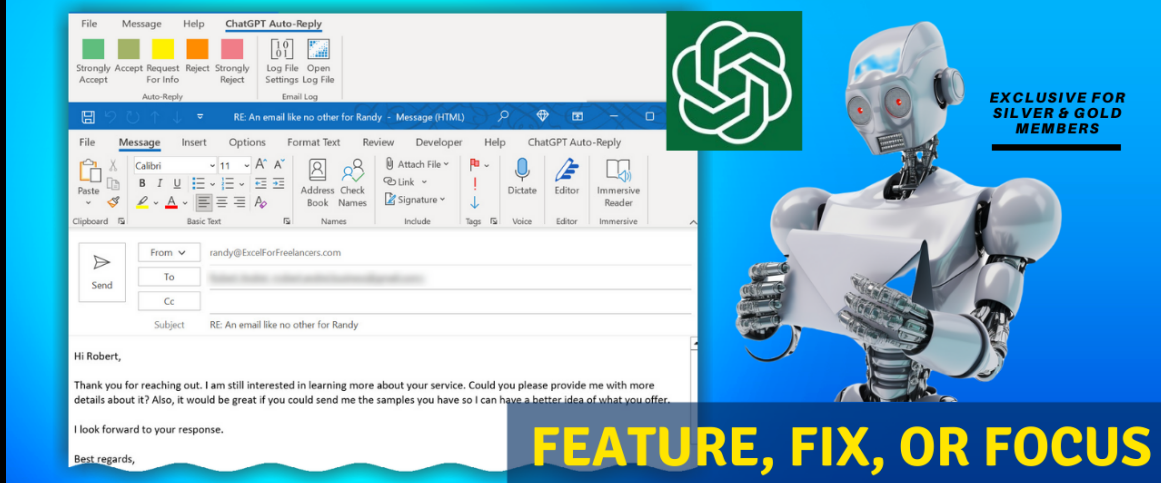


# VBA SOURCE CODE BOOK

## AUTOMATED EMAIL REPLIES WITH CHATGPT - UPDATED



**EXCLUSIVE FOR  
SILVER & GOLD  
MEMBERS**

**FEATURE, FIX, OR FOCUS**



*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**<sup>®</sup>  
Most Valuable  
Professional



# OUR COURSES & PRODUCTS



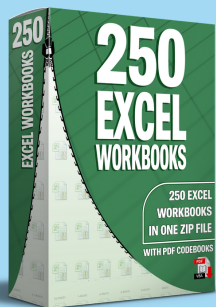
This comprehensive program will take you through a 12-phase process that will turn your enthusiasm for Excel into passive income.

[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](#)



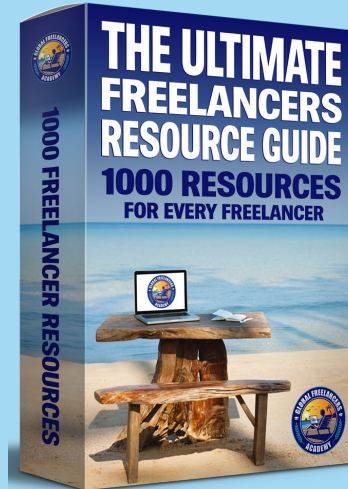
With 1000 live links, continuously updating content, sort-able and filterable items, you will always have exactly what you need, when you need it.

[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](#)

Projects.....	2
Project1 .....	2
Modules .....	2
AutoReply_Macros.....	2
(Declarations) .....	2
CheckForAPIKey [Sub ] .....	2
EmailReply_Accept [Sub ] .....	2
EmailReply_CustomReply [Sub ] .....	2
EmailReply_Reject [Sub ] .....	2
EmailReply_RequestForInfo [Sub ] .....	2
EmailReply_StronglyAccept [Sub ] .....	2
EmailReply_StronglyReject [Sub ] .....	3
ReplyWithChatGPT [Sub ] .....	3
EmailLog_Macros .....	6
(Declarations) .....	6
BrowseForFolder [Sub ] .....	6
LogFileSettings [Sub ] .....	6
OpenLogFile [Sub ] .....	7
SaveLogFormSettings [Sub ] .....	7
JsonConverter.....	9
(Declarations) .....	9
ConvertToIso [Function ] .....	11
ConvertToJson [Function ] .....	12
ConvertToUtc [Function ] .....	17
json_BufferAppend [Sub ] .....	17
json_BufferToString [Function ] .....	18
json_Encode [Function ] .....	18
json_IsUndefined [Function ] .....	19
json_ParseArray [Function ] .....	19
json_ParseErrorMessage [Function ] .....	20
json_ParseKey [Function ] .....	20
json_ParseNumber [Function ] .....	21
json_ParseObject [Function ] .....	21
json_ParseString [Function ] .....	22
json_ParseValue [Function ] .....	23
json_Peek [Function ] .....	24
json_SkipSpaces [Sub ] .....	24
json_StringIsLargeNumber [Function ] .....	24
ParseIso [Function ] .....	25
ParseJson [Function ] .....	26
ParseUtc [Function ] .....	26
utc_ConvertDate [Function ] .....	27
utc_DateToSystemTime [Function ] .....	28
utc_ExecutelnShell [Function ] .....	28
utc_SystemTimeToDate [Sub ] .....	29

```

1 Option Explicit
2 ''''UPDATES
3 ''''Same Language Response - Anni Birgit Sommer
4 ''''Custom Response - Randy
5 ''''No Selected Email Fix - Randy
6
7
8 Dim APIKey As String, LangReply As Boolean
9 Sub CheckForAPIKey()
10     'Check For Existing API Key on Mac & Windows PC
11     #If Mac Then 'Mac Computers
12     Dim Script As String, Domain As String, Key As String
13     Domain = "com.ChatGPT.OutlookKey" 'Set the name of the preferences domain
14     APIKey = "defaults read " & Domain & " " & Key ' Set the name of the preferences
15     key
16     LangReply = "defaults read " & Domain & " " & LangReply 'Same Language Reply
17     Option
18     ' Set the preferences value to the API key
19     'MacScript "defaults write " & Domain & " " & Key & " " & APIKey
20     #Else 'Windows/PC
21     APIKey = GetSetting("ChatGPT" , "Outlook" , "APIKey" ) 'Get API Key from registry
22     (If Any)
23     LangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" ) 'Same Language Reply
24     Option
25     #End If
26     If APIKey = "" Then
27         APIKey = InputBox("Please enter your OpenAI API Key" & vbCrLf & "You can obtain
28             your key from here: https://platform.openai.com/account/api-keys" , "OpenAI API
29             Key" )
30     End If
31     If APIKey = "" Then End
32     #If Mac Then
33     MacScript "defaults write " & Domain & " " & Key & " " & APIKey 'Write Script to
34     save to user defaults system
35     #Else
36     SaveSetting "ChatGPT" , "Outlook" , "APIKey" , APIKey 'Save To Register
37     #End If
38 End Sub
39
40 Sub EmailReply_Accept()
41     ReplyWithChatGPT ("Accept" )
42 End Sub
43
44 Sub EmailReply_CustomReply()
45     Dim CustomReply As String
46     CustomReply = InputBox("Please enter the custom reply text for this email" ,
47         "Custom Reply" )
48     If CustomReply = "" Then Exit Sub
49     ReplyWithChatGPT (CustomReply)
50 End Sub
51
52 Sub EmailReply_Reject()
53     ReplyWithChatGPT ("Reject" )
54 End Sub
55
56 Sub EmailReply_RequestForInfo()
57     ReplyWithChatGPT ("Request for more information" )
58 End Sub
59
60 Sub EmailReply_StronglyAccept()
61     ReplyWithChatGPT ("Strongly Accept" )
62 End Sub

```

1

```

1
54 End Sub
55
56 Sub EmailReply_StronglyReject()
57     ReplyWithChatGPT ("Strongly Reject" )
58 End Sub
59
60 Sub ReplyWithChatGPT(ReplyType As String)
61     Dim selectedEmail As MailItem, ReplyEmail As MailItem
62     Dim EmailBody As String, AIBody As String, URL As String, Response As String,
        CompletedText As String
63     Dim Prompt As String, ReplyText As String, jsonString As String, EmailFrom As String
        , SaveOpt As String, LogPath As String
64     Dim oLink As Object, oXMLHTTP As Object, jsonObject As Object, choices As Object
65     Dim olApp As Object, olMail As Object, olInspector As Object, olDocument As Object
66     Dim olRange As Object, olLink As Object, Json As Object, objExcel As Object,
        objWorkbook As Object 'Late Binding Objects
67     Dim LogRow As Long
68     CheckForAPIKey
69     #If Mac Then 'Mac Computers
70     LangReply = "defaults read " & Domain & " " & LangReply 'Same Language Reply
        Option
71     #Else 'Windows/PC
72     LangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" ) 'Same Language Reply
        Option
73     #End If
74
75     Set olApp = CreateObject("Outlook.Application" )
76     On Error GoTo SelectEmail
77     Set selectedEmail = ActiveExplorer.Selection(1) ' Get the selected email
78     On Error GoTo 0
79     'Check if there is an active inspector window
80     If olApp.ActiveInspector Is Nothing Then
81         MsgBox "No active inspector window found." , vbExclamation
82         Exit Sub
83     End If
84
85     EmailBody = selectedEmail.Body 'Get the visible text of the email body
86     EmailFrom = selectedEmail.SenderEmailAddress 'From Email Address
87     Prompt = "Please create an email response that will " & ReplyType & " the
        following email\n\n" & EmailBody
88     If LangReply = True Then Prompt = Prompt & " Make sure the reply is in the same
        language as the original email"
89     'Create a JSON object
90     Set jsonObject = CreateObject("Scripting.Dictionary" )
91     jsonObject.Add "model" , "text-davinci-003"
92     jsonObject.Add "prompt" , Prompt
93     jsonObject.Add "temperature" , 0.5
94     jsonObject.Add "max_tokens" , 3500
95
96     'Convert the JSON object to a string
97     AIBody = JsonConverter.ConvertToJson(jsonObject)
98     Debug.Print AIBody
99
100     'CheckForAPIKey 'Run macro to check and get API Key
101     Set oXMLHTTP = CreateObject("MSXML2.ServerXMLHTTP" )
102     oXMLHTTP.Open "POST" , "https://api.openai.com/v1/completions" , False 'Set Open
        Posts For Text
103     oXMLHTTP.setRequestHeader "Content-Type" , "application/json" 'Set Header Type
104     oXMLHTTP.setRequestHeader "Authorization" , "Bearer " & APIKey 'Set Header
        Authorization
105     On Error GoTo TimeOut

```

1

```

1
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" , vbCrLf)
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/False)
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

```

```

1 Option Explicit
2 Sub BrowseForFolder()
3     'Used to browse for folder in which Excel Email Log is saved
4     Dim LogFolder As Object, LogShell As Object, WkBk As Object
5     Dim FolderPath As String, LogPath As String
6
7     Set LogShell = CreateObject("Shell.Application" )
8     Set LogFolder = LogShell.BrowseForFolder(0, "Select a folder for the email log:" , 0
9
10    If Not LogFolder Is Nothing Then
11        FolderPath = LogFolder.Path 'Folder Path
12        LogPath = FolderPath & "\EmailLogFile.xlsx"
13        If Dir(LogPath, vbDirectory) = "" Then 'Check If folder contains Email Log
14            File, if not, create it
15            Set WkBk = CreateObject("Excel.Application" )
16            Set WkBk = WkBk.Workbooks.Add
17            WkBk.SaveAs LogPath
18            With WkBk.Sheets("Sheet1" )
19                .Range("A1" ).Value = "Sent On"
20                .Range("B1" ).Value = "Original Email Message"
21                .Range("C1" ).Value = "Sent To"
22                .Range("D1" ).Value = "Reply Type"
23                .Range("E1" ).Value = "Reply Subject"
24                .Range("F1" ).Value = "Reply Message"
25            End With
26            WkBk.Close SaveChanges:=True
27            Set WkBk = Nothing
28        End If
29        SaveToExcelForm.LogFilePath.Value = LogPath 'Set Log Path in Field
30    End If
31 End Sub
32
33 Sub LogFileSettings()
34     With SaveToExcelForm
35         'Check For Existing API Key on Mac & Windows PC
36         #If Mac Then 'Mac Computers
37             .SaveEmailsBox = "defaults read " & "com.ChatGPT.SaveOption " & SaveOption
38             'Extract Option (True/False)
39             .LogFilePath = "defaults read " & "com.ChatGPT.LogPath " & LogFilePath
40             'Extract Log File Path
41             .SameLangReply = "defaults read " & "com.ChatGPT.LangReply " & LangReply 'Same
42             Language Reply
43         #Else 'Windows PC
44             .SaveEmailsBox = GetSetting("ChatGPT" , "Outlook" , "SaveOption" ) 'Extract
45             Option (True/False)
46             .LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" ) 'Extract Log
47             File Path
48             .SameLangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" ) 'True/False
49             Language Reply
50         #End If
51         'Other systems
52         On Error Resume Next
53         .SaveEmailsBox = GetSetting("ChatGPT" , "Outlook" , "SaveOption" ) 'Extract
54         Option (True/False)
55         .LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" ) 'Extract Log
56         File Path
57         .SameLangReply = GetSetting("ChatGPT" , "Outlook" , "LangReply" ) 'True/False
58         Language Reply
59         On Error GoTo 0
60     End With
61 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
58     Log File Path
59     #Else 'Windows PC
60     LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" ) 'Extract Log File
61     Path
62     #End If
63     On Error Resume Next
64     LogFilePath = GetSetting("ChatGPT" , "Outlook" , "LogFilePath" ) 'Extract Log File
65     Path
66     On Error GoTo 0
67     If LogFilePath = "" Then
68         MsgBox "Log file path does not exist. Please browse for a log file location in
69         order to create it"
70         LogFileSettings 'Run macro to check log file settings
71     End If
72     If Dir(LogFilePath, vbDirectory) <> "" Then
73         Set objExcel = CreateObject("Excel.Application") 'Create a new instance of
74         Excel
75         objExcel.ScreenUpdating = False 'Disable screen updating to improve performance
76         Set objWorkbook = objExcel.Workbooks.Open(LogFilePath) 'Open the workbook
77         objExcel.ScreenUpdating = True 'Enable screen updating
78         objExcel.Visible = True 'Make Excel visible
79         objWorkbook.Activate 'Activate the workbook
80     Else
81         MsgBox "Log file path does not exist or is not correct " & LogFilePath
82         LogFileSettings 'Run macro to check log file settings
83     End If
84 End Sub
85
86 Sub SaveLogFormSettings()
87     Dim LogPath As String, SaveOpt As Boolean, LangReply As Boolean
88     With SaveToExcelForm
89         LogPath = .LogFilePath.Value 'Log File Path
90         SaveOpt = .SaveEmailsBox.Value 'Save Email Box Option
91         If .SameLangReply = True Then LangReply = True Else LangReply = False 'Set
92         Language Reply
93         If SaveOpt = True And LogPath = "" Then
94             MsgBox "Please browse for a Log Path or unselect the 'Save Emails To Excel
95             Log option"
96             Exit Sub
97         End If
98
99         'Save Option & Log File Path for Mac & PC
100        #If Mac Then 'Mac Computers
101        Dim SaveOptScript As String, SaveOpt As String, LogPathScript As String
102        SaveOpt = "com.ChatGPT.SaveOption" 'Set the name Save Option location
103        LogPathScript = "com.ChatGPT.LogPath" 'Set the name LogPath
104        MacScript "defaults write " & LogPathScript & " " & SaveOption & " " & SaveOpt
105        'Save Option (True/False)
106        MacScript "defaults write " & LogPathScript & " " & LogFilePath & LogPath
107        'Save File Path Of Excel Email Log
108        MacScript "defaults write " & LogPathScript & " " & LangReply & LangReply
109        'Same Language Reply
110        #Else 'Windows PC
111        SaveSetting "ChatGPT" , "Outlook" , "SaveOption" , SaveOpt 'Save Option

```

```
1 2
101 (True/False)
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
```

```

1  ''
2  ' VBA-JSON v2.3.1
3  ' (c) Tim Hall - https://github.com/VBA-tools/VBA-JSON
4  '
5  ' JSON Converter for VBA
6  '
7  ' Errors:
8  ' 10001 - JSON parse error
9  '
10 '@class JsonConverter
11 '@author tim.hall.engr@gmail.com
12 '@license MIT (http://www.opensource.org/licenses/mit-license.php)
13 ' ~~~~~
14 '
15 ' Based originally on vba-json (with extensive changes)
16 ' BSD license included below
17 '
18 ' JSONLib, http://code.google.com/p/vba-json/
19 '
20 ' Copyright (c) 2013, Ryo Yokoyama
21 ' All rights reserved.
22 '
23 ' Redistribution and use in source and binary forms, with or without
24 ' modification, are permitted provided that the following conditions are met:
25 ' * Redistributions of source code must retain the above copyright
26 '   notice, this list of conditions and the following disclaimer.
27 ' * Redistributions in binary form must reproduce the above copyright
28 '   notice, this list of conditions and the following disclaimer in the
29 '   documentation and/or other materials provided with the distribution.
30 ' * Neither the name of the <organization> nor the
31 '   names of its contributors may be used to endorse or promote products
32 '   derived from this software without specific prior written permission.
33 '
34 ' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
35 ' ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
36 ' WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
37 ' DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY
38 ' DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
39 ' (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
40 ' LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
41 ' ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
42 ' (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
43 ' SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
44 ' ~~~~~
45 Option Explicit
46
47 ' === VBA-UTC Headers
48 #If Mac Then
49
50 #If VBA7 Then
51
52 ' 64-bit Mac (2016)
53 Private Declare PtrSafe Function utc_popen Lib "/usr/lib/libc.dylib" Alias "popen" _
54 () ' (ByVal utc_Command As String, ByVal utc_Mode As String) As LongPtr
55 Private Declare PtrSafe Function utc_pclose Lib "/usr/lib/libc.dylib" Alias "pclose" _
56 () ' (ByVal utc_File As LongPtr) As LongPtr
57 Private Declare PtrSafe Function utc_fread Lib "/usr/lib/libc.dylib" Alias "fread" _
58 () ' (ByVal utc_Buffer As String, ByVal utc_Size As LongPtr, ByVal utc_Number As _
59 LongPtr, ByVal utc_File As LongPtr) As LongPtr
59 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,
71         ByVal utc_File As Long) As Long
72     Private Declare Function utc_feof Lib "libc.dylib" Alias "feof" _
73         (ByVal utc_File As Long) As Long
74
75     #End If
76
77     #ElseIf VBA7 Then
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
82         "GetTimeZoneInformation" _
83         (ByVal utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION) As Long
84     Private Declare PtrSafe Function utc_SystemTimeToTzSpecificLocalTime Lib "kernel32"
85         Alias "SystemTimeToTzSpecificLocalTime" _
86         (ByVal utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, ByVal utc_lpUniversalTime
87         As utc_SYSTEMTIME, ByVal utc_lpLocalTime As utc_SYSTEMTIME) As Long
88     Private Declare PtrSafe Function utc_TzSpecificLocalTimeToSystemTime Lib "kernel32"
89         Alias "TzSpecificLocalTimeToSystemTime" _
90         (ByVal utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, ByVal utc_lpLocalTime As
91         utc_SYSTEMTIME, ByVal utc_lpUniversalTime As utc_SYSTEMTIME) As Long
92
93     #Else
94     Private Declare Function utc_GetTimeZoneInformation Lib "kernel32" Alias
95         "GetTimeZoneInformation" _
96         (ByVal utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION) As Long
97     Private Declare Function utc_SystemTimeToTzSpecificLocalTime Lib "kernel32" Alias
98         "SystemTimeToTzSpecificLocalTime" _
99         (ByVal utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, ByVal utc_lpUniversalTime As
100         utc_SYSTEMTIME, ByVal utc_lpLocalTime As utc_SYSTEMTIME) As Long
101     Private Declare Function utc_TzSpecificLocalTimeToSystemTime Lib "kernel32" Alias
102         "TzSpecificLocalTimeToSystemTime" _
103         (ByVal utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, ByVal utc_lpLocalTime As
104         utc_SYSTEMTIME, ByVal utc_lpUniversalTime As utc_SYSTEMTIME) As Long
105
106     #End If
107
108     #If Mac Then
109     #If VBA7 Then
110     Private Type utc_ShellResult
111         utc_Output As String
112         utc_ExitCode As LongPtr
113     End Type
114
115     #Else
116     Private Type utc_ShellResult
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

1

```

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.
176      Number & " - " & Err.Description
177  End Function
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
184  ' per indentation (Integer) or given string
185  ' @return {String}
186  Public Function ConvertToJson(ByVal JsonValue As Variant, Optional ByVal Whitespace As
187  Variant, Optional ByVal json_CurrentIndentation As Long = 0) As String
188      Dim json_Buffer As String
189      Dim json_BufferPosition As Long
190      Dim json_BufferLength As Long
191      Dim json_Index As Long
192      Dim json_LBound As Long
193      Dim json_UBound As Long
194      Dim json_IsFirstItem As Boolean
195      Dim json_Index2D As Long
196      Dim json_LBound2D As Long
197      Dim json_UBound2D As Long
198      Dim json_IsFirstItem2D As Boolean
199      Dim json_Key As Variant
200      Dim json_Value As Variant
201      Dim json_DateStr As String
202      Dim json_Converted As String
203      Dim json_SkipItem As Boolean
204      Dim json_PrettyPrint As Boolean
205      Dim json_Indentation As String
206      Dim json_InnerIndentation As String
207
208      json_LBound = -1
209      json_UBound = -1
210      json_IsFirstItem = True
211      json_LBound2D = -1
212      json_UBound2D = -1
213      json_IsFirstItem2D = True
214      json_PrettyPrint = Not IsMissing(Whitespace)
215
216      Select Case VBA.VarType(JsonValue)
217      Case VBA.vbNull
218          ConvertToJson = "null"
219      Case VBA.vbDate
220          ' Date
221          json_DateStr = ConvertToIso(VBA.CDate(JsonValue))
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(
226         JsonValue) Then
227         ConvertToJson = JsonValue
228     Else
229         ConvertToJson = "" "" & json_Encode(JsonValue) & "" ""
230     End If
231 Case VBA.vbBoolean
232     If JsonValue Then
233         ConvertToJson = "true"
234     Else
235         ConvertToJson = "false"
236     End If
237 Case VBA.vbArray To VBA.vbArray + VBA.vbByte
238     If json_PrettyPrint Then
239         If VBA.VarType(Whitespace) = VBA.vbString Then
240             json_Indentation = VBA.String$(json_CurrentIndentation + 1, Whitespace)
241             json_InnerIndentation = VBA.String$(json_CurrentIndentation + 2, Whitespace)
242         Else
243             json_Indentation = VBA.Space$((json_CurrentIndentation + 1) * Whitespace)
244             json_InnerIndentation = VBA.Space$((json_CurrentIndentation + 2) *
245                 Whitespace)
246         End If
247     End If
248     ' Array
249     json_BufferAppend json_Buffer, "[" , json_BufferPosition, json_BufferLength
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,
264                     json_BufferLength
265             End If
266
267             If json_LBound2D >= 0 And json_UBound2D >= 0 Then
268                 ' 2D Array
269                 If json_PrettyPrint Then
270                     json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
271                         json_BufferLength
272                 End If
273                 json_BufferAppend json_Buffer, json_Indentation & "[" ,
274                     json_BufferPosition, json_BufferLength
275
276                 For json_Index2D = json_LBound2D To json_UBound2D
277                     If json_IsFirstItem2D Then
278                         json_IsFirstItem2D = False
279                     Else
280                         json_BufferAppend json_Buffer, "," , json_BufferPosition,

```

```

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

```

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,
337     json_BufferLength
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,
356             json_CurrentIndentation + 1)
357         If json_Converted = "" Then
358             json_SkipItem = json_IsUndefined(JsonValue(json_Key))
359         Else
360             json_SkipItem = False
361         End If
362         If Not json_SkipItem Then
363             If json_IsFirstItem Then
364                 json_IsFirstItem = False
365             Else
366                 json_BufferAppend json_Buffer, "," , json_BufferPosition,
367                     json_BufferLength
368             End If
369             If json_PrettyPrint Then
370                 json_Converted = vbNewLine & json_Indentation & "" "" & json_Key &
371                     "" ":" & json_Converted
372             Else
373                 json_Converted = "" "" & json_Key & "" ":" & json_Converted
374             End If
375             json_BufferAppend json_Buffer, json_Converted, json_BufferPosition,
376                 json_BufferLength
377         End If
378     Next json_Key
379     If json_PrettyPrint Then
380         json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
381             json_BufferLength
382     End If
383     If VBA.VarType(Whitespace) = VBA.vbString Then
384         json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)

```

1 2 3 4 5

```

384     1 2 3 4 5
385     }Else
386     }   json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
387     }   End If
388     }   End If
389     }   json_BufferAppend json_Buffer, json_Indentation & "}" , json_BufferPosition,
390     }   json_BufferLength
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,
399     }   }   }   }   }   json_BufferLength
400     }   }   }   }   }   End If
401     }   }   }   }   }   json_Converted = ConvertToJson(json_Value, Whitespace,
402     }   }   }   }   }   json_CurrentIndentation + 1)
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     }   }   }   }   }   }   If json_PrettyPrint Then
411     }   }   }   }   }   }   }   json_Converted = vbNewLine & json_Indentation & json_Converted
412     }   }   }   }   }   }   }   End If
413     }   }   }   }   }   }   }   json_BufferAppend json_Buffer, json_Converted, json_BufferPosition,
414     }   }   }   }   }   }   }   json_BufferLength
415     }   }   }   }   }   }   }   Next json_Value
416     }   }   }   }   }   }   If json_PrettyPrint Then
417     }   }   }   }   }   }   }   json_BufferAppend json_Buffer, vbNewLine, json_BufferPosition,
418     }   }   }   }   }   }   }   json_BufferLength
419     }   }   }   }   }   }   }   If VBA.VarType(Whitespace) = VBA.vbString Then
420     }   }   }   }   }   }   }   }   json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
421     }   }   }   }   }   }   }   }   Else
422     }   }   }   }   }   }   }   }   }   json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
423     }   }   }   }   }   }   }   }   }   End If
424     }   }   }   }   }   }   }   }   }   End If
425     }   }   }   }   }   }   }   }   }   End If
426     }   }   }   }   }   }   }   }   }   json_BufferAppend json_Buffer, json_Indentation & "]" , json_BufferPosition,
427     }   }   }   }   }   }   }   }   }   json_BufferLength
428     }   }   }   }   }   }   }   }   }   End If
429     }   }   }   }   }   }   }   }   }   ConvertToJson = json_BufferToString(json_Buffer, json_BufferPosition)
430     }   }   }   }   }   }   }   }   }   Case VBA.vbInteger, VBA.vbLong, VBA.vbSingle, VBA.vbDouble, VBA.vbCurrency, VBA.
431     }   }   }   }   }   }   }   }   }   vbDecimal
432     }   }   }   }   }   }   }   }   }   }   ' Number (use decimals for numbers)
433     }   }   }   }   }   }   }   }   }   }   ConvertToJson = VBA.Replace(JsonValue, ",", "." )
434     }   }   }   }   }   }   }   }   }   }   Case Else
435     }   }   }   }   }   }   }   }   }   }   ' vbEmpty, vbError, vbDataObject, vbByte, vbUserDefinedType
436     }   }   }   }   }   }   }   }   }   }   ' Use VBA's built-in to-string
437     }   }   }   }   }   }   }   }   }   '

```

```

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(
463         utc_LocalDate), utc_UTCDate
464
465     ConvertToUtc = utc_SystemTimeToDate(utc_UTCDate)
466     #End If
467
468     Exit Function
469
470     utc_ErrorHandling:
471     Err.Raise 10012, "UtcConverter.ConvertToUtc" , "UTC conversion error: " & Err.
472     Number & " - " & Err.Description
473 End Function
474
475 Private Sub json_BufferAppend(ByRef json_Buffer As String, _
476     ByRef json_Append As Variant, _
477     ByRef json_BufferPosition As Long, _
478     ByRef json_BufferLength As Long)
479     ' VBA can be slow to append strings due to allocating a new string for each append
480     ' Instead of using the traditional append, allocate a large empty string and then
481     ' copy string at append position
482     '
483     ' Example:
484     ' Buffer: "abc  "
485     ' Append: "def"
486     ' Buffer Position: 3
487     ' Buffer Length: 5
488     '
489     ' Buffer position + Append length > Buffer length -> Append chunk of blank space to
490     ' buffer
491     ' Buffer: "abc      "
492     ' Buffer Length: 10
493     '
494     ' Put "def" into buffer at position 3 (0-based)
495     ' Buffer: "abcdef  "
496     '
497     ' Approach based on cStringBuilder from vbAccelerator
498     '

```

```
1
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 0xFFFF
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

```

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
607         , "Expecting '['" )
607     Else

```

```

1 2
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

```



```

1 2 3 4
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
670     , "Expecting '"' or ''''")
671     End If
672     ' Check for colon and skip if present or throw if not present
673     json_SkipSpaces json_String, json_Index
674     If VBA.Mid$(json_String, json_Index, 1) <> ":" Then
675     Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String, json_Index
676     , "Expecting ':'")
677     Else
678     json_Index = json_Index + 1
679     End If
680     End Function
681 Private Function json_ParseNumber(json_String As String, ByRef json_Index As Long) As
Variant
682     Dim json_Char As String
683     Dim json_Value As String
684     Dim json_IsLargeNumber As Boolean
685
686     json_SkipSpaces json_String, json_Index
687
688     Do While json_Index > 0 And json_Index <= Len(json_String)
689     json_Char = VBA.Mid$(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
718 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
719 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$(json_String, json_Index, 1) <> "{" Then
726 Err.Raise 10001, "JsonConverter" , json_ParseErrorMessage(json_String, json_Index
726 , "Expecting '{'")
727 Else
728 json_Index = json_Index + 1
729
730 Do
731 json_SkipSpaces json_String, json_Index
732 If VBA.Mid$(json_String, json_Index, 1) = "}" Then
733 json_Index = json_Index + 1
734 Exit Function
735 ElseIf VBA.Mid$(json_String, json_Index, 1) = "," Then
736 json_Index = json_Index + 1
737 json_SkipSpaces json_String, json_Index
738 End If
739
740 json_Key = json_ParseKey(json_String, json_Index)
741 json_NextChar = json_Peek(json_String, json_Index)
742 If json_NextChar = "[" Or json_NextChar = "{" Then
743 Set json_ParseObject.Item(json_Key) = json_ParseValue(json_String,
743 json_Index)
744 Else
745 json_ParseObject.Item(json_Key) = json_ParseValue(json_String, json_Index)
746 End If
747 Loop
748 End If
749 End Function
750
751 Private Function json_ParseString(json_String As String, ByRef json_Index As Long) As
751 String
752 Dim json_Quote As String
753 Dim json_Char As String
754 Dim json_Code As String
755 Dim json_Buffer As String
756 Dim json_BufferPosition As Long
757 Dim json_BufferLength As Long
758
759 json_SkipSpaces json_String, json_Index
760
761 ' Store opening quote to look for matching closing quote
762 json_Quote = VBA.Mid$(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

```

1 2 3
770 ' Escaped string, \, or \/
771 json_Index = json_Index + 1
772 json_Char = VBA.Mid$(json_String, json_Index, 1)
773
774 Select Case json_Char
775 Case "" "" , "\", "/" , ""
776     json_BufferAppend json_Buffer, json_Char, json_BufferPosition,
777         json_BufferLength
778     json_Index = json_Index + 1
779 Case "b"
780     json_BufferAppend json_Buffer, vbBack, json_BufferPosition,
781         json_BufferLength
782     json_Index = json_Index + 1
783 Case "f"
784     json_BufferAppend json_Buffer, vbFormFeed, json_BufferPosition,
785         json_BufferLength
786     json_Index = json_Index + 1
787 Case "n"
788     json_BufferAppend json_Buffer, vbCrLf, json_BufferPosition,
789         json_BufferLength
790     json_Index = json_Index + 1
791 Case "r"
792     json_BufferAppend json_Buffer, vbCr, json_BufferPosition, json_BufferLength
793     json_Index = json_Index + 1
794 Case "t"
795     json_BufferAppend json_Buffer, vbTab, json_BufferPosition,
796         json_BufferLength
797     json_Index = json_Index + 1
798 Case "u"
799     ' Unicode character escape (e.g. \u00a9 = Copyright)
800     json_Index = json_Index + 1
801     json_Code = VBA.Mid$(json_String, json_Index, 4)
802     json_BufferAppend json_Buffer, VBA.Chw(VBA.Val("&h" + json_Code)),
803         json_BufferPosition, json_BufferLength
804     json_Index = json_Index + 4
805 End Select
806 Case json_Quote
807     json_ParseString = json_BufferToString(json_Buffer, json_BufferPosition)
808     json_Index = json_Index + 1
809     Exit Function
810 Case Else
811     json_BufferAppend json_Buffer, json_Char, json_BufferPosition,
812         json_BufferLength
813     json_Index = json_Index + 1
814 End Select
815 Loop
816 End Function
817
818 Private Function json_ParseValue(json_String As String, ByRef json_Index As Long) As
819     Variant
820     json_SkipSpaces json_String, json_Index
821     Select Case VBA.Mid$(json_String, json_Index, 1)
822     Case "{"
823         Set json_ParseValue = json_ParseObject(json_String, json_Index)
824     Case "["
825         Set json_ParseValue = json_ParseArray(json_String, json_Index)
826     Case "" "" , ""
827         json_ParseValue = json_ParseString(json_String, json_Index)
828     Case Else
829         If VBA.Mid$(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$(json_String, json_Index, 1)) Then
831         json_ParseValue = json_ParseNumber(json_String, json_Index)
832     Else
833         Err.Raise 10001, "JSONConverter" , json_ParseErrorMessage(json_String,
834             json_Index, "Expecting 'STRING', 'NUMBER', null, true, false, '{', or '['" )
835     End If
836 End Select
837 End Function
838 Private Function json_Peek(json_String As String, ByVal json_Index As Long, Optional
839     json_NumberOfCharacters As Long = 1) As String
840     ' "Peek" at the next number of characters without incrementing json_Index (ByVal
841     instead of ByRef)
842     json_SkipSpaces json_String, json_Index
843     json_Peek = VBA.Mid$(json_String, json_Index, json_NumberOfCharacters)
844 End Function
845 Private Sub json_SkipSpaces(json_String As String, ByRef json_Index As Long)
846     ' Increment index to skip over spaces
847     Do While json_Index > 0 And json_Index <= VBA.Len(json_String) And VBA.Mid$(
848     json_String, json_Index, 1) = " "
849     json_Index = json_Index + 1
850 Loop
851 End Sub
852 Private Function json_StringIsLargeNumber(json_String As Variant) As Boolean
853     ' Check if the given string is considered a "large number"
854     ' (See json_ParseNumber)
855     Dim json_Length As Long
856     Dim json_CharIndex As Long
857     json_Length = VBA.Len(json_String)
858     ' Length with be at least 16 characters and assume will be less than 100 characters
859     If json_Length >= 16 And json_Length <= 100 Then
860         Dim json_CharCode As String
861         json_StringIsLargeNumber = True
862         For json_CharIndex = 1 To json_Length
863             json_CharCode = VBA.Asc(VBA.Mid$(json_String, json_CharIndex, 1))
864             Select Case json_CharCode
865                 ' Look for .|0-9|E|e
866                 Case 46, 48 To 57, 69, 101
867                     ' Continue through characters
868                 Case Else
869                     json_StringIsLargeNumber = False
870                     Exit Function
871             End Select
872         Next json_CharIndex
873     End If
874 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)),
902     VBA.CInt(utc_DateParts(2)))
903
904     If UBound(utc_Parts) > 0 Then
905         If VBA.InStr(utc_Parts(1), "Z" ) Then
906             utc_TimeParts = VBA.Split(VBA.Replace(utc_Parts(1), "Z" , "" ), ":" )
907         Else
908             utc_OffsetIndex = VBA.InStr(1, utc_Parts(1), "+" )
909             If utc_OffsetIndex = 0 Then
910                 utc_NegativeOffset = True
911                 utc_OffsetIndex = VBA.InStr(1, utc_Parts(1), "-" )
912             End If
913
914             If utc_OffsetIndex > 0 Then
915                 utc_HasOffset = True
916                 utc_TimeParts = VBA.Split(VBA.Left$(utc_Parts(1), utc_OffsetIndex - 1),
917                 ":")
918                 utc_OffsetParts = VBA.Split(VBA.Right$(utc_Parts(1), Len(utc_Parts(1)) -
919                 utc_OffsetIndex), ":" )
920
921                 Select Case UBound(utc_OffsetParts)
922                     Case 0
923                         utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), 0, 0)
924                     Case 1
925                         utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), VBA.CInt(
926                         utc_OffsetParts(1)), 0)
927                     Case 2
928                         ' VBA.Val does not use regional settings, use for seconds to avoid
929                         decimal/comma issues
930                         utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), VBA.CInt(
931                         utc_OffsetParts(1)), Int(VBA.Val(utc_OffsetParts(2))))
932                     End Select
933
934                 If utc_NegativeOffset Then: utc_Offset = -utc_Offset
935             Else
936                 utc_TimeParts = VBA.Split(utc_Parts(1), ":" )
937             End If
938         End If
939     End If

```

1 2

```

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(
939             utc_TimeParts(1)), 0)
940     Case 2
941         ' VBA.Val does not use regional settings, use for seconds to avoid
942         ' decimal/comma issues
943         ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), VBA.CInt(
944             utc_TimeParts(1)), Int(VBA.Val(utc_TimeParts(2))))
945     End Select
946
947     ParseIso = ParseUtc(ParseIso)
948
949     If utc_HasOffset Then
950         ParseIso = ParseIso - utc_Offset
951     End If
952 End If
953
954 Exit Function
955
956 utc_ErrorHandling:
957 Err.Raise 10013, "UtcConverter.ParseIso", "ISO 8601 parsing error for " &
958 utc_IsoString & ": " & Err.Number & " - " & Err.Description
959 End Function
960
961 ' ===== '
962 ' Public Methods
963 ' ===== '
964
965 ''
966 ' Convert JSON string to object (Dictionary/Collection)
967 '
968 ' @method ParseJson
969 ' @param {String} json_String
970 ' @return {Object} (Dictionary or Collection)
971 ' @throws 10001 - JSON parse error
972 ''
973 Public Function ParseJson(ByVal JsonString As String) As Object
974     Dim json_Index As Long
975     json_Index = 1
976
977     ' Remove vbCr, vbLf, and vbTab from json_String
978     JsonString = VBA.Replace(VBA.Replace(VBA.Replace(JsonString, VBA.vbCr, "" ), VBA.
979         vbLf, "" ), VBA.vbTab, "" )
980
981     json_SkipSpaces JsonString, json_Index
982     Select Case VBA.Mid$(JsonString, json_Index, 1)
983     Case "{"
984         Set ParseJson = json_ParseObject(JsonString, json_Index)
985     Case "["
986         Set ParseJson = json_ParseArray(JsonString, json_Index)
987     Case Else
988         ' Error: Invalid JSON string
989         Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(JsonString, json_Index,
990             "Expecting '{' or '['" )
991     End Select
992 End Function
993
994 End Function

```

```

988 ''
989 ' VBA-UTC v1.0.6
990 ' (c) Tim Hall - https://github.com/VBA-tools/VBA-UtcConverter
991 '
992 ' UTC/ISO 8601 Converter for VBA
993 '
994 ' Errors:
995 ' 10011 - UTC parsing error
996 ' 10012 - UTC conversion error
997 ' 10013 - ISO 8601 parsing error
998 ' 10014 - ISO 8601 conversion error
999 '
1000 ' @module UtcConverter
1001 ' @author tim.hall.engr@gmail.com
1002 ' @license MIT (http://www.opensource.org/licenses/mit-license.php)
1003 ' ~~~~~
1004
1005 ' (Declarations moved to top)
1006
1007 ' =====
1008 ' Public Methods
1009 ' =====
1010
1011 ''
1012 ' Parse UTC date to local date
1013 '
1014 ' @method ParseUtc
1015 ' @param {Date} UtcDate
1016 ' @return {Date} Local date
1017 ' @throws 10011 - UTC parsing error
1018 ''
1019 Public Function ParseUtc(utc_UtcDate As Date) As Date
1020     On Error GoTo utc_ErrorHandling
1021
1022     #If Mac Then
1023         ParseUtc = utc_ConvertDate(utc_UtcDate)
1024     #Else
1025         Dim utc_TimeZoneInfo As utc_TIME_ZONE_INFORMATION
1026         Dim utc_LocalDate As utc_SYSTEMTIME
1027
1028         utc_GetTimeZoneInformation utc_TimeZoneInfo
1029         utc_SystemTimeToTzSpecificLocalTime utc_TimeZoneInfo, utc_DateToSystemTime(
1030             utc_UtcDate), utc_LocalDate
1031
1032         ParseUtc = utc_SystemTimeToDate(utc_LocalDate)
1033     #End If
1034
1035     Exit Function
1036
1037     utc_ErrorHandling:
1038     Err.Raise 10011, "UtcConverter.ParseUtc", "UTC parsing error: " & Err.Number & "
1039         - " & Err.Description
1040 End Function
1041
1042 ' =====
1043 ' Private Functions
1044 ' =====
1045
1046 #If Mac Then
1047 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         ) + _
1074         TimeSerial(utc_TimeParts(0), utc_TimeParts(1), utc_TimeParts(2))
1075 End If
1076 End Function
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



```
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,
1120     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
```

\_, 9, 10, 17, 20, 28, 29

## A

Activate, 7  
 ActiveExplorer, 3  
 ActiveInspector, 3  
 Add, 3, 6, 20  
 AIBody, 3, 4  
 AllowUnquotedKeys, 11, 20  
 APIKey, 2-4  
 Asc, 24  
 AscW, 18

## B

Body, 3  
 BrowseForFolder, 6

## C

CheckForAPIKey, 2-4  
 choices, 3  
 Chr, 4  
 ChrW, 23  
 Close, 5, 6  
 Collection, 19  
 CompletedText, 3  
 ConvertToIso, 12  
 ConvertToJson, 3, 12-17  
 ConvertToUtc, 12, 17  
 CreateObject, 3, 5-7  
 CustomReply, 2

## D

DateSerial, 25, 28, 29  
 Day, 28  
 Debug, 3, 4  
 Description, 12, 17, 26, 27  
 Dictionary, 22  
 Dir, 5-7  
 Display, 4  
 Domain, 2-4

## E

EmailBody, 3, 5  
 EmailFrom, 3, 5  
 EmailReply\_Accept, 2  
 EmailReply\_CustomReply, 2  
 EmailReply\_Reject, 2  
 EmailReply\_RequestForInfo, 2  
 EmailReply\_StronglyAccept, 2  
 EmailReply\_StronglyReject, 3  
 Empty, 5  
 Err, 12, 17, 19, 21, 22, 24, 26-28  
 ErrorMessage, 20  
 EscapeSolidus, 11, 19  
 Explicit, 2, 6, 9

## F

FolderPath, 6  
 Format\$, 12, 28

## G

GetSetting, 2-7

## H

Hex\$, 19  
 Hide, 8  
 Hour, 28  
 HTMLBody, 4

## I

IIf, 18, 21  
 InputBox, 2, 4  
 InStr, 4, 21, 24, 25  
 InStrRev, 4  
 Int, 25, 26  
 IsMissing, 12  
 Item, 22

## J

Json, 3  
 json\_AddedLength, 18  
 json\_Append, 17, 18  
 json\_AppendLength, 18  
 json\_AscCode, 18, 19  
 json\_Buffer, 12-19, 22, 23  
 json\_BufferAppend, 13-17, 19, 23  
 json\_BufferLength, 12-19, 22, 23  
 json\_BufferPosition, 12-19, 22, 23  
 json\_BufferToString, 15, 16, 18, 19, 23  
 json\_Char, 18-23  
 json\_CharCode, 24  
 json\_CharIndex, 24  
 json\_Code, 22, 23  
 json\_Converted, 12, 14-16  
 json\_CurrentIndentation, 12-16  
 json\_DateStr, 12  
 json\_Encode, 13, 18, 19  
 json\_Indentation, 12-16  
 json\_Index, 12-14, 18-24, 26  
 json\_Index2D, 12-14  
 json\_InnerIndentation, 12-14  
 json\_IsFirstItem, 12, 13, 15, 16  
 json\_IsFirstItem2D, 12-14  
 json\_IsLargeNumber, 21  
 json\_IsUndefined, 14-16, 19  
 json\_Key, 12, 15, 22  
 json\_LBound, 12, 13  
 json\_LBound2D, 12, 13  
 json\_Length, 24  
 json\_LengthPlusPosition, 18  
 json\_NextChar, 22  
 json\_NumberOfCharacters, 24  
 json\_Options, 11  
 json\_ParseArray, 19, 20, 23, 26  
 json\_ParseErrorMessage, 19-22, 24, 26  
 json\_ParseKey, 20, 22  
 json\_ParseNumber, 21, 24  
 json\_ParseObject, 22, 23, 26  
 json\_ParseString, 20, 22, 23  
 json\_ParseValue, 20, 22-24  
 json\_Peek, 22, 24  
 json\_PrettyPrint, 12-16  
 json\_Quote, 22, 23  
 json\_SkipItem, 12, 15  
 json\_SkipSpaces, 19-24, 26  
 json\_StartIndex, 20  
 json\_StopIndex, 20  
 json\_String, 19-24  
 json\_StringIsLargeNumber, 13, 24  
 json\_Text, 18  
 json\_UBound, 12, 13  
 json\_UBound2D, 12, 13  
 json\_Value, 12, 16, 19, 21  
 JsonConverter, 3  
 JsonObject, 3  
 JsonOptions, 11, 13, 19-21  
 JsonString, 3, 26  
 JsonValue, 12-17

## K

Key, 2, 4  
 Keys, 15

## L

LangReply, 2, 3, 6-8  
 LBound, 13  
 Left, 4  
 Left\$, 18, 25, 29  
 Len, 4, 18, 20-22, 24, 25, 29  
 LogFilePath, 4, 6, 7  
 LogFileSettings, 6, 7  
 LogFolder, 6, 7  
 LogPath, 3-8  
 LogPathScript, 7  
 LogRow, 3, 5  
 LogShell, 6, 7  
 LongPtr, 10, 28

## M

Mac, 2-4, 6, 7, 9, 10, 17, 27  
 MacScript, 2, 4, 7  
 MailItem, 3  
 Mid\$, 18-24, 26  
 Minute, 28  
 Month, 28  
 MsgBox, 3-5, 7

## N

Now, 5  
 Null, 24  
 Number, 12, 17, 26, 27

## O

objExcel, 3, 5, 7  
 objWorkbook, 3, 5, 7  
 olApp, 3  
 olDocument, 3  
 olLink, 3  
 olInspector, 3  
 olLink, 3  
 olMail, 3  
 olRange, 3  
 Open, 3, 5, 7  
 OpenLogFile, 7  
 oXMLHTTP, 3, 4

## P

ParseIso, 25, 26  
 ParseJson, 26  
 ParseUtc, 26, 27  
 Path, 6  
 Print, 3, 4  
 Prompt, 3  
 PtrSafe, 9, 10

## R

Raise, 12, 17, 19, 21, 22, 24, 26-28  
 Range, 5, 6  
 Replace, 4, 16, 25, 26  
 Reply, 4  
 ReplyEmail, 3-5  
 ReplyText, 3-5  
 ReplyType, 3, 5  
 ReplyWithChatGPT, 2, 3  
 Response, 3, 4  
 responseText, 4  
 Right, 4  
 Right\$, 19, 25

Row, 5

## S

SameLangReply, 6, 7  
 SaveAs, 6  
 SaveChanges, 6  
 SaveEmailsBox, 6, 7  
 SaveLogFormSettings, 7  
 SaveOpt, 3-5, 7, 8  
 SaveOption, 4, 6, 7  
 SaveOptScript, 7  
 SaveSetting, 2, 4, 7, 8  
 SaveToExcelForm, 6, 7  
 ScreenUpdating, 5, 7  
 Script, 2  
 Second, 28  
 selectedEmail, 3, 4  
 SelectEmail, 3, 5  
 Selection, 3  
 Self, 6  
 Send, 4  
 SenderEmailAddress, 3  
 setRequestHeader, 3  
 Sheets, 5, 6  
 Show, 6  
 Space\$, 13, 15, 16, 18, 20, 29  
 Split, 25, 28  
 String\$, 13, 15, 16  
 Subject, 5

## T

Timeout, 3, 5  
 TimeSerial, 25, 26, 28, 29  
 TypeName, 15, 16, 19

## U

UBound, 13, 25, 26  
 URL, 3  
 UseDoubleForLargeNumbers, 11, 13, 21  
 utc\_Bias, 11  
 utc\_Buffer, 10  
 utc\_Chunk, 28, 29  
 utc\_Command, 10  
 utc\_ConvertDate, 17, 27, 28  
 utc\_ConvertToUtc, 17, 27, 28  
 utc\_DateParts, 25, 28  
 utc\_DateToSystemTime, 17, 27, 28  
 utc\_DaylightBias, 11  
 utc\_DaylightDate, 11  
 utc\_DaylightName, 11  
 utc\_ErrorHandling, 12, 17, 25-29  
 utc\_ExecutelnShell, 28, 29  
 utc\_ExitCode, 10, 11, 29  
 utc\_feof, 9, 10, 28  
 utc\_File, 10, 28, 29  
 utc\_fread, 9, 10, 29  
 utc\_GetTimeZonelnformation, 10, 17, 27  
 utc\_HasOffset, 25, 26  
 utc\_IsoString, 25, 26  
 utc\_LocalDate, 12, 17, 27  
 utc\_lpLocalTime, 10  
 utc\_lpTimeZonelnformation, 10  
 utc\_lpUniversalTime, 10  
 utc\_Mode, 10  
 utc\_NegativeOffset, 25  
 utc\_Number, 10  
 utc\_Offset, 25, 26  
 utc\_OffsetIndex, 25  
 utc\_OffsetParts, 25  
 utc\_Output, 10, 11, 28, 29

utc\_Parts, 25, 28  
 utc\_pclose, 9, 10, 29  
 utc\_popen, 9, 10, 28  
 utc\_Read, 28, 29  
 utc\_Result, 28  
 utc\_ShellCommand, 28  
 utc\_ShellResult, 10, 28  
 utc\_Size, 10  
 utc\_StandardBias, 11  
 utc\_StandardDate, 11  
 utc\_StandardName, 11  
 utc\_SYSTEMTIME, 10, 11, 17, 27-29  
 utc\_SystemTimeToDate, 17, 27, 29  
 utc\_SystemTimeToTzSpecificLocalTime, 10, 27  
 utc\_TIME\_ZONE\_INFORMATION, 10, 11, 17, 27  
 utc\_TimeParts, 25, 26, 28  
 utc\_TimeZonelnfo, 17, 27  
 utc\_TzSpecificLocalTimeToSystemTime, 10, 17  
 utc\_UtcDate, 17, 27  
 utc\_Value, 27-29  
 utc\_wDay, 11, 28, 29  
 utc\_wDayOfWeek, 11  
 utc\_wHour, 11, 28, 29  
 utc\_wMilliseconds, 11, 28  
 utc\_wMinute, 11, 28, 29  
 utc\_wMonth, 11, 28, 29  
 utc\_wSecond, 11, 28, 29  
 utc\_wYear, 11, 28, 29

## V

Val, 21, 23, 25, 26  
 Value, 5-7  
 VarType, 12, 13, 15, 16, 19  
 VBA, 12, 13, 15, 16, 18-26, 28, 29  
 VBA7, 9, 10, 28  
 vbArray, 13  
 vbBack, 23  
 vbBoolean, 13  
 vbByte, 13  
 vbCr, 23, 26  
 vbCrLf, 2, 4, 23  
 vbCurrency, 16  
 vbDate, 12  
 vbDecimal, 16  
 vbDirectory, 5-7  
 vbDouble, 16  
 vbEmpty, 19  
 vbExclamation, 3  
 vbFormFeed, 23  
 vbInteger, 16  
 vbLf, 26  
 vbLong, 16  
 vbNewLine, 4, 13-16, 20  
 vbNull, 12  
 vbObject, 15, 19  
 vbSingle, 16  
 vbString, 13, 15, 16  
 vbTab, 23, 26  
 Visible, 7

## W

Whitespace, 12-16  
 WkBk, 6  
 Workbooks, 5-7  
 WrapText, 5

## X

xlup, 5

## Y

Year, 28

# 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*