #If Mac Then
124       MacScript "defaults write "  & Domain & " "  & Key & " "  & APIKey 'Write Script
          to save to user defaults system
125       #Else
126       SaveSetting "ChatGPT" , "Outlook" , "APIKey" , APIKey 'Save To Register
127       #End If
128       Exit Sub
129     End If
130     ReplyText = Right(Response, Len(Response) - InStrRev(Response, "[{"  & Chr(34) &
        "text"  & Chr(34) & ":"  & Chr(34)) - 9)
131     ReplyText = Left(ReplyText, InStrRev(ReplyText, Chr(34) & ","  & Chr(34) & "index" )
         - 1)
132     ReplyText = Replace(Replace(Replace(ReplyText, "\n" , Chr(10)), "\"  & Chr(34), Chr(
        34)), "\r" , vbNewLine)
133     If Left(ReplyText, 2) = Chr(10) & Chr(10) Then ReplyText = Right(ReplyText, Len(
        ReplyText) - 2)
134
135     Set ReplyEmail = selectedEmail.Reply     ' Create a reply email
136
137     ' Populate the reply email with ChatGPT response
138     ReplyEmail.HTMLBody = "<p>"  & Replace(ReplyText, vbCrLf, "</p><p>" ) & "</p>"
139     ReplyEmail.Display 'Display the reply email to the user (use "Send" to send without
        Displaying first)
140
141     'Check for Email Log Save
142     #If Mac Then 'Mac Computers
143     SaveOpt = "defaults read"  & "com.ChatGPT.SaveOption "  & SaveOption 'Extract
        Option (True/False)
144     LogPath = "defaults read "  & "com.ChatGPT.LogPath "  & LogFilePath 'Extract Log
        File Path
145     #Else 'Windows PC
146     SaveOpt = GetSetting("ChatGPT" , "Outlook" , "SaveOption" ) 'Extract Option
        (True/False)
147     LogPath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" )  'Extract Log File
        Path
148     #End If
149     'For Other systems
150     On Error Resume Next
151     SaveOpt = GetSetting("ChatGPT" , "Outlook" , "SaveOption" ) 'Extract Option
```

```
1
    (True/Fal
    se)
152     LogPath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" )  'Extract Log File
        Path
153     On Error GoTo 0
154     ''''Save Email To Excel Log File
155     'Check for True Save To Log option & Correct Log Path
156     If LogPath = ""  Then Exit Sub
157     If SaveOpt = True And Dir(LogPath, vbDirectory) <> Empty Then
158       Set objExcel = CreateObject("Excel.Application" ) 'Create a new instance of
          Excel
159       objExcel.ScreenUpdating = False 'Disable screen updating to improve performance
160       Set objWorkbook = objExcel.Workbooks.Open(LogPath) 'Open the workbook
161       objExcel.ScreenUpdating = True 'Enable screen updating
162       'objExcel.Visible = True 'Make Excel visible (this is only needed if you want to
          see the updates in Action)
163       'objWorkbook.Activate 'Activate the workbook
164       With objWorkbook.Sheets("Sheet1" )
165         LogRow = objWorkbook.Sheets("Sheet1" ).Range("A99999" ).End(xlup).Row + 1
          'First available Row
166         .Range("A"  & LogRow).Value = Now 'Sent On Date & Time
167         .Range("B"  & LogRow).Value = EmailBody 'Received Email Body
168         .Range("C"  & LogRow).Value = EmailFrom 'Original Email Sent From (Or Reply
          Sent To)
169         .Range("D"  & LogRow).Value = ReplyType 'Reply/Response Type
170         .Range("E"  & LogRow).Value = ReplyEmail.Subject
171         .Range("F"  & LogRow).Value = ReplyText 'Unformatted reply text
172         .Range(LogRow & ":"  & LogRow).WrapText = False 'Keep row height from
          expanding
173       End With
174       objWorkbook.Close True
175     End If
176     Exit Sub
177     TimeOut:
178     MsgBox "The operation timed out waiting for the response to chatGPT. Please try
        again or use a shorter email"
179     Exit Sub
180     SelectEmail:
181     MsgBox "Please open the email again in which you want to reply"
182 End Sub
183
184
```

```
Option Explicit
Sub BrowseForFolder()
    'Used to browse for folder in which Excel Email Log is saved
    Dim LogFolder As Object, LogShell As Object, WkBk As Object
    Dim FolderPath As String, LogPath As String

    Set LogShell = CreateObject("Shell.Application" )
    Set LogFolder = LogShell.BrowseForFolder(0, "Select a folder for the email log:" , 0
    , 0)
    If Not LogFolder Is Nothing Then
        FolderPath = LogFolder.Self.Path 'Folder Path
        LogPath = FolderPath & "\EmailLogFile.xlsx"
        If Dir(LogPath, vbDirectory) = ""  Then 'Check If folder contains Email Log
        File, if not, create it
            Set WkBk = CreateObject("Excel.Application" )
            Set WkBk = WkBk.Workbooks.Add
            WkBk.SaveAs LogPath
            With WkBk.Sheets("Sheet1" )
                .Range("A1" ).Value = "Sent On"
                .Range("B1" ).Value = "Original Email Message"
                .Range("C1" ).Value = "Sent To"
                .Range("D1" ).Value = "Reply Type"
                .Range("E1" ).Value = "Reply Subject"
                .Range("F1" ).Value = "Reply Message"
            End With
            WkBk.Close SaveChanges:=True
            Set WkBk = Nothing
        End If
        SaveToExcelForm.LogFilePath.Value = LogPath 'Set Log Path in Field
    End If
End Sub
Sub LogFileSettings()
    With SaveToExcelForm
        'Check For Existing API Key on Mac & Windows PC
        #If Mac Then 'Mac Computers
        .SaveEmailsBox = "defaults read"  & "com.ChatGPT.SaveOption "  & SaveOption
        'Extract Option (True/False)
        .LogFilePath = "defaults read "  & "com.ChatGPT.LogPath "  & LogFilePath
        'Extract Log File Path
        .SameLangReply = "defaults read "  & "com.ChatGPT.LangReply "  & LangReply 'Same
        Language Reply
        #Else 'Windows PC
        .SaveEmailsBox = GetSetting("ChatGPT" , "Outlook" , "SaveOption" ) 'Extract
        Option (True/False)
        .LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" )  'Extract Log
        File Path
        .SameLangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" )  'True/False
        Language Reply
        #End If
        'Other systems
        On Error Resume Next
        .SaveEmailsBox = GetSetting("ChatGPT" , "Outlook" , "SaveOption" ) 'Extract
        Option (True/False)
        .LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" )  'Extract Log
        File Path
        .SameLangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" )  'True/False
        Language Reply
        On Error GoTo 0
        .Show
    End With
End Sub
```

```
51
52  Sub OpenLogFile()
53      Dim LogFolder As Object, LogShell As Object, objExcel As Object, objWorkbook As
        Object
54      Dim LogFilePath As String
55
56      #If Mac Then 'Mac Computers
57      LogFilePath = "defaults read "  & "com.ChdtGPT.LogPath "  & LogFilePath 'Extract
        Log File Path
58      #Else 'Windows PC
59      LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" )  'Extract Log File
        Path
60      #End If
61      On Error Resume Next
62      LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" )  'Extract Log File
        Path
63      On Error GoTo 0
64      If LogFilePath = ""  Then
65        MsgBox "Log file path does not exist. Please browse for a log file location in
          order to create it"
66        LogFileSettings 'Run macro to check log file settings
67      End If
68      If Dir(LogFilePath, vbDirectory) <> ""  Then
69        Set objExcel = CreateObject("Excel.Application" ) 'Create a new instance of
          Excel
70        objExcel.ScreenUpdating = False 'Disable screen updating to improve performance
71        Set objWorkbook = objExcel.Workbooks.Open(LogFilePath) 'Open the workbook
72        objExcel.ScreenUpdating = True 'Enable screen updating
73        objExcel.Visible = True 'Make Excel visible
74        objWorkbook.Activate 'Activate the workbook
75      Else
76        MsgBox "Log file path does not exist or is not correct "  & LogFilePath
77        LogFileSettings 'Run macro to check log file settings
78      End If
79  End Sub
80  Sub SaveLogFormSettings()
81      Dim LogPath As String, SaveOpt As Boolean, LangReply As Boolean
82      With SaveToExcelForm
83        LogPath = .LogFilePath.Value 'Log File Path
84        SaveOpt = .SaveEmailsBox.Value 'Save Email Box Option
85        If .SameLangReply = True Then LangReply = True Else LangReply = False 'Set
          Language Reply
86        If SaveOpt = True And LogPath = ""  Then
87          MsgBox "Please browse for a Log Path or unselect the 'Save Emails To Excel
            Log option"
88          Exit Sub
89        End If
90
91        'Save Option & Log File Path for Mac & PC
92        #If Mac Then 'Mac Computers
93        Dim SaveOptScript As String, SaveOpt As String, LogPathScript As String
94        SaveOpt = "com.ChatGPT.SaveOption"  'Set the name Save Option location
95        LogPathScript = "com.ChatGPT.LogPath"  'Set the name LogPath
96        MacScript "defaults write "  & LogPathScript & " "  & SaveOption & " "  & SaveOpt
            'Save Option (True/False)
97        MacScript "defaults write "  & LogPathScript & " "  & LogFilePath & LogPath
          'Save File Path Of Excel Email Log
98        MacScript "defaults write "  & LogPathScript & " "  & LangReply & LangReply
          'Same Language Reply
99        #Else 'Windows PC
100       SaveSetting "ChatGPT" , "Outlook" , "SaveOption" , SaveOpt 'Save Option
    1  2
```

```
1 2
        (True/False)
101     SaveSetting "ChatGPT" , "Outlook" , "LogFilePath" , LogPath 'Save File Path Of
        Excel Email Log
102     SaveSetting "ChatGPT" , "Outlook" , "LangReply" , LangReply 'Same Language Reply

103     #End If
104     'other systems
105     On Error Resume Next
106     SaveSetting "ChatGPT" , "Outlook" , "SaveOption" , SaveOpt 'Save Option
        (True/False)
107     SaveSetting "ChatGPT" , "Outlook" , "LogFilePath" , LogPath 'Save File Path Of
        Excel Email Log
108     SaveSetting "ChatGPT" , "Outlook" , "LangReply" , LangReply 'Same Language Reply

109     On Error GoTo 0
110     .Hide
111   End With
112 End Sub
```

```
''
' VBA-JSON v2.3.1
' (c) Tim Hall - https://github.com/VBA-tools/VBA-JSON
'
' JSON Converter for VBA
'
' Errors:
' 10001 - JSON parse error
'
' @class JsonConverter
' @author tim.hall.engr@gmail.com
' @license MIT (http://www.opensource.org/licenses/mit-license.php)
'' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ '
'
' Based originally on vba-json (with extensive changes)
' BSD license included below
'
' JSONLib, http://code.google.com/p/vba-json/
'
' Copyright (c) 2013, Ryo Yokoyama
' All rights reserved.
'
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'     * Redistributions of source code must retain the above copyright
'       notice, this list of conditions and the following disclaimer.
'     * Redistributions in binary form must reproduce the above copyright
'       notice, this list of conditions and the following disclaimer in the
'       documentation and/or other materials provided with the distribution.
'     * Neither the name of the <organization> nor the
'       names of its contributors may be used to endorse or promote products
'       derived from this software without specific prior written permission.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
' ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
' WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
' DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY
' DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
' (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
' LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
' ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
' (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
' SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ '
Option Explicit

' === VBA-UTC Headers
#If Mac Then

#If VBA7 Then

' 64-bit Mac (2016)
Private Declare PtrSafe Function utc_popen Lib "/usr/lib/libc.dylib"  Alias "popen"  _
() '    (ByVal utc_Command As String, ByVal utc_Mode As String) As LongPtr
Private Declare PtrSafe Function utc_pclose Lib "/usr/lib/libc.dylib"  Alias "pclose"
_
() '    (ByVal utc_File As LongPtr) As LongPtr
Private Declare PtrSafe Function utc_fread Lib "/usr/lib/libc.dylib"  Alias "fread"  _
() '    (ByVal utc_Buffer As String, ByVal utc_Size As LongPtr, ByVal utc_Number As
LongPtr, ByVal utc_File As LongPtr) As LongPtr
Private Declare PtrSafe Function utc_feof Lib "/usr/lib/libc.dylib"  Alias "feof"  _
```

```
60      () '    (ByVal utc_File As LongPtr) As LongPtr
61
62      #Else
63
64      ' 32-bit Mac
65      Private Declare Function utc_popen Lib "libc.dylib"  Alias "popen"  _
66          (ByVal utc_Command As String, ByVal utc_Mode As String) As Long
67      Private Declare Function utc_pclose Lib "libc.dylib"  Alias "pclose"  _
68          (ByVal utc_File As Long) As Long
69      Private Declare Function utc_fread Lib "libc.dylib"  Alias "fread"  _
70          (ByVal utc_Buffer As String, ByVal utc_Size As Long, ByVal utc_Number As Long,
        ByVal utc_File As Long) As Long
71      Private Declare Function utc_feof Lib "libc.dylib"  Alias "feof"  _
72          (ByVal utc_File As Long) As Long
73
74      #End If
75
76      #ElseIf VBA7 Then
77
78      ' http://msdn.microsoft.com/en-us/library/windows/desktop/ms724421.aspx
79      ' http://msdn.microsoft.com/en-us/library/windows/desktop/ms724949.aspx
80      ' http://msdn.microsoft.com/en-us/library/windows/desktop/ms725485.aspx
81      Private Declare PtrSafe Function utc_GetTimeZoneInformation Lib "kernel32"  Alias
        "GetTimeZoneInformation"  _
82      () '    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION) As Long
83      Private Declare PtrSafe Function utc_SystemTimeToTzSpecificLocalTime Lib "kernel32"
        Alias "SystemTimeToTzSpecificLocalTime"  _
84      () '    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpUniversalTime
        As utc_SYSTEMTIME, utc_lpLocalTime As utc_SYSTEMTIME) As Long
85      Private Declare PtrSafe Function utc_TzSpecificLocalTimeToSystemTime Lib "kernel32"
        Alias "TzSpecificLocalTimeToSystemTime"  _
86      () '    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpLocalTime As
        utc_SYSTEMTIME, utc_lpUniversalTime As utc_SYSTEMTIME) As Long
87
88      #Else
89
90      Private Declare Function utc_GetTimeZoneInformation Lib "kernel32"  Alias
        "GetTimeZoneInformation"  _
91          (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION) As Long
92      Private Declare Function utc_SystemTimeToTzSpecificLocalTime Lib "kernel32"  Alias
        "SystemTimeToTzSpecificLocalTime"  _
93          (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpUniversalTime As
        utc_SYSTEMTIME, utc_lpLocalTime As utc_SYSTEMTIME) As Long
94      Private Declare Function utc_TzSpecificLocalTimeToSystemTime Lib "kernel32"  Alias
        "TzSpecificLocalTimeToSystemTime"  _
95          (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpLocalTime As
        utc_SYSTEMTIME, utc_lpUniversalTime As utc_SYSTEMTIME) As Long
96
97      #End If
98
99      #If Mac Then
100
101     #If VBA7 Then
102     Private Type utc_ShellResult
103       utc_Output As String
104       utc_ExitCode As LongPtr
105     End Type
106
107     #Else
108
109     Private Type utc_ShellResult
```

```
1
110         utc_Output As String
111         utc_ExitCode As Long
112     End Type
113
114     #End If
115
116     #Else
117
118     Private Type utc_SYSTEMTIME
119         utc_wYear As Integer
120         utc_wMonth As Integer
121         utc_wDayOfWeek As Integer
122         utc_wDay As Integer
123         utc_wHour As Integer
124         utc_wMinute As Integer
125         utc_wSecond As Integer
126         utc_wMilliseconds As Integer
127     End Type
128
129     Private Type utc_TIME_ZONE_INFORMATION
130         utc_Bias As Long
131         utc_StandardName(0 To 31) As Integer
132         utc_StandardDate As utc_SYSTEMTIME
133         utc_StandardBias As Long
134         utc_DaylightName(0 To 31) As Integer
135         utc_DaylightDate As utc_SYSTEMTIME
136         utc_DaylightBias As Long
137     End Type
138
139     #End If
140     ' === End VBA-UTC
141
142     Private Type json_Options
143         ' VBA only stores 15 significant digits, so any numbers larger than that are
            truncated
144         ' This can lead to issues when BIGINT's are used (e.g. for Ids or Credit Cards), as
            they will be invalid above 15 digits
145         ' See: http://support.microsoft.com/kb/269370
146         '
147         ' By default, VBA-JSON will use String for numbers longer than 15 characters that
            contain only digits
148         ' to override set `JsonConverter.JsonOptions.UseDoubleForLargeNumbers = True`
149         UseDoubleForLargeNumbers As Boolean
150
151         ' The JSON standard requires object keys to be quoted (" or '), use this option to
            allow unquoted keys
152         AllowUnquotedKeys As Boolean
153
154         ' The solidus (/) is not required to be escaped, use this option to escape them as
            \/ in ConvertToJson
155         EscapeSolidus As Boolean
156     End Type
157     Public JsonOptions As json_Options
158
159     ''
160     ' Convert local date to ISO 8601 string
161     '
162     ' @method ConvertToIso
163     ' @param {Date} utc_LocalDate
164     ' @return {Date} ISO 8601 string
165     ' @throws 10014 - ISO 8601 conversion error
```

```
166     ''
167     Public Function ConvertToIso(utc_LocalDate As Date) As String
168       On Error GoTo utc_ErrorHandling
169
170       ConvertToIso = VBA.Format$(ConvertToUtc(utc_LocalDate), "yyyy-mm-ddTHH:mm:ss.000Z" )
171
172       Exit Function
173
174       utc_ErrorHandling:
175       Err.Raise 10014, "UtcConverter.ConvertToIso" , "ISO 8601 conversion error: "  & Err.Number & " - "  & Err.Description
176     End Function
177
178     ''
179     ' Convert object (Dictionary/Collection/Array) to JSON
180     '
181     ' @method ConvertToJson
182     ' @param {Variant} JsonValue (Dictionary, Collection, or Array)
183     ' @param {Integer|String} Whitespace "Pretty" print json with given number of spaces per indentation (Integer) or given string
184     ' @return {String}
185     ''
186     Public Function ConvertToJson(ByVal JsonValue As Variant, Optional ByVal Whitespace As Variant, Optional ByVal json_CurrentIndentation As Long = 0) As String
187       Dim json_Buffer As String
188       Dim json_BufferPosition As Long
189       Dim json_BufferLength As Long
190       Dim json_Index As Long
191       Dim json_LBound As Long
192       Dim json_UBound As Long
193       Dim json_IsFirstItem As Boolean
194       Dim json_Index2D As Long
195       Dim json_LBound2D As Long
196       Dim json_UBound2D As Long
197       Dim json_IsFirstItem2D As Boolean
198       Dim json_Key As Variant
199       Dim json_Value As Variant
200       Dim json_DateStr As String
201       Dim json_Converted As String
202       Dim json_SkipItem As Boolean
203       Dim json_PrettyPrint As Boolean
204       Dim json_Indentation As String
205       Dim json_InnerIndentation As String
206
207       json_LBound = -1
208       json_UBound = -1
209       json_IsFirstItem = True
210       json_LBound2D = -1
211       json_UBound2D = -1
212       json_IsFirstItem2D = True
213       json_PrettyPrint = Not IsMissing(Whitespace)
214
215       Select Case VBA.VarType(JsonValue)
216       Case VBA.vbNull
217         ConvertToJson = "null"
218       Case VBA.vbDate
219         ' Date
220         json_DateStr = ConvertToIso(VBA.CDate(JsonValue))
221
222         ConvertToJson = "" ""  & json_DateStr & "" ""
1 2
```

```
    1  2
223     Case VBA.vbString
224       ' String (or large number encoded as string)
225       If Not JsonOptions.UseDoubleForLargeNumbers And json_StringIsLargeNumber(
          JsonValue) Then
226         ConvertToJson = JsonValue
227       Else
228         ConvertToJson = "" ""  & json_Encode(JsonValue) & "" ""
229       End If
230     Case VBA.vbBoolean
231       If JsonValue Then
232         ConvertToJson = "true"
233       Else
234         ConvertToJson = "false"
235       End If
236     Case VBA.vbArray To VBA.vbArray + VBA.vbByte
237       If json_PrettyPrint Then
238         If VBA.VarType(Whitespace) = VBA.vbString Then
239           json_Indentation = VBA.String$(json_CurrentIndentation + 1, Whitespace)
240           json_InnerIndentation = VBA.String$(json_CurrentIndentation + 2, Whitespace
              )
241         Else
242           json_Indentation = VBA.Space$((json_CurrentIndentation + 1) * Whitespace)
243           json_InnerIndentation = VBA.Space$((json_CurrentIndentation + 2) *
              Whitespace)
244         End If
245       End If
246
247       ' Array
248       json_BufferAppend json_Buffer, "[" , json_BufferPosition, json_BufferLength
249
250       On Error Resume Next
251
252       json_LBound = LBound(JsonValue, 1)
253       json_UBound = UBound(JsonValue, 1)
254       json_LBound2D = LBound(JsonValue, 2)
255       json_UBound2D = UBound(JsonValue, 2)
256
257       If json_LBound >= 0 And json_UBound >= 0 Then
258         For json_Index = json_LBound To json_UBound
259           If json_IsFirstItem Then
260             json_IsFirstItem = False
261           Else
262             ' Append comma to previous line
263             json_BufferAppend json_Buffer, "," , json_BufferPosition,
                json_BufferLength
264           End If
265
266           If json_LBound2D >= 0 And json_UBound2D >= 0 Then
267             ' 2D Array
268             If json_PrettyPrint Then
269               json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
                  json_BufferLength
270             End If
271             json_BufferAppend json_Buffer, json_Indentation & "[" ,
                json_BufferPosition, json_BufferLength
272
273             For json_Index2D = json_LBound2D To json_UBound2D
274               If json_IsFirstItem2D Then
275                 json_IsFirstItem2D = False
276               Else
277                 json_BufferAppend json_Buffer, "," , json_BufferPosition,
    1  2  3  4  5  6  7
```

```
1 2 3 4 5 6 7
          json_BufferLength
278       End If
279
280       json_Converted = ConvertToJson(JsonValue(json_Index, json_Index2D),
          Whitespace, json_CurrentIndentation + 2)
281
282       ' For Arrays/Collections, undefined (Empty/Nothing) is treated as
          null
283       If json_Converted = ""  Then
284         ' (nest to only check if converted = "")
285         If json_IsUndefined(JsonValue(json_Index, json_Index2D)) Then
286           json_Converted = "null"
287         End If
288       End If
289
290       If json_PrettyPrint Then
291         json_Converted = vbNewLine & json_InnerIndentation &
            json_Converted
292       End If
293
294       json_BufferAppend json_Buffer, json_Converted, json_BufferPosition,
          json_BufferLength
295     Next json_Index2D
296
297     If json_PrettyPrint Then
298       json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
          json_BufferLength
299     End If
300
301     json_BufferAppend json_Buffer, json_Indentation & "]" ,
        json_BufferPosition, json_BufferLength
302     json_IsFirstItem2D = True
303   Else
304     ' 1D Array
305     json_Converted = ConvertToJson(JsonValue(json_Index), Whitespace,
        json_CurrentIndentation + 1)
306
307     ' For Arrays/Collections, undefined (Empty/Nothing) is treated as null
308     If json_Converted = ""  Then
309       ' (nest to only check if converted = "")
310       If json_IsUndefined(JsonValue(json_Index)) Then
311         json_Converted = "null"
312       End If
313     End If
314
315     If json_PrettyPrint Then
316       json_Converted = vbNewLine & json_Indentation & json_Converted
317     End If
318
319     json_BufferAppend json_Buffer, json_Converted, json_BufferPosition,
        json_BufferLength
320   End If
321 Next json_Index
322 End If
323
324 On Error GoTo 0
325
326 If json_PrettyPrint Then
327   json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
      json_BufferLength
328
1 2 3
```

```
1 2 3
329             If VBA.VarType(Whitespace) = VBA.vbString Then
330               json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
331             Else
332               json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
333             End If
334           End If
335
336           json_BufferAppend json_Buffer, json_Indentation & "]" , json_BufferPosition,
              json_BufferLength
337
338           ConvertToJson = json_BufferToString(json_Buffer, json_BufferPosition)
339
340           ' Dictionary or Collection
341         Case VBA.vbObject
342           If json_PrettyPrint Then
343             If VBA.VarType(Whitespace) = VBA.vbString Then
344               json_Indentation = VBA.String$(json_CurrentIndentation + 1, Whitespace)
345             Else
346               json_Indentation = VBA.Space$((json_CurrentIndentation + 1) * Whitespace)
347             End If
348           End If
349
350           ' Dictionary
351           If VBA.TypeName(JsonValue) = "Dictionary"  Then
352             json_BufferAppend json_Buffer, "{" , json_BufferPosition, json_BufferLength
353             For Each json_Key In JsonValue.Keys
354               ' For Objects, undefined (Empty/Nothing) is not added to object
355               json_Converted = ConvertToJson(JsonValue(json_Key), Whitespace,
                  json_CurrentIndentation + 1)
356               If json_Converted = ""  Then
357                 json_SkipItem = json_IsUndefined(JsonValue(json_Key))
358               Else
359                 json_SkipItem = False
360               End If
361
362               If Not json_SkipItem Then
363                 If json_IsFirstItem Then
364                   json_IsFirstItem = False
365                 Else
366                   json_BufferAppend json_Buffer, "," , json_BufferPosition,
                      json_BufferLength
367                 End If
368
369                 If json_PrettyPrint Then
370                   json_Converted = vbNewLine & json_Indentation & "" ""  & json_Key &
                      "" ": "  & json_Converted
371                 Else
372                   json_Converted = "" ""  & json_Key & "" ":"  & json_Converted
373                 End If
374
375                 json_BufferAppend json_Buffer, json_Converted, json_BufferPosition,
                    json_BufferLength
376               End If
377             Next json_Key
378
379             If json_PrettyPrint Then
380               json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
                  json_BufferLength
381
382               If VBA.VarType(Whitespace) = VBA.vbString Then
383                 json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
1 2 3 4 5
```

```
1 2 3 4 5
384         Else
385             json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
386         End If
387       End If
388
389       json_BufferAppend json_Buffer, json_Indentation & "}" , json_BufferPosition,
          json_BufferLength
390
391       ' Collection
392     ElseIf VBA.TypeName(JsonValue) = "Collection"  Then
393       json_BufferAppend json_Buffer, "[" , json_BufferPosition, json_BufferLength
394       For Each json_Value In JsonValue
395         If json_IsFirstItem Then
396           json_IsFirstItem = False
397         Else
398           json_BufferAppend json_Buffer, "," , json_BufferPosition,
              json_BufferLength
399         End If
400
401         json_Converted = ConvertToJson(json_Value, Whitespace,
            json_CurrentIndentation + 1)
402
403         ' For Arrays/Collections, undefined (Empty/Nothing) is treated as null
404         If json_Converted = ""  Then
405           ' (nest to only check if converted = "")
406           If json_IsUndefined(json_Value) Then
407             json_Converted = "null"
408           End If
409         End If
410
411         If json_PrettyPrint Then
412           json_Converted = vbNewLine & json_Indentation & json_Converted
413         End If
414
415         json_BufferAppend json_Buffer, json_Converted, json_BufferPosition,
            json_BufferLength
416       Next json_Value
417
418       If json_PrettyPrint Then
419         json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
            json_BufferLength
420
421         If VBA.VarType(Whitespace) = VBA.vbString Then
422           json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
423         Else
424           json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
425         End If
426       End If
427
428       json_BufferAppend json_Buffer, json_Indentation & "]" , json_BufferPosition,
          json_BufferLength
429     End If
430
431     ConvertToJson = json_BufferToString(json_Buffer, json_BufferPosition)
432   Case VBA.vbInteger, VBA.vbLong, VBA.vbSingle, VBA.vbDouble, VBA.vbCurrency, VBA.
      vbDecimal
433     ' Number (use decimals for numbers)
434     ConvertToJson = VBA.Replace(JsonValue, "," , "." )
435   Case Else
436     ' vbEmpty, vbError, vbDataObject, vbByte, vbUserDefinedType
437     ' Use VBA's built-in to-string
1 2
```

```
438         On Error Resume Next
439         ConvertToJson = JsonValue
440         On Error GoTo 0
441       End Select
442     End Function
443
444     ''
445     ' Convert local date to UTC date
446     '
447     ' @method ConvertToUrc
448     ' @param {Date} utc_LocalDate
449     ' @return {Date} UTC date
450     ' @throws 10012 - UTC conversion error
451     ''
452     Public Function ConvertToUtc(utc_LocalDate As Date) As Date
453       On Error GoTo utc_ErrorHandling
454
455       #If Mac Then
456       ConvertToUtc = utc_ConvertDate(utc_LocalDate, utc_ConvertToUtc:=True)
457       #Else
458       Dim utc_TimeZoneInfo As utc_TIME_ZONE_INFORMATION
459       Dim utc_UtcDate As utc_SYSTEMTIME
460
461       utc_GetTimeZoneInformation utc_TimeZoneInfo
462       utc_TzSpecificLocalTimeToSystemTime utc_TimeZoneInfo, utc_DateToSystemTime(
          utc_LocalDate), utc_UtcDate
463
464       ConvertToUtc = utc_SystemTimeToDate(utc_UtcDate)
465       #End If
466
467       Exit Function
468
469       utc_ErrorHandling:
470       Err.Raise 10012, "UtcConverter.ConvertToUtc" , "UTC conversion error: "  & Err.
          Number & " - "  & Err.Description
471     End Function
472
473     Private Sub json_BufferAppend(ByRef json_Buffer As String, _
474                                   ByRef json_Append As Variant, _
475                                   ByRef json_BufferPosition As Long, _
476                                   ByRef json_BufferLength As Long)
477       ' VBA can be slow to append strings due to allocating a new string for each append
478       ' Instead of using the traditional append, allocate a large empty string and then
          copy string at append position
479       '
480       ' Example:
481       ' Buffer: "abc  "
482       ' Append: "def"
483       ' Buffer Position: 3
484       ' Buffer Length: 5
485       '
486       ' Buffer position + Append length > Buffer length -> Append chunk of blank space to
          buffer
487       ' Buffer: "abc       "
488       ' Buffer Length: 10
489       '
490       ' Put "def" into buffer at position 3 (0-based)
491       ' Buffer: "abcdef    "
492       '
493       ' Approach based on cStringBuilder from vbAccelerator
494       '
```

```
    http://www.vbaccelerator.com/home/VB/Code/Techniques/RunTime_Debug_Tracing/VB6_Tracer_Utility_zip_cStringBuilder_cls.asp
495 '
496 ' and clsStringAppend from Philip Swannell
497 ' https://github.com/VBA-tools/VBA-JSON/pull/82
498
499 Dim json_AppendLength As Long
500 Dim json_LengthPlusPosition As Long
501
502 json_AppendLength = VBA.Len(json_Append)
503 json_LengthPlusPosition = json_AppendLength + json_BufferPosition
504
505 If json_LengthPlusPosition > json_BufferLength Then
506   ' Appending would overflow buffer, add chunk
507   ' (double buffer length or append length, whichever is bigger)
508   Dim json_AddedLength As Long
509   json_AddedLength = IIf(json_AppendLength > json_BufferLength, json_AppendLength, json_BufferLength)
510
511   json_Buffer = json_Buffer & VBA.Space$(json_AddedLength)
512   json_BufferLength = json_BufferLength + json_AddedLength
513 End If
514
515 ' Note: Namespacing with VBA.Mid$ doesn't work properly here, throwing compile error:
516 ' Function call on left-hand side of assignment must return Variant or Object
517 Mid$(json_Buffer, json_BufferPosition + 1, json_AppendLength) = CStr(json_Append)
518 json_BufferPosition = json_BufferPosition + json_AppendLength
519 End Sub
520
521 Private Function json_BufferToString(ByRef json_Buffer As String, ByVal json_BufferPosition As Long) As String
522   If json_BufferPosition > 0 Then
523     json_BufferToString = VBA.Left$(json_Buffer, json_BufferPosition)
524   End If
525 End Function
526
527 Private Function json_Encode(ByVal json_Text As Variant) As String
528   ' Reference: http://www.ietf.org/rfc/rfc4627.txt
529   ' Escape: ", \, /, backspace, form feed, line feed, carriage return, tab
530   Dim json_Index As Long
531   Dim json_Char As String
532   Dim json_AscCode As Long
533   Dim json_Buffer As String
534   Dim json_BufferPosition As Long
535   Dim json_BufferLength As Long
536
537   For json_Index = 1 To VBA.Len(json_Text)
538     json_Char = VBA.Mid$(json_Text, json_Index, 1)
539     json_AscCode = VBA.AscW(json_Char)
540
541     ' When AscW returns a negative number, it returns the twos complement form of that number.
542     ' To convert the twos complement notation into normal binary notation, add 0xFFF to the return result.
543     ' https://support.microsoft.com/en-us/kb/272138
544     If json_AscCode < 0 Then
545       json_AscCode = json_AscCode + 65536
546     End If
547
548     ' From spec, ", \, and control characters must be escaped (solidus is optional)
1 2
```

```
549
550         Select Case json_AscCode
551           Case 34
552             ' " -> 34 -> \"
553             json_Char = "\" ""
554           Case 92
555             ' \ -> 92 -> \
556             json_Char = "\"
557           Case 47
558             ' / -> 47 -> \/ (optional)
559             If JsonOptions.EscapeSolidus Then
560               json_Char = "\/"
561             End If
562           Case 8
563             ' backspace -> 8 -> \b
564             json_Char = "\b"
565           Case 12
566             ' form feed -> 12 -> \f
567             json_Char = "\f"
568           Case 10
569             ' line feed -> 10 -> \n
570             json_Char = "\n"
571           Case 13
572             ' carriage return -> 13 -> \r
573             json_Char = "\r"
574           Case 9
575             ' tab -> 9 -> \t
576             json_Char = "\t"
577           Case 0 To 31, 127 To 65535
578             ' Non-ascii characters -> convert to 4-digit hex
579             json_Char = "\u"  & VBA.Right$("0000"  & VBA.Hex$(json_AscCode), 4)
580           End Select
581
582         json_BufferAppend json_Buffer, json_Char, json_BufferPosition, json_BufferLength
583       Next json_Index
584
585       json_Encode = json_BufferToString(json_Buffer, json_BufferPosition)
586     End Function
587
588     Private Function json_IsUndefined(ByVal json_Value As Variant) As Boolean
589       ' Empty / Nothing -> undefined
590       Select Case VBA.VarType(json_Value)
591         Case VBA.vbEmpty
592           json_IsUndefined = True
593         Case VBA.vbObject
594           Select Case VBA.TypeName(json_Value)
595             Case "Empty" , "Nothing"
596               json_IsUndefined = True
597           End Select
598       End Select
599     End Function
600
601     Private Function json_ParseArray(json_String As String, ByRef json_Index As Long) As
      Collection
602       Set json_ParseArray = New Collection
603
604       json_SkipSpaces json_String, json_Index
605       If VBA.Mid$(json_String, json_Index, 1) <> "["  Then
606         Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String, json_Index
          , "Expecting '['" )
607       Else
```

```
608         json_Index = json_Index + 1
609
610         Do
611           json_SkipSpaces json_String, json_Index
612           If VBA.Mid$(json_String, json_Index, 1) = "]"  Then
613             json_Index = json_Index + 1
614             Exit Function
615           ElseIf VBA.Mid$(json_String, json_Index, 1) = ","  Then
616             json_Index = json_Index + 1
617             json_SkipSpaces json_String, json_Index
618           End If
619
620           json_ParseArray.Add json_ParseValue(json_String, json_Index)
621         Loop
622       End If
623     End Function
624
625     Private Function json_ParseErrorMessage(json_String As String, ByRef json_Index As Long
      , ErrorMessage As String)
626       ' Provide detailed parse error message, including details of where and what
      occurred
627       '
628       ' Example:
629       ' Error parsing JSON:
630       ' {"abcde":True}
631       '           ^
632       ' Expecting 'STRING', 'NUMBER', null, true, false, '{', or '['
633
634       Dim json_StartIndex As Long
635       Dim json_StopIndex As Long
636
637       ' Include 10 characters before and after error (if possible)
638       json_StartIndex = json_Index - 10
639       json_StopIndex = json_Index + 10
640       If json_StartIndex <= 0 Then
641         json_StartIndex = 1
642       End If
643       If json_StopIndex > VBA.Len(json_String) Then
644         json_StopIndex = VBA.Len(json_String)
645       End If
646
647       json_ParseErrorMessage = "Error parsing JSON:"  & VBA.vbNewLine & _
648                                VBA.Mid$(json_String, json_StartIndex, json_StopIndex -
      json_StartIndex + 1) & VBA.vbNewLine & _
649                                VBA.Space$(json_Index - json_StartIndex) & "^"  & VBA.
      vbNewLine & _
650                                ErrorMessage
651     End Function
652
653     Private Function json_ParseKey(json_String As String, ByRef json_Index As Long) As
      String
654       ' Parse key with single or double quotes
655       If VBA.Mid$(json_String, json_Index, 1) = "" ""  Or VBA.Mid$(json_String, json_Index
      , 1) = "'"  Then
656         json_ParseKey = json_ParseString(json_String, json_Index)
657       ElseIf JsonOptions.AllowUnquotedKeys Then
658         Dim json_Char As String
659         Do While json_Index > 0 And json_Index <= Len(json_String)
660           json_Char = VBA.Mid$(json_String, json_Index, 1)
661           If (json_Char <> " " ) And (json_Char <> ":" ) Then
662             json_ParseKey = json_ParseKey & json_Char
```

```
663             json_Index = json_Index + 1
664           Else
665             Exit Do
666           End If
667         Loop
668       Else
669         Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String, json_Index
            , "Expecting '" "' or '''" )
670       End If
671
672       ' Check for colon and skip if present or throw if not present
673       json_SkipSpaces json_String, json_Index
674       If VBA.Mid$(json_String, json_Index, 1) <> ":"  Then
675         Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String, json_Index
            , "Expecting ':'" )
676       Else
677         json_Index = json_Index + 1
678       End If
679     End Function
680
681     Private Function json_ParseNumber(json_String As String, ByRef json_Index As Long) As
        Variant
682       Dim json_Char As String
683       Dim json_Value As String
684       Dim json_IsLargeNumber As Boolean
685
686       json_SkipSpaces json_String, json_Index
687
688       Do While json_Index > 0 And json_Index <= Len(json_String)
689         json_Char = VBA.Mid$(json_String, json_Index, 1)
690
691         If VBA.InStr("+-0123456789.eE" , json_Char) Then
692           ' Unlikely to have massive number, so use simple append rather than buffer
              here
693           json_Value = json_Value & json_Char
694           json_Index = json_Index + 1
695         Else
696           ' Excel only stores 15 significant digits, so any numbers larger than that
              are truncated
697           ' This can lead to issues when BIGINT's are used (e.g. for Ids or Credit
              Cards), as they will be invalid above 15 digits
698           ' See: http://support.microsoft.com/kb/269370
699           '
700           ' Fix: Parse -> String, Convert -> String longer than 15/16 characters
              containing only numbers and decimal points -> Number
701           ' (decimal doesn't factor into significant digit count, so if present check
              for 15 digits + decimal = 16)
702           json_IsLargeNumber = IIf(InStr(json_Value, "." ), Len(json_Value) >= 17, Len(
              json_Value) >= 16)
703           If Not JsonOptions.UseDoubleForLargeNumbers And json_IsLargeNumber Then
704             json_ParseNumber = json_Value
705           Else
706             ' VBA.Val does not use regional settings, so guard for comma is not needed
707             json_ParseNumber = VBA.Val(json_Value)
708           End If
709           Exit Function
710         End If
711       Loop
712     End Function
713
```

```
714     ' ============================================ '
715     ' Private Functions
716     ' ============================================ '
717
718     ''''' IF YOU GET A BUG ON THE LINE BELOW THEN RESET YOUR CODE IN THE MENU ABOVE GO TO
        TOOLS > REFERENCES THEN SCROLL DOWN AND  SELECT "MICROSOFT SCRIPTING RUNTIME"
719     Private Function json_ParseObject(json_String As String, ByRef json_Index As Long) As
        Dictionary
720       Dim json_Key As String
721       Dim json_NextChar As String
722
723       Set json_ParseObject = New Dictionary
724       json_SkipSpaces json_String, json_Index
725       If VBA.Mid$(json_String, json_Index, 1) <> "{"  Then
726         Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String, json_Index
          , "Expecting '{'" )
727       Else
728         json_Index = json_Index + 1
729
730         Do
731           json_SkipSpaces json_String, json_Index
732           If VBA.Mid$(json_String, json_Index, 1) = "}"  Then
733             json_Index = json_Index + 1
734             Exit Function
735           ElseIf VBA.Mid$(json_String, json_Index, 1) = ","  Then
736             json_Index = json_Index + 1
737             json_SkipSpaces json_String, json_Index
738           End If
739
740           json_Key = json_ParseKey(json_String, json_Index)
741           json_NextChar = json_Peek(json_String, json_Index)
742           If json_NextChar = "["  Or json_NextChar = "{"  Then
743             Set json_ParseObject.Item(json_Key) = json_ParseValue(json_String,
              json_Index)
744           Else
745             json_ParseObject.Item(json_Key) = json_ParseValue(json_String, json_Index)
746           End If
747         Loop
748       End If
749     End Function
750
751     Private Function json_ParseString(json_String As String, ByRef json_Index As Long) As
        String
752       Dim json_Quote As String
753       Dim json_Char As String
754       Dim json_Code As String
755       Dim json_Buffer As String
756       Dim json_BufferPosition As Long
757       Dim json_BufferLength As Long
758
759       json_SkipSpaces json_String, json_Index
760
761       ' Store opening quote to look for matching closing quote
762       json_Quote = VBA.Mid$(json_String, json_Index, 1)
763       json_Index = json_Index + 1
764
765       Do While json_Index > 0 And json_Index <= Len(json_String)
766         json_Char = VBA.Mid$(json_String, json_Index, 1)
767
768         Select Case json_Char
769         Case "\"
1 2 3
```

```
770             ' Escaped string, \, or \/
771             json_Index = json_Index + 1
772             json_Char = VBA.Mid$(json_String, json_Index, 1)
773
774             Select Case json_Char
775               Case "" "" , "\" , "/" , "'"
776                 json_BufferAppend json_Buffer, json_Char, json_BufferPosition,
                    json_BufferLength
777                 json_Index = json_Index + 1
778               Case "b"
779                 json_BufferAppend json_Buffer, vbBack, json_BufferPosition,
                    json_BufferLength
780                 json_Index = json_Index + 1
781               Case "f"
782                 json_BufferAppend json_Buffer, vbFormFeed, json_BufferPosition,
                    json_BufferLength
783                 json_Index = json_Index + 1
784               Case "n"
785                 json_BufferAppend json_Buffer, vbCrLf, json_BufferPosition,
                    json_BufferLength
786                 json_Index = json_Index + 1
787               Case "r"
788                 json_BufferAppend json_Buffer, vbCr, json_BufferPosition, json_BufferLength

789                 json_Index = json_Index + 1
790               Case "t"
791                 json_BufferAppend json_Buffer, vbTab, json_BufferPosition,
                    json_BufferLength
792                 json_Index = json_Index + 1
793               Case "u"
794                 ' Unicode character escape (e.g. \u00a9 = Copyright)
795                 json_Index = json_Index + 1
796                 json_Code = VBA.Mid$(json_String, json_Index, 4)
797                 json_BufferAppend json_Buffer, VBA.ChrW(VBA.Val("&h"  + json_Code)),
                    json_BufferPosition, json_BufferLength
798                 json_Index = json_Index + 4
799               End Select
800             Case json_Quote
801               json_ParseString = json_BufferToString(json_Buffer, json_BufferPosition)
802               json_Index = json_Index + 1
803               Exit Function
804             Case Else
805               json_BufferAppend json_Buffer, json_Char, json_BufferPosition,
                    json_BufferLength
806               json_Index = json_Index + 1
807             End Select
808           Loop
809         End Function
810
811         Private Function json_ParseValue(json_String As String, ByRef json_Index As Long) As
            Variant
812           json_SkipSpaces json_String, json_Index
813           Select Case VBA.Mid$(json_String, json_Index, 1)
814             Case "{"
815               Set json_ParseValue = json_ParseObject(json_String, json_Index)
816             Case "["
817               Set json_ParseValue = json_ParseArray(json_String, json_Index)
818             Case "" "" , "'"
819               json_ParseValue = json_ParseString(json_String, json_Index)
820             Case Else
821               If VBA.Mid$(json_String, json_Index, 4) = "true"  Then
```

```
1 2 3
822             json_ParseValue = True
823             json_Index = json_Index + 4
824           ElseIf VBA.Mid$(json_String, json_Index, 5) = "false"  Then
825             json_ParseValue = False
826             json_Index = json_Index + 5
827           ElseIf VBA.Mid$(json_String, json_Index, 4) = "null"  Then
828             json_ParseValue = Null
829             json_Index = json_Index + 4
830           ElseIf VBA.InStr("+-0123456789" , VBA.Mid$(json_String, json_Index, 1)) Then
831             json_ParseValue = json_ParseNumber(json_String, json_Index)
832           Else
833             Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String,
              json_Index, "Expecting 'STRING', 'NUMBER', null, true, false, '{', or '['" )
834           End If
835         End Select
836       End Function
837
838       Private Function json_Peek(json_String As String, ByVal json_Index As Long, Optional
          json_NumberOfCharacters As Long = 1) As String
839         ' "Peek" at the next number of characters without incrementing json_Index (ByVal
            instead of ByRef)
840         json_SkipSpaces json_String, json_Index
841         json_Peek = VBA.Mid$(json_String, json_Index, json_NumberOfCharacters)
842       End Function
843
844       Private Sub json_SkipSpaces(json_String As String, ByRef json_Index As Long)
845         ' Increment index to skip over spaces
846         Do While json_Index > 0 And json_Index <= VBA.Len(json_String) And VBA.Mid$(
            json_String, json_Index, 1) = " "
847           json_Index = json_Index + 1
848         Loop
849       End Sub
850
851       Private Function json_StringIsLargeNumber(json_String As Variant) As Boolean
852         ' Check if the given string is considered a "large number"
853         ' (See json_ParseNumber)
854
855         Dim json_Length As Long
856         Dim json_CharIndex As Long
857         json_Length = VBA.Len(json_String)
858
859         ' Length with be at least 16 characters and assume will be less than 100 characters
860         If json_Length >= 16 And json_Length <= 100 Then
861           Dim json_CharCode As String
862
863           json_StringIsLargeNumber = True
864
865           For json_CharIndex = 1 To json_Length
866             json_CharCode = VBA.Asc(VBA.Mid$(json_String, json_CharIndex, 1))
867             Select Case json_CharCode
868               ' Look for .|0-9|E|e
869             Case 46, 48 To 57, 69, 101
870               ' Continue through characters
871             Case Else
872               json_StringIsLargeNumber = False
873               Exit Function
874             End Select
875           Next json_CharIndex
876         End If
877       End Function
```

```
878
879     ''
880     ' Parse ISO 8601 date string to local date
881     '
882     ' @method ParseIso
883     ' @param {Date} utc_IsoString
884     ' @return {Date} Local date
885     ' @throws 10013 - ISO 8601 parsing error
886     ''
887     Public Function ParseIso(utc_IsoString As String) As Date
888       On Error GoTo utc_ErrorHandling
889
890       Dim utc_Parts() As String
891       Dim utc_DateParts() As String
892       Dim utc_TimeParts() As String
893       Dim utc_OffsetIndex As Long
894       Dim utc_HasOffset As Boolean
895       Dim utc_NegativeOffset As Boolean
896       Dim utc_OffsetParts() As String
897       Dim utc_Offset As Date
898
899       utc_Parts = VBA.Split(utc_IsoString, "T" )
900       utc_DateParts = VBA.Split(utc_Parts(0), "-" )
901       ParseIso = VBA.DateSerial(VBA.CInt(utc_DateParts(0)), VBA.CInt(utc_DateParts(1)),
          VBA.CInt(utc_DateParts(2)))
902
903       If UBound(utc_Parts) > 0 Then
904         If VBA.InStr(utc_Parts(1), "Z" ) Then
905           utc_TimeParts = VBA.Split(VBA.Replace(utc_Parts(1), "Z" , "" ), ":" )
906         Else
907           utc_OffsetIndex = VBA.InStr(1, utc_Parts(1), "+" )
908           If utc_OffsetIndex = 0 Then
909             utc_NegativeOffset = True
910             utc_OffsetIndex = VBA.InStr(1, utc_Parts(1), "-" )
911           End If
912
913           If utc_OffsetIndex > 0 Then
914             utc_HasOffset = True
915             utc_TimeParts = VBA.Split(VBA.Left$(utc_Parts(1), utc_OffsetIndex - 1),
                ":" )
916             utc_OffsetParts = VBA.Split(VBA.Right$(utc_Parts(1), Len(utc_Parts(1)) -
                utc_OffsetIndex), ":" )
917
918             Select Case UBound(utc_OffsetParts)
919               Case 0
920                 utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), 0, 0)
921               Case 1
922                 utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), VBA.CInt(
                  utc_OffsetParts(1)), 0)
923               Case 2
924                 ' VBA.Val does not use regional settings, use for seconds to avoid
                  decimal/comma issues
925                 utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), VBA.CInt(
                  utc_OffsetParts(1)), Int(VBA.Val(utc_OffsetParts(2))))
926             End Select
927
928             If utc_NegativeOffset Then: utc_Offset = -utc_Offset
929           Else
930             utc_TimeParts = VBA.Split(utc_Parts(1), ":" )
931           End If
932         End If
    1   2
```

```
933
934         Select Case UBound(utc_TimeParts)
935           Case 0
936             ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), 0, 0)
937           Case 1
938             ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), VBA.CInt(
                utc_TimeParts(1)), 0)
939           Case 2
940             ' VBA.Val does not use regional settings, use for seconds to avoid
                decimal/comma issues
941             ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), VBA.CInt(
                utc_TimeParts(1)), Int(VBA.Val(utc_TimeParts(2))))
942         End Select
943
944         ParseIso = ParseUtc(ParseIso)
945
946         If utc_HasOffset Then
947           ParseIso = ParseIso - utc_Offset
948         End If
949       End If
950
951       Exit Function
952
953       utc_ErrorHandling:
954       Err.Raise 10013, "UtcConverter.ParseIso" , "ISO 8601 parsing error for "  &
          utc_IsoString & ": "  & Err.Number & " - "  & Err.Description
955     End Function
956
957     ' ============================================ '
958     ' Public Methods
959     ' ============================================ '
960
961     ''
962     ' Convert JSON string to object (Dictionary/Collection)
963     '
964     ' @method ParseJson
965     ' @param {String} json_String
966     ' @return {Object} (Dictionary or Collection)
967     ' @throws 10001 - JSON parse error
968     ''
969     Public Function ParseJson(ByVal JsonString As String) As Object
970       Dim json_Index As Long
971       json_Index = 1
972
973       ' Remove vbCr, vbLf, and vbTab from json_String
974       JsonString = VBA.Replace(VBA.Replace(VBA.Replace(JsonString, VBA.vbCr, "" ), VBA.
          vbLf, "" ), VBA.vbTab, "" )
975
976       json_SkipSpaces JsonString, json_Index
977       Select Case VBA.Mid$(JsonString, json_Index, 1)
978         Case "{"
979           Set ParseJson = json_ParseObject(JsonString, json_Index)
980         Case "["
981           Set ParseJson = json_ParseArray(JsonString, json_Index)
982         Case Else
983           ' Error: Invalid JSON string
984           Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(JsonString, json_Index,
               "Expecting '{' or '['" )
985         End Select
986     End Function
987
```

```
    ''
    ' VBA-UTC v1.0.6
    ' (c) Tim Hall - https://github.com/VBA-tools/VBA-UtcConverter
    '
    ' UTC/ISO 8601 Converter for VBA
    '
    ' Errors:
    ' 10011 - UTC parsing error
    ' 10012 - UTC conversion error
    ' 10013 - ISO 8601 parsing error
    ' 10014 - ISO 8601 conversion error
    '
    ' @module UtcConverter
    ' @author tim.hall.engr@gmail.com
    ' @license MIT (http://www.opensource.org/licenses/mit-license.php)
    '' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ '

    ' (Declarations moved to top)

    ' ============================================= '
    ' Public Methods
    ' ============================================= '

    ''
    ' Parse UTC date to local date
    '
    ' @method ParseUtc
    ' @param {Date} UtcDate
    ' @return {Date} Local date
    ' @throws 10011 - UTC parsing error
    ''
    Public Function ParseUtc(utc_UtcDate As Date) As Date
        On Error GoTo utc_ErrorHandling

        #If Mac Then
        ParseUtc = utc_ConvertDate(utc_UtcDate)
        #Else
        Dim utc_TimeZoneInfo As utc_TIME_ZONE_INFORMATION
        Dim utc_LocalDate As utc_SYSTEMTIME

        utc_GetTimeZoneInformation utc_TimeZoneInfo
        utc_SystemTimeToTzSpecificLocalTime utc_TimeZoneInfo, utc_DateToSystemTime(
        utc_UtcDate), utc_LocalDate

        ParseUtc = utc_SystemTimeToDate(utc_LocalDate)
        #End If

        Exit Function

        utc_ErrorHandling:
        Err.Raise 10011, "UtcConverter.ParseUtc" , "UTC parsing error: "  & Err.Number & "
        - "  & Err.Description
    End Function

    ' ============================================= '
    ' Private Functions
    ' ============================================= '

    #If Mac Then

    Private Function utc_ConvertDate(utc_Value As Date, Optional utc_ConvertToUtc As
1
```

```
1
Boolean = False) As Date
1047    Dim utc_ShellCommand As String
1048    Dim utc_Result As utc_ShellResult
1049    Dim utc_Parts() As String
1050    Dim utc_DateParts() As String
1051    Dim utc_TimeParts() As String
1052
1053    If utc_ConvertToUtc Then
1054      utc_ShellCommand = "date -ur `date -jf '%Y-%m-%d %H:%M:%S' "  & _
1055          "'"  & VBA.Format$(utc_Value, "yyyy-mm-dd HH:mm:ss" ) & "' "  & _
1056          " +'%s'` +'%Y-%m-%d %H:%M:%S'"
1057    Else
1058      utc_ShellCommand = "date -jf '%Y-%m-%d %H:%M:%S %z' "  & _
1059          "'"  & VBA.Format$(utc_Value, "yyyy-mm-dd HH:mm:ss" ) & " +0000' "  & _
1060          "+'%Y-%m-%d %H:%M:%S'"
1061    End If
1062
1063    utc_Result = utc_ExecuteInShell(utc_ShellCommand)
1064
1065    If utc_Result.utc_Output = ""  Then
1066      Err.Raise 10015, "UtcConverter.utc_ConvertDate" , "'date' command failed"
1067    Else
1068      utc_Parts = Split(utc_Result.utc_Output, " " )
1069      utc_DateParts = Split(utc_Parts(0), "-" )
1070      utc_TimeParts = Split(utc_Parts(1), ":" )
1071
1072      utc_ConvertDate = DateSerial(utc_DateParts(0), utc_DateParts(1), utc_DateParts(2)
      ) + _
1073          TimeSerial(utc_TimeParts(0), utc_TimeParts(1), utc_TimeParts(2))
1074    End If
1075  End Function
1076
1077  #Else
1078
1079  Private Function utc_DateToSystemTime(utc_Value As Date) As utc_SYSTEMTIME
1080    utc_DateToSystemTime.utc_wYear = VBA.Year(utc_Value)
1081    utc_DateToSystemTime.utc_wMonth = VBA.Month(utc_Value)
1082    utc_DateToSystemTime.utc_wDay = VBA.Day(utc_Value)
1083    utc_DateToSystemTime.utc_wHour = VBA.Hour(utc_Value)
1084    utc_DateToSystemTime.utc_wMinute = VBA.Minute(utc_Value)
1085    utc_DateToSystemTime.utc_wSecond = VBA.Second(utc_Value)
1086    utc_DateToSystemTime.utc_wMilliseconds = 0
1087  End Function
1088
1089  Private Function utc_ExecuteInShell(utc_ShellCommand As String) As utc_ShellResult
1090    #If VBA7 Then
1091    Dim utc_File As LongPtr
1092    Dim utc_Read As LongPtr
1093    #Else
1094    Dim utc_File As Long
1095    Dim utc_Read As Long
1096    #End If
1097
1098    Dim utc_Chunk As String
1099
1100    On Error GoTo utc_ErrorHandling
1101    utc_File = utc_popen(utc_ShellCommand, "r" )
1102
1103    If utc_File = 0 Then: Exit Function
1104
1105    Do While utc_feof(utc_File) = 0
1 2
```

```
1106        utc_Chunk = VBA.Space$(50)
1107        utc_Read = CLng(utc_fread(utc_Chunk, 1, Len(utc_Chunk) - 1, utc_File))
1108        If utc_Read > 0 Then
1109          utc_Chunk = VBA.Left$(utc_Chunk, CLng(utc_Read))
1110          utc_ExecuteInShell.utc_Output = utc_ExecuteInShell.utc_Output & utc_Chunk
1111        End If
1112      Loop
1113
1114      utc_ErrorHandling:
1115      utc_ExecuteInShell.utc_ExitCode = CLng(utc_pclose(utc_File))
1116    End Function
1117
1118    Private Function utc_SystemTimeToDate(utc_Value As utc_SYSTEMTIME) As Date
1119      utc_SystemTimeToDate = DateSerial(utc_Value.utc_wYear, utc_Value.utc_wMonth,
          utc_Value.utc_wDay) + _
1120          TimeSerial(utc_Value.utc_wHour, utc_Value.utc_wMinute, utc_Value.utc_wSecond)
1121    End Function
1122
1123  #End If
```

Thank You!

This source code was created and made available to help you gain a better understanding of how VBA is used to create amazing Excel-based applications.

Thank you so much for your continued shares, likes and support. It really helps.

Excel For Freelancers