Annex A: Full Code

```
"

'Library created by Pol Sarmiento Lozano @TDdeviceAPI

'

'This library is to be used in the context of data management of IoT devices

'

'This library relates to the URL requests available in the device API of the TD SENSOR platform

'

'The URL requests where last checked to be available 2016, September 28th

'

'This library is better used in conjuction with  "TDdataprocess" "TDfunctionsDescriptions" and "JsonConverter"




Option Explicit 'forces explicit declaration of all the variables in the file



#If VBA7 Then 'supports Office 64-bit

    Private Declare PtrSafe Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef Source As Any, ByVal Length As LongPtr)



#Else ' supports Office 32-bit
```

```
    Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef
Destination As Any, ByRef Source As Any, ByVal Length As Long)

#End If


'this process manages the different http requests by creating objects XML and setting the
headers required


Sub HttpRequest(sUrl As String, token As String, id As Variant, key As Variant, Method As
String, _

Typo As Integer, response As String)


Dim answer As String

Dim loading As String


Select Case Typo


Case 1 'GET request with authentication and json content


   With CreateObject("Microsoft.XMLHTTP")

      .Open Method, sUrl, False, id, key

      .SetRequestHeader "X-Snsr-Device-Key", token

      .SetRequestHeader "Content-Type", "application/json"

      .Send

      response = .responseText
```

```
        answer = .Status
        loading = .ReadyState

    End With

Case 2 'GET request without authentication

    With CreateObject("Microsoft.XMLHTTP")
        .Open Method, sUrl, False
        .Send
         response = .responseText
         answer = .Status
         loading = .ReadyState
    End With

Case 3 ' POST Request with Authentication

    With CreateObject("Microsoft.XMLHTTP")
        .Open Method, sUrl, False
        .SetRequestHeader "X-Snsr-Device-Key", token
        .Send ("sn=" + id + "&key=" + key)
         response = .responseText
         answer = .Status
```

```
        loading = .ReadyState
    End With

Case 4 'GET request without authentication and json content

    With CreateObject("Microsoft.XMLHTTP")
        .Open Method, sUrl, False, id, key
        .SetRequestHeader "Content-Type", "application/json"
        .Send
        response = .responseText
        answer = .Status
        loading = .ReadyState

    End With
End Select

If loading = 4 Then

    If answer = "200" Then
        If answer = "200" Then
        Else
        response = "Url error"
        End If
```

```
    Else
    End If

Else
    response = "Loading error"
End If

End Sub

'http request for post with extra content

Sub HttpRequest2(sUrl As String, token As String, id As Variant, key As Variant, Method As String, Typo As Integer, value As Variant, response As String)

Dim answer As String
Dim loading As String

Select Case Typo
Case 5 ' POST Request with Authentication and extra content

    With CreateObject("Microsoft.XMLHTTP")
        .Open Method, sUrl, False
        .SetRequestHeader "X-Snsr-Device-Key", token
        .Send ("sn=" + id + "&key=" + key + "&value=" + value)
```

```
        response = .responseText
        answer = .Status
        loading = .ReadyState
    End With
End Select

If loading = 4 Then

    If answer = "200" Then
        If answer = "200" Then
        Else
        response = "Url error"
        End If
    Else
    End If

Else
    response = "Loading error"
End If

End Sub

'this function gets the PAC key of the module
```

```
Function GetPAC(id As String, key As String) As String

Dim response As String
Dim sUrl As String
Dim token As String
Dim Method As String
Dim Typo As Integer

token = GetToken(id, key)
sUrl = "https://sensor.insgroup.fr/iot/devices/pacs.csv?"
Method = "POST"
Typo = 3

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function
```

```
Else

   GetPAC = Right(response, 17) 'only the 16characters of the PAC Key in the .csv file needed

End If

End Function


'this function gets the authentication token for the device api requests


Function GetToken(id, key) As String


Dim response As String

Dim sUrl As String

Dim token As String

Dim Method As String

Dim Typo As Integer


sUrl = "http://sensor.insgroup.fr/iot/devices/crc.json?" + "&sn=" + id + "&key=" + key

Method = "GET"

Typo = 2


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then
```

```
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    GetToken = response
End If

End Function

'this function gets punctual payload data from a single IoT message by method history

Function GetPayloadMessage(id, key, n As Integer, gmt As String, m As Integer) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
```

```
token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

   MsgBox ("Url Error, please try again in a few minutes")

   Exit Function


ElseIf response = "Loading error" Then

   MsgBox ("Loading Error, please try again in a few minutes")

   Exit Function


Else

   Set Json = JsonConverter.ParseJson(response)


   GetPayloadMessage = Json("items")(m)("payload")


End If


End Function
```

```
'this function gets punctual type of message data from a single IoT message by method history

Function GetTypeMessage(id, key, n As Integer, gmt As String, m As Integer) As String

Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer

token = GetToken(id, key)
sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)
Method = "GET"
Typo = 1

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
```

```
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    Set Json = JsonConverter.ParseJson(response)

    GetTypeMessage = Json("items")(m)("type")

End If

End Function


'this function gets punctual ascii message data from a single IoT message by method history

Function GetAsciiMessage(id, key, n As Integer, gmt As String, m As Integer) As String

Dim response As String
Dim sUrl As String
Dim Json As Object
```

```
Dim token As String

Dim Method As String

Dim Typo As Integer


token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function


ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")

    Exit Function


Else

    Set Json = JsonConverter.ParseJson(response)


    If Json("items")(m)("type") = "raw" Then
```

```
        GetAsciiMessage = Json("items")(m)("extra")("asciiraw")

    ElseIf Json("items")(m)("type") = "service" Then

        GetAsciiMessage = Json("items")(m)("extra")("asciidata")
    Else
    End If

End If

End Function

'this function gets punctual data field from a single IoT message by method history

Function GetFieldData(id, key, n As Integer, gmt As String, m As Integer) As String

Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
```

```
token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function


ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")

    Exit Function


Else

    Set Json = JsonConverter.ParseJson(response)


    GetFieldData = Json("items")(m)("extra")("data")


End If
```

```
End Function

'this function gets punctual encoded message data from a single IoT message by method history

Function GetEncodedMessage(id, key, n As Integer, gmt As String, m As Integer) As String

Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer

token = GetToken(id, key)
sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)
Method = "GET"
Typo = 1

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
```

```
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    Set Json = JsonConverter.ParseJson(response)

    GetEncodedMessage = Json("items")(m)("extra")("message")

End If

End Function

'this function gets puncutal coordinates data from a single IoT message by method history

Function GetCoordinates(id, key, n As Integer, gmt As String, m As Integer) As String

Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
```

```
Dim Method As String

Dim Typo As Integer


token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

   MsgBox ("Url Error, please try again in a few minutes")

   Exit Function


ElseIf response = "Loading error" Then

   MsgBox ("Loading Error, please try again in a few minutes")

   Exit Function


Else

   Set Json = JsonConverter.ParseJson(response)


   GetCoordinates = Json("items")(m)("ctxt")("position")("latitude") & "ºN " & "" & Json("items")(m)("ctxt")("position")("longitude") & "ºE"
```

```
End If


End Function


'this function gets total data from IoT messages by method history


Function GetDataHistory(id As String, key As String, n As Integer, gmt As String) As String


Dim response As String

Dim sUrl As String

Dim Json As Object

Dim token As String

Dim Method As String

Dim Typo As Integer


token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)
```

```
If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    GetDataHistory = response

End If

End Function

'this function gets total data from IoT messages by method recent

Private Function GetDataRecent(id As String, key As String, n As Integer, gmt As String) As
String

Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
```

```
Dim Method As String

Dim Typo As Integer


token = GetToken(id, key)

sUrl = "https://sensor.insgroup.fr/iot/devices/msgs/recents.json?&amount=" + CStr(n) + "&after=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function


ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")

    Exit Function


Else

    GetDataRecent = response


End If
```

```
End Function

'this function gets the childrens information of a gateway module

Function GetChildren(id, key) As String

Dim response As String
Dim sUrl As String
Dim token As String
Dim Method As String
Dim Typo As Integer

token = GetToken(id, key)
sUrl = "https://sensor.insgroup.fr/iot/devices/children.json?"
Method = "GET"
Typo = 1

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function
```

```
ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    GetChildren = response
End If

End Function

'this function clears data from the Iot messages database providing the admin pass first

Function ClearData(id, key, Adminpass As String) As String

Dim response As String
Dim sUrl As String
Dim token As String
Dim Method As String
Dim Typo As Integer

If Adminpass = "Admin" Then

token = GetToken(id, key)
```

```
sUrl = "https://sensor.insgroup.fr/iot/devices/clear.json?"
Method = "POST"
Typo = 2

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    ClearData = response & "Data Cleared successfuly"
End If

Else

MsgBox "You either are not the Admin or do not know the password"
Exit Function
```

```
End If

End Function

'This function changes the device functional status value

Function ChangeDfuncstat(id As String, key As String, value As Variant, Adminpass As String) As String

Dim response As String

Dim sUrl As String

Dim token As String

Dim Method As String

Dim Typo As Integer

If Adminpass = "Admin" Then

token = GetToken(id, key)

sUrl = "https://sensor.insgroup.fr/iot/devices/status.json?"

Method = "POST"

Typo = 5

Call HttpRequest2(sUrl, token, id, key, Method, Typo, value, response)
```

```
If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    ChangeDfuncstat = "Functional status changed to" & vbCr & value
End If

Else

MsgBox "You either are not the Admin or do not know the password"
Exit Function

End If

End Function

'This function toggles the active flag value
```

```
Function Toggleactiveflag(id As String, key As String, value As Variant, Adminpass As String) As String

Dim response As String

Dim sUrl As String

Dim token As String

Dim Method As String

Dim Typo As Integer

If Adminpass = "Admin" Then

token = GetToken(id, key)

sUrl = "https://sensor.insgroup.fr/iot/devices/status.json?"

Method = "POST"

Typo = 5

Call HttpRequest2(sUrl, token, id, key, Method, Typo, value, response)

If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function

ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")
```

```
    Exit Function

Else
    Toggleactiveflag = "Active flag changed to" & vbCr & value
End If

Else

MsgBox "You either are not the Admin or do not know the password"
Exit Function

End If

End Function

'This function changes the monitoring flag value

Function Togglemonitorflag(id As String, key As String, value As Variant, Adminpass As String) As String

Dim response As String
Dim sUrl As String
Dim token As String
Dim Method As String
```

```
Dim Typo As Integer

If Adminpass = "Admin" Then

token = GetToken(id, key)
sUrl = "https://sensor.insgroup.fr/iot/devices/status.json?"
Method = "POST"
Typo = 5

Call HttpRequest2(sUrl, token, id, key, Method, Typo, value, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function

ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    Togglemonitorflag = "Monitoring flag changed to" & vbCr & value
End If
```

```
Else

MsgBox "You either are not the Admin or do not know the password"
Exit Function

End If

End Function

'This function changes the bidir value

Function ChangeBidir(id As String, key As String, value As Variant, Adminpass As String)
As String

Dim response As String
Dim sUrl As String
Dim token As String
Dim Method As String
Dim Typo As Integer

If Adminpass = "Admin" Then

token = GetToken(id, key)
sUrl = "https://sensor.insgroup.fr/iot/devices/status.json?"
```

```
Method = "POST"

Typo = 5


Call HttpRequest2(sUrl, token, id, key, Method, Typo, value, response)


If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function


ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")

    Exit Function


Else

    ChangeBidir = "Bidir value changed to" & vbCr & value

End If


Else


MsgBox "You either are not the Admin or do not know the password"

Exit Function


End If
```

```
End Function

'this function gets the last date the device was operational

Function GetLastSeen(id, key, n As Integer, gmt As String, m As Integer) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer

token = GetToken(id, key)
sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)
Method = "GET"
Typo = 1

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function
```

```
ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    Set Json = JsonConverter.ParseJson(response)

    GetLastSeen = Json("items")(m)("ctxt")("lastseen")

End If

End Function

'this function gets the first date the device was operational

Function GetFirstSeen(id, key, n As Integer, gmt As String, m As Integer) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
```

```
token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

   MsgBox ("Url Error, please try again in a few minutes")

   Exit Function


ElseIf response = "Loading error" Then

   MsgBox ("Loading Error, please try again in a few minutes")

   Exit Function


Else

   Set Json = JsonConverter.ParseJson(response)


   GetFirstSeen = Json("items")(m)("ctxt")("firstseen")


End If
```

```
End Function

'this function gets the date the message was sent by the device

Function GetMessageSent(id, key, n As Integer, gmt As String, m As Integer) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
Dim fresponse As Double

token = GetToken(id, key)
sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)
Method = "GET"
Typo = 1

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function
```

```
ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    Set Json = JsonConverter.ParseJson(response)

    fresponse = Json("items")(m)("received")

    GetMessageSent = Format(MilistoDate(fresponse), "mm/dd/yy hh:mm:ss")

End If

End Function

'this function gets the date the message was received by SENSOR

Function GetMessageReceived(id, key, n As Integer, gmt As String, m As Integer) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
```

```
Dim token As String

Dim Method As String

Dim Typo As Integer

Dim fresponse As Double


token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function


ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")

    Exit Function


Else

    Set Json = JsonConverter.ParseJson(response)
```

```
    fresponse = Json("items")(m)("when")

    GetMessageReceived = Format(MilistoDate(fresponse), "mm/dd/yy hh:mm:ss")

End If

End Function

'this function gets the time difference between the date the message was received by SENSOR and the date the message was sent by the device

Function GetTravelTimeMessage(id, key, n As Integer, gmt As String, m As Integer) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
Dim fresponse As Double

token = GetToken(id, key)
sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)
```

```
Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

    MsgBox ("Url Error, please try again in a few minutes")

    Exit Function


ElseIf response = "Loading error" Then

    MsgBox ("Loading Error, please try again in a few minutes")

    Exit Function


Else

    Set Json = JsonConverter.ParseJson(response)


    fresponse = Json("items")(m)("when") - Json("items")(m)("received")


    GetTravelTimeMessage = Format(MilistoDate(fresponse), "hh:mm:ss")


End If


End Function
```

```
'this function gets the current number of messages stored at the server

Function GetCurrentMessagesServer(id, key, gmt As String) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
Dim n As Integer

n = 1
token = GetToken(id, key)
sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)
Method = "GET"
Typo = 1

Call HttpRequest(sUrl, token, id, key, Method, Typo, response)

If response = "Url error" Then
    MsgBox ("Url Error, please try again in a few minutes")
    Exit Function
```

```
ElseIf response = "Loading error" Then
    MsgBox ("Loading Error, please try again in a few minutes")
    Exit Function

Else
    Set Json = JsonConverter.ParseJson(response)

    GetCurrentMessagesServer = Json("count")

End If

End Function

'this function gets the total number of messages that have been stored at the server at some point

Function GetTotalMessagesServer(id, key, gmt As String) As String
Dim response As String
Dim sUrl As String
Dim Json As Object
Dim token As String
Dim Method As String
Dim Typo As Integer
```

```
Dim n As Integer


n = 1

token = GetToken(id, key)

sUrl = "http://sensor.insgroup.fr/iot/devices/msgs/history.json?&amount=" + CStr(n) + "&until=" + CStr(gmt)

Method = "GET"

Typo = 1


Call HttpRequest(sUrl, token, id, key, Method, Typo, response)


If response = "Url error" Then

   MsgBox ("Url Error, please try again in a few minutes")

   Exit Function


ElseIf response = "Loading error" Then

   MsgBox ("Loading Error, please try again in a few minutes")

   Exit Function


Else

   Set Json = JsonConverter.ParseJson(response)


   GetTotalMessagesServer = Json("items")(n)("ctxt")("msgs")
```

```
End If


End Function

'This sub assigns function description fields by using macrooptions

'This sub is here to be used as a guide if "TDfunctionsDescriptions" is not installed


'This function transforms epoch in miliseconds timestamp to excel date


Function MilistoDateFormat(mymilis As Double) 'transforms epoch in miliseconds timestamp to excel date


MilistoDateFormat = Format((mymilis / 86400) + 25569, "mm/dd/yyyy hh:mm:ss")

'divides the epoch timestamp by the number of miliseconds in a day and adds the days that passed

'from 1900, January 1st at 00:00 to 1970, January 1st at 00:00


End Function


'This function transforms epoch timestamp to excel date


Function EpochtoDateFormat(myepoch As Double) 'transforms epoch in seconds timestamp to excel date
```

```
EpochtoDateFormat = Format((myepoch / 86400) + 25569, "mm/dd/yyyy hh:mm:ss")

'divides the epoch timestamp by the number of seconds in a day and adds the days that passed
'from 1900, January 1st at 00:00 to 1970, January 1st at 00:00

'alternative not implemented: EpochtoDate = DateAdd("s", myepoch, "01/01/1970 00:00:00")

End Function

Private Sub FuncDesc(fname As String, fdescrip As String, fcateg As String, argdescrip() As String)

Application.MacroOptions _
    Macro:=fname, _
    Description:=fdescrip, _
    Category:=fcateg, _
    ArgumentDescriptions:=argdescrip

End Sub

' this function is a way to help users who don't have the module TDfunctionsDescriptions
```

```
Function HelpIOT() As String


HelpIOT = "Remember you can use 'Ctrl'+'Shift'+'A' to see what arguments are need in a function. Import 'TDfunctionsDescriptions' module to find the description of the functions in the Insert function tab."


End Function
```

```
''
'Library created by Pol Sarmiento Lozano  @TDdataprocess
'
'This library is to be used in the context of data management of IoT devices
'
'This library relates to the URL requests available in the device API of the TD SENSOR platform
'
'The URL requests where last checked to be available 2016, September 28th
'
'This library is better used in conjuction with "TDdeviceAPI"  "TDfunctionsDescriptions" and "JsonConverter"


Option Explicit 'forces explicit declaration of all the variables in the file


#If VBA7 Then ' supports Office 64-bit
    Private Declare PtrSafe Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef Source As Any, ByVal Length As LongPtr)

#Else ' supports Office 32-bit
    Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef Source As Any, ByVal Length As Long)
```

```
#End If

'This function allows the user to call the process main from a worksheet cell

Function ReportGeneration() As String

Call Main

ReportGeneration = "Done"

End Function

'this sub allows a user guided process of generating reports for IoT messages

Sub Main() 'main procedure for users that wish to not use the userform
Dim id As String 'module ID number
Dim key As String 'module key number
Dim n As Integer 'number of messages to be retrieved
Dim intn As Variant
Dim gmt As String 'date limit to be stablished
Dim wsname As String 'name of the report sheet
Dim reporttype As Integer 'type of report
Dim intreporttype As Variant
```

```
Dim starterexpression As String

Dim quotationmark As String


quotationmark = Chr$(34) ' Ascii for quotation mark


starterexpression = "To start the report generation program, six values are needed:" _

& vbCr & "1.The module ID number" & vbCr & "2.The module Key number" _

& vbCr & "3.The number of messages to retrieve" _

& vbCr & "4.The date limit" & vbCr & "5.The name of the worksheet" _

& vbCr & "6.The type of report" & vbCr & "Do you want to procede?"


If MsgBox(starterexpression, vbOKCancel) = vbOK Then 'guided start



   id = InputBox("Please insert module ID number" & vbCr & "without using quotation marks." _

   & vbCr & "e.g. E4A3", "1. Insert Module ID")


   'error handling

   If id = "" Then

      MsgBox ("You entered some invalid value!" & vbCr & "Please try again")

      Exit Sub

   ElseIf InStr(id, quotationmark) > 0 Then

      MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
```

```
    Exit Sub
Else
End If

key = InputBox("Please insert module Key number" & vbCr & "without using quotation marks." _
& vbCr & "e.g. CFE3A995", "2. Insert Module Key")

'error handling

If key = "" Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
ElseIf InStr(key, quotationmark) > 0 Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
Else
End If

intn = InputBox("How many messages do you want to retrieve?", "3. Number of messages", 20)

'error handling
```

```
If Not IsNumeric(intn) Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
ElseIf Not Round(intn, 0) / intn = 1 Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
Else
    n = CInt(intn)
End If

Select Case n
    Case 0
        MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
        Exit Sub
    Case Else
End Select

gmt = InputBox("Indicate until what day" _
& vbCr & "do you want to retrieve data. e.g. 18/08/2016 00:00:00" _
& vbCr & "as dd/mm/yyyy hh:mm:ss", "4. Date limit", Now())

'error handling
```

```
If gmt = "" Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
ElseIf InStr(gmt, quotationmark) > 0 Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
Else
End If

wsname = InputBox("How do you want to name the worksheet?" _
& vbCr & "e.g. Report1", "5. Destination Worksheet name")

'error handling

If wsname = "" Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
ElseIf InStr(wsname, quotationmark) > 0 Then
    MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
    Exit Sub
Else
End If
```

```
    intreporttype = InputBox("Thank you. If you want to Present Message Data insert 1." _
    & vbCr & "If you want to Present Full Data insert 2", "6. Report type", 1)

    'error handling

    If Not IsNumeric(intreporttype) Then
        MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
        Exit Sub
    Else
        reporttype = CInt(intreporttype)
    End If

    Select Case reporttype
        Case 1
        Case 2
        Case Else
            MsgBox ("You entered some invalid value!" & vbCr & "Please try again")
            Exit Sub
    End Select

    Call PresentData(id, key, n, gmt, wsname, reporttype)
    Exit Sub
Else
```

```
    Exit Sub
End If


End Sub



'This sub generates a report of the relevant data of Iot messages


Sub PresentData(id As String, key As String, n As Integer, gmt As String, wsname As String, _
reporttype As Integer)


Dim ws As Worksheet
Dim WorkSheetExists As Boolean
Dim response As String
Dim responseJson As Object
Dim rangedates As range
Dim i As Integer
Dim j As Integer
Dim k As Integer


'creation of new worsheet


WorkSheetExists = False
```

```
For Each ws In ThisWorkbook.Worksheets 'check if exists wsname
    If UCase(ws.Name) = UCase(wsname) Then
        WorkSheetExists = True
        Exit For
    End If
Next ws

If WorkSheetExists Then 'create worksheet
    MsgBox "Worksheet name already taken, please select another name.", vbExclamation
    wsname = InputBox("Select new name")

    For Each ws In ThisWorkbook.Worksheets 'check if exists wsname
    If UCase(ws.Name) = UCase(wsname) Then
        WorkSheetExists = True
        MsgBox "Please, make sure you choose an available name" _
        & vbCr & "for your worksheet and try again", vbCritical
        Exit Sub
    End If
    Next ws

    Sheets.Add.Name = wsname
Else
```

```
    Sheets.Add.Name = wsname
End If

Set ws = Worksheets(wsname)

'starting the process of data gathering
If MsgBox("Processing time may vary depending on" _
& vbCr & "amount of data present in the workbook" _
& vbCr & "or Internet connection." & vbCr & "Do you want to procede?", vbOKCancel + vbInformation, "Report Generation") = vbOK Then

    response = GetDataHistory(id, key, n, gmt)
    Set responseJson = JsonConverter.ParseJson(response) 'data parsing

    i = 1 'number of index in the json file
    k = 1 'number of child
    j = 1 'number of the column position in the excel grid

    For k = 1 To n 'loop through childs

        If ws.Cells(1, 1) = "" Then 'msg number header
            ws.Cells(1, 1) = "msg number"
        Else
        End If
```

```
    ws.Cells(k + 1, 1) = k 'adds msg number


    i = 1   'restart index and column position
    j = 2


    If reporttype = 3 Then 'restarts special case report 3
       reporttype = 2
    Else
    End If


    If responseJson("items")(k)("type") = "event:connectionok" And reporttype = 2 Then
'evaluates special case report 3
       reporttype = 3
    Else
    End If



  Do While i < 68


    If ws.Cells(1, j) = "" Then 'write header
       ws.Cells(1, j) = Fieldheader(i)
    Else
    End If
```

```
If arraylevel(i) = 1 Then 'write content
   ws.Cells(k + 1, j) = responseJson("items")(k)(Fieldname(i))
ElseIf arraylevel(i) = 2 Then
   ws.Cells(k + 1, j) = responseJson("items")(k)("ctxt")(Fieldname(i))
ElseIf arraylevel(i) = 3 Then
   ws.Cells(k + 1, j) = responseJson("items")(k)("ctxt")("position")(Fieldname(i))
ElseIf arraylevel(i) = 4 Then
   ws.Cells(k + 1, j) = responseJson("items")(k)("ctxt")(Fieldname(i))(0)
ElseIf arraylevel(i) = 5 Then
   ws.Cells(k + 1, i) = responseJson("items")(k)("contrib")(Fieldname(i))
ElseIf arraylevel(i) = 6 Then
   ws.Cells(k + 1, j) = responseJson("items")(k)("contrib")(Fieldname(i))(0)
ElseIf arraylevel(i) = 7 Then
   ws.Cells(k + 1, j) = responseJson("items")(k)(Fieldname(i))(0)
ElseIf arraylevel(i) = 8 Then
   ws.Cells(k + 1, j) = responseJson("items")(k)("extra")(Fieldname(i))
Else
   ws.Cells(k + 1, j) = "DisplayError" 'content error
   If MsgBox("Array level error", vbOKOnly) = vbOK Then
   Exit Sub
   Else
   End If
End If
```

```
    If reporttype = 3 And j = 9 Then
        j = j + 26
    Else
        j = j + 1 'increase column
    End If

    i = nextindex(i, reporttype) 'increase index with desired nextindex function

  Loop

  Next

  ws.Cells(n + 3, 1) = "Report Created " & Now() 'report creation time

  'format changes
  Call autosize

  Call formatcellstimedate(ws, rangedates, n)

    If MsgBox("Report Ready", vbOKOnly + vbInformation, "Report Information") = vbOK
Then 'conclude report message
    Else
    End If
```

```
Else
  Call Deleteworksheet(wsname)
  Exit Sub
End If

End Sub

'This sub changes the format of dates from epoch in miliseconds timestamp to "mm/dd/yy hh:mm:ss"

Private Sub formatcellstimedate(ws As Worksheet, rangedates As range, n As Integer) 'formats the cells in "mm/dd/yy hh:mm:ss"

Set rangedates = ws.range(Cells(2, 2), Cells(n + 1, 3))

   rangedates.NumberFormat = "General"

   For Each rangedates In ws.range(Cells(2, 2), Cells(n + 1, 3)).Cells

      rangedates.value = MilistoDate(rangedates.value)
      rangedates.NumberFormat = "mm/dd/yy hh:mm:ss"

   Next rangedates
```

```
End Sub

'This sub erasesData if given the admin pass

Sub EraseData(id, key, Adminpass As String)

    Call ClearData(id, key, Adminpass)

End Sub

'This function establishes the data indexes

Private Function nextindex(i As Integer, reporttype As Integer) As Integer 'index of  data

If reporttype = 1 Then 'message data

    If i = 1 Then
    nextindex = 2
    ElseIf i = 2 Then
    nextindex = 45
    ElseIf i = 45 Then
    nextindex = 46
    ElseIf i = 46 Then
```

```
    nextindex = 47
    ElseIf i >= 47 Then
    nextindex = i + 1
    Else
    nextindex = 500
    End If

ElseIf reporttype = 2 Then 'full data

    If i = 30 Then 'increase index, we dont need at the moment the array levels discarded
    nextindex = 36
    ElseIf i = 37 Then
    nextindex = 40
    ElseIf i = 41 Then
    nextindex = 44
    Else
    nextindex = i + 1
    End If

ElseIf reporttype = 3 Then

    If i = 8 Then 'increase index, we dont need at the moment the array levels discarded
    nextindex = 41
```

```
    ElseIf i = 41 Then
    nextindex = 44
    Else
    nextindex = i + 1
    End If

Else
MsgBox "Introduce an available mode and try again"
End If

End Function

'this function structures the jsonobject array levels

Private Function arraylevel(i As Integer) As Integer 'number of array structure type

Select Case i
  Case 1
  arraylevel = 1
  Case 2
  arraylevel = 1
  Case 3
  arraylevel = 1
```

Case 4

arraylevel = 1

Case 5

arraylevel = 1

Case 6

arraylevel = 1

Case 7

arraylevel = 1

Case 8

arraylevel = 1

Case 9

arraylevel = 2

Case 10

arraylevel = 2

Case 11

arraylevel = 2

Case 12

arraylevel = 2

Case 13

arraylevel = 2

Case 14

arraylevel = 2

Case 15

```
arraylevel = 2

Case 16

arraylevel = 2

Case 17

arraylevel = 2

Case 18

arraylevel = 2

Case 19

arraylevel = 2

Case 20

arraylevel = 2

Case 21

arraylevel = 2

Case 22

arraylevel = 2

Case 23

arraylevel = 2

Case 24

arraylevel = 2

Case 25

arraylevel = 2

Case 26

arraylevel = 2
```

Case 27

arraylevel = 2

Case 28

arraylevel = 2

Case 29

arraylevel = 2

Case 30

arraylevel = 2

Case 31

arraylevel = 3

Case 32

arraylevel = 3

Case 33

arraylevel = 3

Case 34

arraylevel = 3

Case 35

arraylevel = 3

Case 36

arraylevel = 2

Case 37

arraylevel = 2

Case 38

```
arraylevel = 4

Case 39

arraylevel = 4

Case 40

arraylevel = 2

Case 41

arraylevel = 5

Case 42

arraylevel = 6

Case 43

arraylevel = 6

Case 44

arraylevel = 7

Case 45

arraylevel = 1

Case 46

arraylevel = 1

Case 47

arraylevel = 8

Case 48

arraylevel = 8

Case 49

arraylevel = 8
```

Case 50

arraylevel = 8

Case 51

arraylevel = 8

Case 52

arraylevel = 8

Case 53

arraylevel = 8

Case 54

arraylevel = 8

Case 55

arraylevel = 8

Case 56

arraylevel = 8

Case 57

arraylevel = 8

Case 58

arraylevel = 8

Case 59

arraylevel = 8

Case 60

arraylevel = 8

Case 61

```
  arraylevel = 8
  Case 62
  arraylevel = 8
  Case 63
  arraylevel = 8
  Case 64
  arraylevel = 8
  Case 65
  arraylevel = 8
  Case 66
  arraylevel = 8
  Case 67
  arraylevel = 8
  Case Else
  arraylevel = 400
End Select

End Function

'This function establishes the headers for the messages to be retrieved

Private Function Fieldheader(indexfield As Integer) As String 'field names array of the retrievable data as presented in the json file
```

```
Select Case indexfield
  Case 1
  Fieldheader = "when"
  Case 2
  Fieldheader = "received"
  Case 3
  Fieldheader = "lvl"
  Case 4
  Fieldheader = "rssi"
  Case 5
  Fieldheader = "station"
  Case 6
  Fieldheader = "lat"
  Case 7
  Fieldheader = "lng"
  Case 8
  Fieldheader = "avglvl"
  Case 9
  Fieldheader = "ctxt.id"
  Case 10
  Fieldheader = "ctxt.module"
  Case 11
  Fieldheader = "ctxt.serial"
```

```
Case 12
Fieldheader = "ctxt.lastseen"
Case 13
Fieldheader = "ctxt.firstseen"
Case 14
Fieldheader = "ctxt.active"
Case 15
Fieldheader = "ctxt.watched"
Case 16
Fieldheader = "ctxt.type"
Case 17
Fieldheader = "ctxt.category"
Case 18
Fieldheader = "ctxt.index"
Case 19
Fieldheader = "ctxt.battery"
Case 20
Fieldheader = "ctxt.battery_value"
Case 21
Fieldheader = "ctxt.network"
Case 22
Fieldheader = "ctxt.tamper"
Case 23
```

```
Fieldheader = "ctxt.level"

Case 24

Fieldheader = "ctxt.temp"

Case 25

Fieldheader = "ctxt.temperature_value"

Case 26

Fieldheader = "ctxt.status"

Case 27

Fieldheader = "ctxt.msgs"

Case 28

Fieldheader = "ctxt.losts"

Case 29

Fieldheader = "ctxt.alerts"

Case 30

Fieldheader = "ctxt.alerts_ack"

Case 31

Fieldheader = "ctxt.position.latitude"

Case 32

Fieldheader = "ctxt.position.longitude"

Case 33

Fieldheader = "ctxt.position.altitude"

Case 34

Fieldheader = "ctxt.position.precision"
```

```
Case 35
Fieldheader = "ctxt.position.satellites"
Case 36
Fieldheader = "ctxt.keepalivefrequency"
Case 37
Fieldheader = "ctxt.isinstalled"
Case 38
Fieldheader = "ctxt.parentOrganisationApps.()" '()
Case 39
Fieldheader = "ctxt.organisationApps.()" '()
Case 40
Fieldheader = "ctxt.deviceClass"
Case 41
Fieldheader = "contrib.isinstalled"
Case 42
Fieldheader = "contrib.parentOrganisationApps" '()
Case 43
Fieldheader = "contrib.organisationApps" '()
Case 44
Fieldheader = "custom"
Case 45
Fieldheader = "type" 'all
Case 46
```

Fieldheader = "payload" 'all

Case 47

Fieldheader = "extra.message" 'all

Case 48

Fieldheader = "extra.entry_id" 'all

Case 49

Fieldheader = "extra.frame_type" 'all

Case 50

Fieldheader = "extra.type" 'all almost

Case 51

Fieldheader = "extra.raw" 'raw

Case 52

Fieldheader = "extra.asciiraw" 'raw

Case 53

Fieldheader = "extra.asciidata" 'sms

Case 54

Fieldheader = "extra.data" 'event and sms

Case 55

Fieldheader = "extra.mobile" 'phone

Case 56

Fieldheader = "extra.value_id" 'phone

Case 57

Fieldheader = "extra.altitude" 'gps

```
Case 58
Fieldheader = "extra.quality" 'gps
Case 59
Fieldheader = "extra.longitude" 'gps
Case 60
Fieldheader = "extra.latitude" 'gps
Case 61
Fieldheader = "extra.satelliteinview" 'gps
Case 62
Fieldheader = "extra.volt" 'ka
Case 63
Fieldheader = "extra.temp" 'ka
Case 64
Fieldheader = "extra.frequency" 'ka
Case 65
Fieldheader = "extra.code" 'event
Case 66
Fieldheader = "extra.voltage" 'battery
Case 67
Fieldheader = "extra.temperature" 'temperature

Case Else
Fieldheader = "ERRORINFIELDNAME"
```

```
End Select

End Function

'This function establishes the fieldnames of the json objects

Private Function Fieldname(indexfield As Integer) As String 'string array of the data parameters needed to retrieve data

Select Case indexfield
  Case 1
  Fieldname = "when"
  Case 2
  Fieldname = "received"
  Case 3
  Fieldname = "lvl"
  Case 4
  Fieldname = "rssi"
  Case 5
  Fieldname = "station"
  Case 6
  Fieldname = "lat"
  Case 7
  Fieldname = "lng"
```

```
Case 8
Fieldname = "avglvl"
Case 9
Fieldname = "id"
Case 10
Fieldname = "module"
Case 11
Fieldname = "serial"
Case 12
Fieldname = "lastseen"
Case 13
Fieldname = "firstseen"
Case 14
Fieldname = "active"
Case 15
Fieldname = "watched"
Case 16
Fieldname = "type"
Case 17
Fieldname = "category"
Case 18
Fieldname = "index"
Case 19
```

Fieldname = "battery"

Case 20

Fieldname = "battery_value"

Case 21

Fieldname = "network"

Case 22

Fieldname = "tamper"

Case 23

Fieldname = "level"

Case 24

Fieldname = "temp"

Case 25

Fieldname = "temperature_value"

Case 26

Fieldname = "status"

Case 27

Fieldname = "msgs"

Case 28

Fieldname = "losts"

Case 29

Fieldname = "alerts"

Case 30

Fieldname = "alerts_ack"

```
Case 31
Fieldname = "latitude"
Case 32
Fieldname = "longitude"
Case 33
Fieldname = "altitude"
Case 34
Fieldname = "precision"
Case 35
Fieldname = "satellites"
Case 36
Fieldname = "keepalivefrequency"
Case 37
Fieldname = "isinstalled"
Case 38
Fieldname = "parentOrganisationApps" '()
Case 39
Fieldname = "organisationApps" '()
Case 40
Fieldname = "deviceClass"
Case 41
Fieldname = "isinstalled"
Case 42
```

Fieldname = "parentOrganisationApps" '()

Case 43

Fieldname = "organisationApps" '()

Case 44

Fieldname = "custom" '()

Case 45

Fieldname = "type" 'all

Case 46

Fieldname = "payload" 'all

Case 47

Fieldname = "message" 'all

Case 48

Fieldname = "entry_id" 'all

Case 49

Fieldname = "frame_type" 'all

Case 50

Fieldname = "type" 'all almost

Case 51

Fieldname = "raw" 'raw

Case 52

Fieldname = "asciiraw" 'raw

Case 53

Fieldname = "asciidata" 'sms

```
Case 54
Fieldname = "data" 'event and sms
Case 55
Fieldname = "mobile" 'phone
Case 56
Fieldname = "value_id" 'phone
Case 57
Fieldname = "altitude" 'gps
Case 58
Fieldname = "quality" 'gps
Case 59
Fieldname = "longitude" 'gps
Case 60
Fieldname = "latitude" 'gps
Case 61
Fieldname = "satelliteinview" 'gps
Case 62
Fieldname = "volt" 'ka
Case 63
Fieldname = "temp" 'ka
Case 64
Fieldname = "frequency" 'ka
Case 65
```

```
  Fieldname = "code" 'event
  Case 66
  Fieldname = "voltage" 'battery
  Case 67
  Fieldname = "temperature" 'temperature
  Case Else
  Fieldname = "ERRORINFIELDNAME"
End Select

End Function

'This sub deletes worksheets

Private Sub Deleteworksheet(ByRef wsname As String) 'deletes worksheets without displaying annoying alert

    Application.DisplayAlerts = False
    Sheets(wsname).Delete
    Application.DisplayAlerts = True

End Sub

'This function transforms epoch in miliseconds timestamp to excel date
```

```
Function MilistoDate(mymilis As Double) 'transforms epoch in miliseconds timestamp to excel date

MilistoDate = (mymilis / 86400000) + 25569

'divides the epoch timestamp by the number of miliseconds in a day and adds the days that passed

'from 1900, January 1st at 00:00 to 1970, January 1st at 00:00

End Function

'This function transforms epoch timestamp to excel date

Function EpochtoDate(myepoch As Double) 'transforms epoch in seconds timestamp to excel date

EpochtoDate = (myepoch / 86400) + 25569

'divides the epoch timestamp by the number of seconds in a day and adds the days that passed

'from 1900, January 1st at 00:00 to 1970, January 1st at 00:00

End Function

'This function autosizes the active sheet columns
```

```
Private Sub autosize() 'autosizes the columns in the active sheet to properly display the data

ActiveSheet.Columns.AutoFit

End Sub
```

```
''
'Library created by Pol Sarmiento Lozano @TDfunctionsDescription
'
'This library is to be used in the context of data management of IoT devices
'
'This library relates to the URL requests available in the device API of the TD SENSOR platform
'
'The URL requests where last checked to be available 2016, September 28th
'
'This library is better used in conjuction with  "TDdataprocess" "TDdeviceAPI" and "JsonConverter"



Option Explicit 'forces explicit declaration of all the variables in the file



#If VBA7 Then 'Office 64-bit
    Private Declare PtrSafe Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef Source As Any, ByVal Length As LongPtr)


#Else ' Office 32-bit
    Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef Source As Any, ByVal Length As Long)
#End If
```

```
'This function adds the function descriptions to the insert functions database from the worksheet

Function HelpIOTDescriptions() As String

Call FunctionDescriptionImplem

HelpIOTDescriptions = "Check the insert function button." _
& vbCr & "You will find the IoT related functions in the IoT category"

End Function

'this process is used inside function HelpIot

Sub FunctionDescriptionImplem()

Call DGetpayload
Call DGetTypeMessage
Call DGetAsciiMessage
Call DGetFieldData
Call DGetEncodedMessage
Call DGetCoordinates
```

```
Call DGetDataHistory

Call DGetToken

Call DGetPAC

Call DGetLastSeen

Call DGetFirstSeen

Call DGetMessageSent

Call DGetMessageReceived

Call DGetTravelTimeMessage

Call DGetCurrentMessagesServer

Call DGetTotalMessagesServer

Call DGetChildren

Call DClearData

Call DChangeDfuncstat

Call DChangeBidir

Call DToggleactiveflag

Call DTogglemonitorflag

Call DMilistoDateFormat

Call DEpochtoDateFormat


End Sub


'get payload info
```

```
Private Sub DGetpayload()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetPayloadMessage"

fdescrip = "Obtains the payload of the 'm' message of a list of last 'n' messages"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub



'get type of message info
```

```
Private Sub DGetTypeMessage()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetTypeMessage"

fdescrip = "Obtains the type of message of the 'm' message of a list of last 'n' messages"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub



'get ascii message info
```

```
Private Sub DGetAsciiMessage()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetAsciiMessage"

fdescrip = "Obtains the Ascii conversion of the 'm' message of a list of last 'n' messages"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub



'get field data info
```

```
Private Sub DGetFieldData()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetFieldData"

fdescrip = "Obtains the data of the field data contained on the 'm' message of a list of last 'n' messages"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub



'get encoded message info
```

```
Private Sub DGetEncodedMessage()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetEncodedMessage"

fdescrip = "Obtains the encoded  'm' message of a list of last 'n' messages"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub



'get coordinates info
```

```
Private Sub DGetCoordinates()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetCoordinates"

fdescrip = "Obtains the latitude and longitude of the 'm' message of a list of last 'n' messages. Only available if GPS antenna connected"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub
```

```
'get data history info

Private Sub DGetDataHistory()
Dim fname As String
Dim fdescrip As String
Dim fcateg As String
Dim argdescrip(1 To 4) As String

fname = "GetDataHistory"
fdescrip = "Obtains the data history of 'n' messages in format JSON"
fcateg = "TD IoT"
argdescrip(1) = "Module's ID number inside brackets."
argdescrip(2) = "Module's Key number inside brackets."
argdescrip(3) = "Number of messages to retrieve. E.g.: 1"
argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip

End Sub

'get GetToken info
```

```
Private Sub DGetToken()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 2) As String


fname = "GetToken"

fdescrip = "Obtains the authentication token required to make HTTP requests to SENSOR"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg,
ArgumentDescriptions:=argdescrip


End Sub


' get PAC key


Private Sub DGetPAC()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 2) As String
```

```
fname = "GetPAC"

fdescrip = "Obtains the 16 digit PAC key of your module"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get lastseen message info


Private Sub DGetLastSeen()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetLastSeen"

fdescrip = "Obtains the last date the device was operational"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."
```

```
argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get firstseen message info


Private Sub DGetFirstSeen()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetFirstSeen"

fdescrip = "Obtains the first date the device was operational"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."
```

```
argdescrip(3) = "Number of messages to retrieve. E.g.: 1"

argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get sent message info


Private Sub DGetMessageSent()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetMessageSent"

fdescrip = "Obtains the date the message was sent by the device"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"
```

```
argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get received message info


Private Sub DGetMessageReceived()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetMessageReceived"

fdescrip = "Obtains the date the message was received by SENSOR"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"
```

```
argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get received-sent difference message info


Private Sub DGetTravelTimeMessage()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 5) As String


fname = "GetTravelTimeMessage"

fdescrip = "Obtains the time difference between the date the message was received by SENSOR and the date the message was sent by the device"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Number of messages to retrieve. E.g.: 1"
```

```
argdescrip(4) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "

argdescrip(5) = "The number of the message you want to show. E.g.: 1"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get current number messages on server message info


Private Sub DGetCurrentMessagesServer()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 3) As String


fname = "GetCurrentMessagesServer"

fdescrip = "Obtains current number of messages available at the server sent with this device"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "
```

```
Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get total number messages on server message info


Private Sub DGetTotalMessagesServer()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 3) As String


fname = "GetTotalMessagesServer"

fdescrip = "Obtains the total number of messages sent to the server with this device"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Excel timestamp of GMT time in 'dd/mm/yyyy hh:mm:ss' format E.g.: 18/08/2016 00:00:00 "


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip
```

```
End Sub

'get GetChildren info

Private Sub DGetChildren()
Dim fname As String
Dim fdescrip As String
Dim fcateg As String
Dim argdescrip(1 To 2) As String

fname = "GetChildren"
fdescrip = "Obtains the children information of a gateway module"
fcateg = "TD IoT"
argdescrip(1) = "Module's ID number inside brackets."
argdescrip(2) = "Module's Key number inside brackets."

Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg,
ArgumentDescriptions:=argdescrip

End Sub

'get clear data info
```

```
Private Sub DClearData()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 3) As String


fname = "ClearData"

fdescrip = "Obtains the children information of a gateway module"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "Administrator password."


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get ChangeDfuncstat info


Private Sub DChangeDfuncstat()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String
```

```
Dim argdescrip(1 To 4) As String

fname = "ChangeDfuncstat"
fdescrip = "Obtains the children information of a gateway module"
fcateg = "TD IoT"
argdescrip(1) = "Module's ID number inside brackets."
argdescrip(2) = "Module's Key number inside brackets."
argdescrip(3) = "New functional status value of the device."
argdescrip(4) = "Administrator password."

Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip

End Sub

'get Toggleactiveflag info

Private Sub DToggleactiveflag()
Dim fname As String
Dim fdescrip As String
Dim fcateg As String
Dim argdescrip(1 To 4) As String

fname = "Toggleactiveflag"
```

```
fdescrip = "This function toggles the active flag value"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "New active flag value of the device."

argdescrip(4) = "Administrator password."


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get Togglemonitorflag info


Private Sub DTogglemonitorflag()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 4) As String


fname = "Togglemonitorflag"

fdescrip = "This function toggles the monitoring flag value of the device"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."
```

```
argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "New monitoring flag value of the device."

argdescrip(4) = "Administrator password."


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub


'get ChangeBidir info


Private Sub DChangeBidir()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1 To 4) As String


fname = "ChangeBidir"

fdescrip = "This function changes the bidir value of the device"

fcateg = "TD IoT"

argdescrip(1) = "Module's ID number inside brackets."

argdescrip(2) = "Module's Key number inside brackets."

argdescrip(3) = "New Bidirectional value of the device."

argdescrip(4) = "Administrator password."
```

```
Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip

End Sub

'get MilistoDateFormat info

Private Sub DMilistoDateFormat()
Dim fname As String
Dim fdescrip As String
Dim fcateg As String
Dim argdescrip(1) As String

fname = "MilistoDateFormat"
fdescrip = "This function transforms epoch in miliseconds timestamp to excel date"
fcateg = "TD IoT"
argdescrip(1) = "Epoch timestamp with miliseconds"

Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip

End Sub
```

```
'get EpochtoDateFormat info


Private Sub DEpochtoDateFormat()

Dim fname As String

Dim fdescrip As String

Dim fcateg As String

Dim argdescrip(1) As String


fname = "EpochtoDateFormat"

fdescrip = "This function transforms epoch timestamp to excel date"

fcateg = "TD IoT"

argdescrip(1) = "Epoch timestamp"


Application.MacroOptions Macro:=fname, Description:=fdescrip, Category:=fcateg, ArgumentDescriptions:=argdescrip


End Sub
```

```
''
' VBA-JSON v2.2.1
' (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.
'
```

```
' (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

' SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

'
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~ '


' === VBA-UTC Headers

#If Mac Then


Private Declare Function utc_popen Lib "libc.dylib" Alias "popen" (ByVal utc_Command As String, ByVal utc_Mode As String) As Long

Private Declare Function utc_pclose Lib "libc.dylib" Alias "pclose" (ByVal utc_File As Long) As Long

Private Declare Function utc_fread Lib "libc.dylib" Alias "fread" (ByVal utc_Buffer As String, ByVal utc_Size As Long, ByVal utc_Number As Long, ByVal utc_File As Long) As Long

Private Declare Function utc_feof Lib "libc.dylib" Alias "feof" (ByVal utc_File As Long) As Long


#ElseIf VBA7 Then


' http://msdn.microsoft.com/en-us/library/windows/desktop/ms724421.aspx

' http://msdn.microsoft.com/en-us/library/windows/desktop/ms724949.aspx

' http://msdn.microsoft.com/en-us/library/windows/desktop/ms725485.aspx

Private Declare PtrSafe Function utc_GetTimeZoneInformation Lib "kernel32" Alias "GetTimeZoneInformation" _
```

```
    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION) As Long

Private Declare PtrSafe Function utc_SystemTimeToTzSpecificLocalTime Lib "kernel32" Alias "SystemTimeToTzSpecificLocalTime" _

    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpUniversalTime As utc_SYSTEMTIME, utc_lpLocalTime As utc_SYSTEMTIME) As Long

Private Declare PtrSafe Function utc_TzSpecificLocalTimeToSystemTime Lib "kernel32" Alias "TzSpecificLocalTimeToSystemTime" _

    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpLocalTime As utc_SYSTEMTIME, utc_lpUniversalTime As utc_SYSTEMTIME) As Long


#Else


Private Declare Function utc_GetTimeZoneInformation Lib "kernel32" Alias "GetTimeZoneInformation" _

    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION) As Long

Private Declare Function utc_SystemTimeToTzSpecificLocalTime Lib "kernel32" Alias "SystemTimeToTzSpecificLocalTime" _

    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpUniversalTime As utc_SYSTEMTIME, utc_lpLocalTime As utc_SYSTEMTIME) As Long

Private Declare Function utc_TzSpecificLocalTimeToSystemTime Lib "kernel32" Alias "TzSpecificLocalTimeToSystemTime" _

    (utc_lpTimeZoneInformation As utc_TIME_ZONE_INFORMATION, utc_lpLocalTime As utc_SYSTEMTIME, utc_lpUniversalTime As utc_SYSTEMTIME) As Long


#End If


#If Mac Then
```

```
Private Type utc_ShellResult
    utc_Output As String
    utc_ExitCode As Long
End Type

#Else

Private Type utc_SYSTEMTIME
    utc_wYear As Integer
    utc_wMonth As Integer
    utc_wDayOfWeek As Integer
    utc_wDay As Integer
    utc_wHour As Integer
    utc_wMinute As Integer
    utc_wSecond As Integer
    utc_wMilliseconds As Integer
End Type

Private Type utc_TIME_ZONE_INFORMATION
    utc_Bias As Long
    utc_StandardName(0 To 31) As Integer
    utc_StandardDate As utc_SYSTEMTIME
```

```
    utc_StandardBias As Long
    utc_DaylightName(0 To 31) As Integer
    utc_DaylightDate As utc_SYSTEMTIME
    utc_DaylightBias As Long
End Type

#End If
' === End VBA-UTC

#If Mac Then
#ElseIf VBA7 Then

Private Declare PtrSafe Sub json_CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
    (json_MemoryDestination As Any, json_MemorySource As Any, ByVal json_ByteLength
As Long)

#Else

Private Declare Sub json_CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
    (json_MemoryDestination As Any, json_MemorySource As Any, ByVal json_ByteLength
As Long)

#End If
```

```
Private Type json_Options

    ' VBA only stores 15 significant digits, so any numbers larger than that are truncated

    ' 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

    ' See: http://support.microsoft.com/kb/269370

    '

    ' By default, VBA-JSON will use String for numbers longer than 15 characters that contain only digits

    ' to override set `JsonConverter.JsonOptions.UseDoubleForLargeNumbers = True`

    UseDoubleForLargeNumbers As Boolean


    ' The JSON standard requires object keys to be quoted (" or '), use this option to allow unquoted keys

    AllowUnquotedKeys As Boolean


    ' The solidus (/) is not required to be escaped, use this option to escape them as \/ in ConvertToJson

    EscapeSolidus As Boolean

End Type

Public JsonOptions As json_Options


' ============================================= '

' Public Methods

' ============================================= '
```

```
'
' Convert JSON string to object (Dictionary/Collection)
'
' @method ParseJson
' @param {String} json_String
' @return {Object} (Dictionary or Collection)
' @throws 10001 - JSON parse error
'
Public Function ParseJson(ByVal JsonString As String) As Object
    Dim json_Index As Long
    json_Index = 1

    ' Remove vbCr, vbLf, and vbTab from json_String
    JsonString = VBA.Replace(VBA.Replace(VBA.Replace(JsonString, VBA.vbCr, ""), VBA.vbLf, ""), VBA.vbTab, "")

    json_SkipSpaces JsonString, json_Index
    Select Case VBA.Mid$(JsonString, json_Index, 1)
    Case "{"
        Set ParseJson = json_ParseObject(JsonString, json_Index)
    Case "["
        Set ParseJson = json_ParseArray(JsonString, json_Index)
    Case Else
        ' Error: Invalid JSON string
```

```
        Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(JsonString, json_Index,
"Expecting '{' or '['")

    End Select

End Function



''

' Convert object (Dictionary/Collection/Array) to JSON

'

' @method ConvertToJson

' @param {Variant} JsonValue (Dictionary, Collection, or Array)

' @param {Integer|String} Whitespace "Pretty" print json with given number of spaces per
indentation (Integer) or given string

' @return {String}

''

Public Function ConvertToJson(ByVal JsonValue As Variant, Optional ByVal Whitespace
As Variant, Optional ByVal json_CurrentIndentation As Long = 0) As String

    Dim json_buffer As String

    Dim json_BufferPosition As Long

    Dim json_BufferLength As Long

    Dim json_Index As Long

    Dim json_LBound As Long

    Dim json_UBound As Long

    Dim json_IsFirstItem As Boolean

    Dim json_Index2D As Long
```

```
    Dim json_LBound2D As Long
    Dim json_UBound2D As Long
    Dim json_IsFirstItem2D As Boolean
    Dim json_Key As Variant
    Dim json_Value As Variant
    Dim json_DateStr As String
    Dim json_Converted As String
    Dim json_SkipItem As Boolean
    Dim json_PrettyPrint As Boolean
    Dim json_Indentation As String
    Dim json_InnerIndentation As String

    json_LBound = -1
    json_UBound = -1
    json_IsFirstItem = True
    json_LBound2D = -1
    json_UBound2D = -1
    json_IsFirstItem2D = True
    json_PrettyPrint = Not IsMissing(Whitespace)

    Select Case VBA.VarType(JsonValue)
    Case VBA.vbNull
        ConvertToJson = "null"
```

```
    Case VBA.vbDate
        ' Date
        json_DateStr = ConvertToIso(VBA.CDate(JsonValue))

        ConvertToJson = """" & json_DateStr & """"
    Case VBA.vbString
        ' String (or large number encoded as string)
        If Not JsonOptions.UseDoubleForLargeNumbers And json_StringIsLargeNumber(JsonValue) Then
            ConvertToJson = JsonValue
        Else
            ConvertToJson = """" & json_Encode(JsonValue) & """"
        End If
    Case VBA.vbBoolean
        If JsonValue Then
            ConvertToJson = "true"
        Else
            ConvertToJson = "false"
        End If
    Case VBA.vbArray To VBA.vbArray + VBA.vbByte
        If json_PrettyPrint Then
            If VBA.VarType(Whitespace) = VBA.vbString Then
                json_Indentation = VBA.String$(json_CurrentIndentation + 1, Whitespace)
                json_InnerIndentation = VBA.String$(json_CurrentIndentation + 2, Whitespace)
```

```
        Else
            json_Indentation = VBA.Space$((json_CurrentIndentation + 1) * Whitespace)
            json_InnerIndentation = VBA.Space$((json_CurrentIndentation + 2) * Whitespace)
        End If
    End If

    ' Array
    json_BufferAppend json_buffer, "[", json_BufferPosition, json_BufferLength

    On Error Resume Next

    json_LBound = LBound(JsonValue, 1)
    json_UBound = UBound(JsonValue, 1)
    json_LBound2D = LBound(JsonValue, 2)
    json_UBound2D = UBound(JsonValue, 2)

    If json_LBound >= 0 And json_UBound >= 0 Then
        For json_Index = json_LBound To json_UBound
            If json_IsFirstItem Then
                json_IsFirstItem = False
            Else
                ' Append comma to previous line
                json_BufferAppend json_buffer, ",", json_BufferPosition, json_BufferLength
```

```
        End If

        If json_LBound2D >= 0 And json_UBound2D >= 0 Then
            ' 2D Array
            If json_PrettyPrint Then
                json_BufferAppend json_buffer, vbNewLine, json_BufferPosition, json_BufferLength
            End If
            json_BufferAppend json_buffer, json_Indentation & "[", json_BufferPosition, json_BufferLength

            For json_Index2D = json_LBound2D To json_UBound2D
                If json_IsFirstItem2D Then
                    json_IsFirstItem2D = False
                Else
                    json_BufferAppend json_buffer, ",", json_BufferPosition, json_BufferLength
                End If

                json_Converted = ConvertToJson(JsonValue(json_Index, json_Index2D), Whitespace, json_CurrentIndentation + 2)

                ' For Arrays/Collections, undefined (Empty/Nothing) is treated as null
                If json_Converted = "" Then
                    ' (nest to only check if converted = "")
```

```
                If json_IsUndefined(JsonValue(json_Index, json_Index2D)) Then
                    json_Converted = "null"
                End If
            End If

            If json_PrettyPrint Then
                json_Converted = vbNewLine & json_InnerIndentation & json_Converted
            End If

            json_BufferAppend json_buffer, json_Converted, json_BufferPosition, json_BufferLength
        Next json_Index2D

        If json_PrettyPrint Then
            json_BufferAppend json_buffer, vbNewLine, json_BufferPosition, json_BufferLength
        End If

        json_BufferAppend json_buffer, json_Indentation & "]", json_BufferPosition, json_BufferLength
        json_IsFirstItem2D = True
    Else
        ' 1D Array
        json_Converted = ConvertToJson(JsonValue(json_Index), Whitespace, json_CurrentIndentation + 1)
```

```
                ' For Arrays/Collections, undefined (Empty/Nothing) is treated as null
                If json_Converted = "" Then
                    ' (nest to only check if converted = "")
                    If json_IsUndefined(JsonValue(json_Index)) Then
                        json_Converted = "null"
                    End If
                End If

                If json_PrettyPrint Then
                    json_Converted = vbNewLine & json_Indentation & json_Converted
                End If

                json_BufferAppend json_buffer, json_Converted, json_BufferPosition, json_BufferLength
            End If
        Next json_Index
    End If

    On Error GoTo 0

    If json_PrettyPrint Then
        json_BufferAppend json_buffer, vbNewLine, json_BufferPosition, json_BufferLength
```

```
        If VBA.VarType(Whitespace) = VBA.vbString Then
            json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
        Else
            json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
        End If
    End If

    json_BufferAppend json_buffer, json_Indentation & "]", json_BufferPosition, json_BufferLength

    ConvertToJson = json_BufferToString(json_buffer, json_BufferPosition, json_BufferLength)

  ' Dictionary or Collection
  Case VBA.vbObject
    If json_PrettyPrint Then
        If VBA.VarType(Whitespace) = VBA.vbString Then
            json_Indentation = VBA.String$(json_CurrentIndentation + 1, Whitespace)
        Else
            json_Indentation = VBA.Space$((json_CurrentIndentation + 1) * Whitespace)
        End If
    End If

    ' Dictionary
```

```
        If VBA.TypeName(JsonValue) = "Dictionary" Then
            json_BufferAppend json_buffer, "{", json_BufferPosition, json_BufferLength
            For Each json_Key In JsonValue.Keys
                ' For Objects, undefined (Empty/Nothing) is not added to object
                json_Converted = ConvertToJson(JsonValue(json_Key), Whitespace, json_CurrentIndentation + 1)
                If json_Converted = "" Then
                    json_SkipItem = json_IsUndefined(JsonValue(json_Key))
                Else
                    json_SkipItem = False
                End If

                If Not json_SkipItem Then
                    If json_IsFirstItem Then
                        json_IsFirstItem = False
                    Else
                        json_BufferAppend json_buffer, ",", json_BufferPosition, json_BufferLength
                    End If

                    If json_PrettyPrint Then
                        json_Converted = vbNewLine & json_Indentation & """" & json_Key & """: " & json_Converted
                    Else
                        json_Converted = """" & json_Key & """:" & json_Converted
```

```
            End If

            json_BufferAppend json_buffer, json_Converted, json_BufferPosition, json_BufferLength
        End If
    Next json_Key

    If json_PrettyPrint Then
        json_BufferAppend json_buffer, vbNewLine, json_BufferPosition, json_BufferLength

        If VBA.VarType(Whitespace) = VBA.vbString Then
            json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
        Else
            json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
        End If
    End If

    json_BufferAppend json_buffer, json_Indentation & "}", json_BufferPosition, json_BufferLength

' Collection
ElseIf VBA.TypeName(JsonValue) = "Collection" Then
    json_BufferAppend json_buffer, "[", json_BufferPosition, json_BufferLength
```

```
        For Each json_Value In JsonValue
            If json_IsFirstItem Then
                json_IsFirstItem = False
            Else
                json_BufferAppend json_buffer, ",", json_BufferPosition, json_BufferLength
            End If

            json_Converted = ConvertToJson(json_Value, Whitespace, json_CurrentIndentation + 1)

            ' For Arrays/Collections, undefined (Empty/Nothing) is treated as null
            If json_Converted = "" Then
                ' (nest to only check if converted = "")
                If json_IsUndefined(json_Value) Then
                    json_Converted = "null"
                End If
            End If

            If json_PrettyPrint Then
                json_Converted = vbNewLine & json_Indentation & json_Converted
            End If

            json_BufferAppend json_buffer, json_Converted, json_BufferPosition, json_BufferLength
```

```
        Next json_Value

        If json_PrettyPrint Then
            json_BufferAppend json_buffer, vbNewLine, json_BufferPosition, json_BufferLength

            If VBA.VarType(Whitespace) = VBA.vbString Then
                json_Indentation = VBA.String$(json_CurrentIndentation, Whitespace)
            Else
                json_Indentation = VBA.Space$(json_CurrentIndentation * Whitespace)
            End If
        End If

        json_BufferAppend json_buffer, json_Indentation & "]", json_BufferPosition, json_BufferLength
    End If

    ConvertToJson = json_BufferToString(json_buffer, json_BufferPosition, json_BufferLength)
  Case VBA.vbInteger, VBA.vbLong, VBA.vbSingle, VBA.vbDouble, VBA.vbCurrency, VBA.vbDecimal
    ' Number (use decimals for numbers)
    ConvertToJson = VBA.Replace(JsonValue, ",", ".")
  Case Else
    ' vbEmpty, vbError, vbDataObject, vbByte, vbUserDefinedType
```

```
        ' Use VBA's built-in to-string
        On Error Resume Next
        ConvertToJson = JsonValue
        On Error GoTo 0
    End Select
End Function

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

Private Function json_ParseObject(json_String As String, ByRef json_Index As Long) As Dictionary
    Dim json_Key As String
    Dim json_NextChar As String

    Set json_ParseObject = New Dictionary
    json_SkipSpaces json_String, json_Index
    If VBA.Mid$(json_String, json_Index, 1) <> "{" Then
        Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(json_String, json_Index, "Expecting '{'")
    Else
        json_Index = json_Index + 1
```

```
        Do
            json_SkipSpaces json_String, json_Index
            If VBA.Mid$(json_String, json_Index, 1) = "}" Then
                json_Index = json_Index + 1
                Exit Function
            ElseIf VBA.Mid$(json_String, json_Index, 1) = "," Then
                json_Index = json_Index + 1
                json_SkipSpaces json_String, json_Index
            End If

            json_Key = json_ParseKey(json_String, json_Index)
            json_NextChar = json_Peek(json_String, json_Index)
            If json_NextChar = "[" Or json_NextChar = "{" Then
                Set json_ParseObject.Item(json_Key) = json_ParseValue(json_String,
json_Index)
            Else
                json_ParseObject.Item(json_Key) = json_ParseValue(json_String, json_Index)
            End If
        Loop
    End If
End Function

Private Function json_ParseArray(json_String As String, ByRef json_Index As Long) As
Collection
```

```
    Set json_ParseArray = New Collection


    json_SkipSpaces json_String, json_Index
    If VBA.Mid$(json_String, json_Index, 1) <> "[" Then
        Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(json_String, json_Index,
"Expecting '['")
    Else
        json_Index = json_Index + 1


        Do
            json_SkipSpaces json_String, json_Index
            If VBA.Mid$(json_String, json_Index, 1) = "]" Then
                json_Index = json_Index + 1
                Exit Function
            ElseIf VBA.Mid$(json_String, json_Index, 1) = "," Then
                json_Index = json_Index + 1
                json_SkipSpaces json_String, json_Index
            End If


            json_ParseArray.Add json_ParseValue(json_String, json_Index)
        Loop
    End If
End Function
```

```
Private Function json_ParseValue(json_String As String, ByRef json_Index As Long) As Variant
    json_SkipSpaces json_String, json_Index
    Select Case VBA.Mid$(json_String, json_Index, 1)
    Case "{"
        Set json_ParseValue = json_ParseObject(json_String, json_Index)
    Case "["
        Set json_ParseValue = json_ParseArray(json_String, json_Index)
    Case """", "'"
        json_ParseValue = json_ParseString(json_String, json_Index)
    Case Else
        If VBA.Mid$(json_String, json_Index, 4) = "true" Then
            json_ParseValue = True
            json_Index = json_Index + 4
        ElseIf VBA.Mid$(json_String, json_Index, 5) = "false" Then
            json_ParseValue = False
            json_Index = json_Index + 5
        ElseIf VBA.Mid$(json_String, json_Index, 4) = "null" Then
            json_ParseValue = Null
            json_Index = json_Index + 4
        ElseIf VBA.InStr("+-0123456789", VBA.Mid$(json_String, json_Index, 1)) Then
            json_ParseValue = json_ParseNumber(json_String, json_Index)
        Else
```

```
            Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(json_String, json_Index, "Expecting 'STRING', 'NUMBER', null, true, false, '{', or '['")
        End If
    End Select
End Function

Private Function json_ParseString(json_String As String, ByRef json_Index As Long) As String
    Dim json_Quote As String
    Dim json_Char As String
    Dim json_Code As String
    Dim json_buffer As String
    Dim json_BufferPosition As Long
    Dim json_BufferLength As Long

    json_SkipSpaces json_String, json_Index

    ' Store opening quote to look for matching closing quote
    json_Quote = VBA.Mid$(json_String, json_Index, 1)
    json_Index = json_Index + 1

    Do While json_Index > 0 And json_Index <= Len(json_String)
        json_Char = VBA.Mid$(json_String, json_Index, 1)
```

```
        Select Case json_Char
        Case "\"
            ' Escaped string, \\, or \/
            json_Index = json_Index + 1
            json_Char = VBA.Mid$(json_String, json_Index, 1)

            Select Case json_Char
            Case """", "\", "/", "'"
                json_BufferAppend json_buffer, json_Char, json_BufferPosition, json_BufferLength
                json_Index = json_Index + 1
            Case "b"
                json_BufferAppend json_buffer, vbBack, json_BufferPosition, json_BufferLength
                json_Index = json_Index + 1
            Case "f"
                json_BufferAppend json_buffer, vbFormFeed, json_BufferPosition, json_BufferLength
                json_Index = json_Index + 1
            Case "n"
                json_BufferAppend json_buffer, vbCrLf, json_BufferPosition, json_BufferLength
                json_Index = json_Index + 1
            Case "r"
                json_BufferAppend json_buffer, vbCr, json_BufferPosition, json_BufferLength
                json_Index = json_Index + 1
```

```
            Case "t"
                json_BufferAppend json_buffer, vbTab, json_BufferPosition, json_BufferLength
                json_Index = json_Index + 1
            Case "u"
                ' Unicode character escape (e.g. \u00a9 = Copyright)
                json_Index = json_Index + 1
                json_Code = VBA.Mid$(json_String, json_Index, 4)
                json_BufferAppend json_buffer, VBA.ChrW(VBA.Val("&h" + json_Code)), json_BufferPosition, json_BufferLength
                json_Index = json_Index + 4
            End Select
        Case json_Quote
            json_ParseString = json_BufferToString(json_buffer, json_BufferPosition, json_BufferLength)
            json_Index = json_Index + 1
            Exit Function
        Case Else
            json_BufferAppend json_buffer, json_Char, json_BufferPosition, json_BufferLength
            json_Index = json_Index + 1
        End Select
    Loop
End Function
```

```
Private Function json_ParseNumber(json_String As String, ByRef json_Index As Long) As Variant
    Dim json_Char As String
    Dim json_Value As String
    Dim json_IsLargeNumber As Boolean

    json_SkipSpaces json_String, json_Index

    Do While json_Index > 0 And json_Index <= Len(json_String)
        json_Char = VBA.Mid$(json_String, json_Index, 1)

        If VBA.InStr("+-0123456789.eE", json_Char) Then
            ' Unlikely to have massive number, so use simple append rather than buffer here
            json_Value = json_Value & json_Char
            json_Index = json_Index + 1
        Else
            ' Excel only stores 15 significant digits, so any numbers larger than that are truncated
            ' 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
            ' See: http://support.microsoft.com/kb/269370
            '
            ' Fix: Parse -> String, Convert -> String longer than 15/16 characters containing only numbers and decimal points -> Number
```

```
        ' (decimal doesn't factor into significant digit count, so if present check for 15 digits + decimal = 16)
        json_IsLargeNumber = IIf(InStr(json_Value, "."), Len(json_Value) >= 17, Len(json_Value) >= 16)
        If Not JsonOptions.UseDoubleForLargeNumbers And json_IsLargeNumber Then
            json_ParseNumber = json_Value
        Else
            ' VBA.Val does not use regional settings, so guard for comma is not needed
            json_ParseNumber = VBA.Val(json_Value)
        End If
        Exit Function
    End If
  Loop
End Function

Private Function json_ParseKey(json_String As String, ByRef json_Index As Long) As String
  ' Parse key with single or double quotes
  If VBA.Mid$(json_String, json_Index, 1) = """" Or VBA.Mid$(json_String, json_Index, 1) = "'" Then
    json_ParseKey = json_ParseString(json_String, json_Index)
  ElseIf JsonOptions.AllowUnquotedKeys Then
    Dim json_Char As String
    Do While json_Index > 0 And json_Index <= Len(json_String)
      json_Char = VBA.Mid$(json_String, json_Index, 1)
```

```
            If (json_Char <> " ") And (json_Char <> ":") Then
                json_ParseKey = json_ParseKey & json_Char
                json_Index = json_Index + 1
            Else
                Exit Do
            End If
        Loop
    Else
        Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(json_String, json_Index,
"Expecting """ or """)
    End If

    ' Check for colon and skip if present or throw if not present
    json_SkipSpaces json_String, json_Index
    If VBA.Mid$(json_String, json_Index, 1) <> ":" Then
        Err.Raise 10001, "JSONConverter", json_ParseErrorMessage(json_String, json_Index,
"Expecting ':'")
    Else
        json_Index = json_Index + 1
    End If
End Function

Private Function json_IsUndefined(ByVal json_Value As Variant) As Boolean
    ' Empty / Nothing -> undefined
```

```
    Select Case VBA.VarType(json_Value)
    Case VBA.vbEmpty
        json_IsUndefined = True
    Case VBA.vbObject
        Select Case VBA.TypeName(JsonValue)
        Case "Empty", "Nothing"
            json_IsUndefined = True
        End Select
    End Select
End Function

Private Function json_Encode(ByVal json_Text As Variant) As String
    ' Reference: http://www.ietf.org/rfc/rfc4627.txt
    ' Escape: ", \, /, backspace, form feed, line feed, carriage return, tab
    Dim json_Index As Long
    Dim json_Char As String
    Dim json_AscCode As Long
    Dim json_buffer As String
    Dim json_BufferPosition As Long
    Dim json_BufferLength As Long

    For json_Index = 1 To VBA.Len(json_Text)
        json_Char = VBA.Mid$(json_Text, json_Index, 1)
```

```
    json_AscCode = VBA.AscW(json_Char)


    ' When AscW returns a negative number, it returns the twos complement form of that number.
    ' To convert the twos complement notation into normal binary notation, add 0xFFF to the return result.
    ' https://support.microsoft.com/en-us/kb/272138
    If json_AscCode < 0 Then
        json_AscCode = json_AscCode + 65536
    End If


    ' From spec, ", \, and control characters must be escaped (solidus is optional)


    Select Case json_AscCode
    Case 34
        ' " -> 34 -> \"
        json_Char = "\"""
    Case 92
        ' \ -> 92 -> \\
        json_Char = "\\"
    Case 47
        ' / -> 47 -> \/ (optional)
        If JsonOptions.EscapeSolidus Then
            json_Char = "\/"
```

```
            End If
        Case 8
            ' backspace -> 8 -> \b
            json_Char = "\b"
        Case 12
            ' form feed -> 12 -> \f
            json_Char = "\f"
        Case 10
            ' line feed -> 10 -> \n
            json_Char = "\n"
        Case 13
            ' carriage return -> 13 -> \r
            json_Char = "\r"
        Case 9
            ' tab -> 9 -> \t
            json_Char = "\t"
        Case 0 To 31, 127 To 65535
            ' Non-ascii characters -> convert to 4-digit hex
            json_Char = "\u" & VBA.Right$("0000" & VBA.Hex$(json_AscCode), 4)
        End Select

        json_BufferAppend json_buffer, json_Char, json_BufferPosition, json_BufferLength
    Next json_Index
```

```
    json_Encode = json_BufferToString(json_buffer, json_BufferPosition, json_BufferLength)

End Function


Private Function json_Peek(json_String As String, ByVal json_Index As Long, Optional
json_NumberOfCharacters As Long = 1) As String

    ' "Peek" at the next number of characters without incrementing json_Index (ByVal instead
of ByRef)

    json_SkipSpaces json_String, json_Index

    json_Peek = VBA.Mid$(json_String, json_Index, json_NumberOfCharacters)

End Function


Private Sub json_SkipSpaces(json_String As String, ByRef json_Index As Long)

    ' Increment index to skip over spaces

    Do While json_Index > 0 And json_Index <= VBA.Len(json_String) And
VBA.Mid$(json_String, json_Index, 1) = " "

        json_Index = json_Index + 1

    Loop

End Sub


Private Function json_StringIsLargeNumber(json_String As Variant) As Boolean

    ' Check if the given string is considered a "large number"

    ' (See json_ParseNumber)
```

```
Dim json_Length As Long
Dim json_CharIndex As Long
json_Length = VBA.Len(json_String)

' Length with be at least 16 characters and assume will be less than 100 characters
If json_Length >= 16 And json_Length <= 100 Then
    Dim json_CharCode As String
    Dim json_Index As Long

    json_StringIsLargeNumber = True

    For json_CharIndex = 1 To json_Length
        json_CharCode = VBA.Asc(VBA.Mid$(json_String, json_CharIndex, 1))
        Select Case json_CharCode
        ' Look for .|0-9|E|e
        Case 46, 48 To 57, 69, 101
            ' Continue through characters
        Case Else
            json_StringIsLargeNumber = False
            Exit Function
        End Select
    Next json_CharIndex
End If
```

```
End Function


Private Function json_ParseErrorMessage(json_String As String, ByRef json_Index As Long, ErrorMessage As String)
    ' Provide detailed parse error message, including details of where and what occurred
    '
    ' Example:
    ' Error parsing JSON:
    ' {"abcde":True}
    '          ^
    ' Expecting 'STRING', 'NUMBER', null, true, false, '{', or '['

    Dim json_StartIndex As Long
    Dim json_StopIndex As Long

    ' Include 10 characters before and after error (if possible)
    json_StartIndex = json_Index - 10
    json_StopIndex = json_Index + 10
    If json_StartIndex <= 0 Then
        json_StartIndex = 1
    End If
    If json_StopIndex > VBA.Len(json_String) Then
        json_StopIndex = VBA.Len(json_String)
    End If
```

```
    json_ParseErrorMessage = "Error parsing JSON:" & VBA.vbNewLine & _
                             VBA.Mid$(json_String, json_StartIndex, json_StopIndex - json_StartIndex + 1) & VBA.vbNewLine & _
                             VBA.Space$(json_Index - json_StartIndex) & "^" & VBA.vbNewLine & _
                             ErrorMessage
End Function

Private Sub json_BufferAppend(ByRef json_buffer As String, _
                              ByRef json_Append As Variant, _
                              ByRef json_BufferPosition As Long, _
                              ByRef json_BufferLength As Long)
#If Mac Then
    json_buffer = json_buffer & json_Append
#Else
    ' VBA can be slow to append strings due to allocating a new string for each append
    ' Instead of using the traditional append, allocate a large empty string and then copy string at append position
    '
    ' Example:
    ' Buffer: "abc  "
    ' Append: "def"
    ' Buffer Position: 3
    ' Buffer Length: 5
```

```
    '
    ' Buffer position + Append length > Buffer length -> Append chunk of blank space to buffer
    ' Buffer: "abc       "
    ' Buffer Length: 10
    '
    ' Copy memory for "def" into buffer at position 3 (0-based)
    ' Buffer: "abcdef    "
    '
    ' Approach based on cStringBuilder from vbAccelerator
    '
http://www.vbaccelerator.com/home/VB/Code/Techniques/RunTime_Debug_Tracing/VB6_Tracer_Utility_zip_cStringBuilder_cls.asp

    Dim json_AppendLength As Long
    Dim json_LengthPlusPosition As Long

    json_AppendLength = VBA.LenB(json_Append)
    json_LengthPlusPosition = json_AppendLength + json_BufferPosition

    If json_LengthPlusPosition > json_BufferLength Then
        ' Appending would overflow buffer, add chunks until buffer is long enough
        Dim json_TemporaryLength As Long
```

```
        json_TemporaryLength = json_BufferLength
        Do While json_TemporaryLength < json_LengthPlusPosition
            ' Initially, initialize string with 255 characters,
            ' then add large chunks (8192) after that
            '
            ' Size: # Characters x 2 bytes / character
            If json_TemporaryLength = 0 Then
                json_TemporaryLength = json_TemporaryLength + 510
            Else
                json_TemporaryLength = json_TemporaryLength + 16384
            End If
        Loop

        json_buffer = json_buffer & VBA.Space$((json_TemporaryLength - json_BufferLength)
\ 2)
        json_BufferLength = json_TemporaryLength
    End If

    ' Copy memory from append to buffer at buffer position
    json_CopyMemory ByVal json_UnsignedAdd(StrPtr(json_buffer), _
                    json_BufferPosition), _
                    ByVal StrPtr(json_Append), _
                    json_AppendLength
```

```
    json_BufferPosition = json_BufferPosition + json_AppendLength

#End If

End Sub


Private Function json_BufferToString(ByRef json_buffer As String, ByVal
json_BufferPosition As Long, ByVal json_BufferLength As Long) As String

#If Mac Then

    json_BufferToString = json_buffer

#Else

    If json_BufferPosition > 0 Then

        json_BufferToString = VBA.Left$(json_buffer, json_BufferPosition \ 2)

    End If

#End If

End Function


#If VBA7 Then

Private Function json_UnsignedAdd(json_Start As LongPtr, json_Increment As Long) As
LongPtr

#Else

Private Function json_UnsignedAdd(json_Start As Long, json_Increment As Long) As Long

#End If


    If json_Start And &H80000000 Then

        json_UnsignedAdd = json_Start + json_Increment
```

```
    ElseIf (json_Start Or &H80000000) < -json_Increment Then
        json_UnsignedAdd = json_Start + json_Increment
    Else
        json_UnsignedAdd = (json_Start + &H80000000) + (json_Increment + &H80000000)
    End If
End Function

''
' VBA-UTC v1.0.2
' (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

''
' Convert local date to UTC date
'
' @method ConvertToUrc
' @param {Date} utc_LocalDate
' @return {Date} UTC date
' @throws 10012 - UTC conversion error
```

```
"

Public Function ConvertToUtc(utc_LocalDate As Date) As Date

    On Error GoTo utc_ErrorHandling


#If Mac Then

    ConvertToUtc = utc_ConvertDate(utc_LocalDate, utc_ConvertToUtc:=True)

#Else

    Dim utc_TimeZoneInfo As utc_TIME_ZONE_INFORMATION

    Dim utc_UtcDate As utc_SYSTEMTIME


    utc_GetTimeZoneInformation utc_TimeZoneInfo

    utc_TzSpecificLocalTimeToSystemTime                    utc_TimeZoneInfo,
utc_DateToSystemTime(utc_LocalDate), utc_UtcDate


    ConvertToUtc = utc_SystemTimeToDate(utc_UtcDate)

#End If


    Exit Function


utc_ErrorHandling:

    Err.Raise 10012, "UtcConverter.ConvertToUtc", "UTC conversion error: " & Err.number &
" - " & Err.Description

End Function
```

```
''
' Parse ISO 8601 date string to local date
'
' @method ParseIso
' @param {Date} utc_IsoString
' @return {Date} Local date
' @throws 10013 - ISO 8601 parsing error
''
Public Function ParseIso(utc_IsoString As String) As Date
    On Error GoTo utc_ErrorHandling

    Dim utc_Parts() As String
    Dim utc_DateParts() As String
    Dim utc_TimeParts() As String
    Dim utc_OffsetIndex As Long
    Dim utc_HasOffset As Boolean
    Dim utc_NegativeOffset As Boolean
    Dim utc_OffsetParts() As String
    Dim utc_Offset As Date

    utc_Parts = VBA.Split(utc_IsoString, "T")
    utc_DateParts = VBA.Split(utc_Parts(0), "-")
    ParseIso = VBA.DateSerial(VBA.CInt(utc_DateParts(0)), VBA.CInt(utc_DateParts(1)), VBA.CInt(utc_DateParts(2)))
```

```
    If UBound(utc_Parts) > 0 Then
        If VBA.InStr(utc_Parts(1), "Z") Then
            utc_TimeParts = VBA.Split(VBA.Replace(utc_Parts(1), "Z", ""), ":")
        Else
            utc_OffsetIndex = VBA.InStr(1, utc_Parts(1), "+")
            If utc_OffsetIndex = 0 Then
                utc_NegativeOffset = True
                utc_OffsetIndex = VBA.InStr(1, utc_Parts(1), "-")
            End If

            If utc_OffsetIndex > 0 Then
                utc_HasOffset = True
                utc_TimeParts = VBA.Split(VBA.Left$(utc_Parts(1), utc_OffsetIndex - 1), ":")
                utc_OffsetParts = VBA.Split(VBA.Right$(utc_Parts(1), Len(utc_Parts(1)) -
utc_OffsetIndex), ":")

                Select Case UBound(utc_OffsetParts)
                Case 0
                    utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), 0, 0)
                Case 1
                    utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)),
VBA.CInt(utc_OffsetParts(1)), 0)
                Case 2
```

```
            ' VBA.Val does not use regional settings, use for seconds to avoid decimal/comma issues
            utc_Offset = TimeSerial(VBA.CInt(utc_OffsetParts(0)), VBA.CInt(utc_OffsetParts(1)), Int(VBA.Val(utc_OffsetParts(2))))
        End Select

        If utc_NegativeOffset Then: utc_Offset = -utc_Offset
    Else
        utc_TimeParts = VBA.Split(utc_Parts(1), ":")
    End If
End If

Select Case UBound(utc_TimeParts)
Case 0
    ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), 0, 0)
Case 1
    ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), VBA.CInt(utc_TimeParts(1)), 0)
Case 2
    ' VBA.Val does not use regional settings, use for seconds to avoid decimal/comma issues
    ParseIso = ParseIso + VBA.TimeSerial(VBA.CInt(utc_TimeParts(0)), VBA.CInt(utc_TimeParts(1)), Int(VBA.Val(utc_TimeParts(2))))
End Select
```

```
        ParseIso = ParseUtc(ParseIso)


        If utc_HasOffset Then
            ParseIso = ParseIso + utc_Offset
        End If
    End If


    Exit Function


utc_ErrorHandling:
    Err.Raise 10013, "UtcConverter.ParseIso", "ISO 8601 parsing error for " & utc_IsoString
& ": " & Err.number & " - " & Err.Description
End Function


''
' Convert local date to ISO 8601 string
'
' @method ConvertToIso
' @param {Date} utc_LocalDate
' @return {Date} ISO 8601 string
' @throws 10014 - ISO 8601 conversion error
''
Public Function ConvertToIso(utc_LocalDate As Date) As String
    On Error GoTo utc_ErrorHandling
```

```
    ConvertToIso = VBA.Format$(ConvertToUtc(utc_LocalDate), "yyyy-mm-ddTHH:mm:ss.000Z")

    Exit Function

utc_ErrorHandling:
    Err.Raise 10014, "UtcConverter.ConvertToIso", "ISO 8601 conversion error: " & Err.number & " - " & Err.Description
End Function

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

#If Mac Then

Private Function utc_ConvertDate(utc_Value As Date, Optional utc_ConvertToUtc As Boolean = False) As Date
    Dim utc_ShellCommand As String
    Dim utc_Result As utc_ShellResult
    Dim utc_Parts() As String
    Dim utc_DateParts() As String
    Dim utc_TimeParts() As String
```

```
    If utc_ConvertToUtc Then
        utc_ShellCommand = "date -ur `date -jf '%Y-%m-%d %H:%M:%S' " & _
            """ & VBA.Format$(utc_Value, "yyyy-mm-dd HH:mm:ss") & "' " & _
            " +'%s'` +'%Y-%m-%d %H:%M:%S'"
    Else
        utc_ShellCommand = "date -jf '%Y-%m-%d %H:%M:%S %z' " & _
            """ & VBA.Format$(utc_Value, "yyyy-mm-dd HH:mm:ss") & " +0000' " & _
            "+'%Y-%m-%d %H:%M:%S'"
    End If

    utc_Result = utc_ExecuteInShell(utc_ShellCommand)

    If utc_Result.utc_Output = "" Then
        Err.Raise 10015, "UtcConverter.utc_ConvertDate", "'date' command failed"
    Else
        utc_Parts = Split(utc_Result.utc_Output, " ")
        utc_DateParts = Split(utc_Parts(0), "-")
        utc_TimeParts = Split(utc_Parts(1), ":")

        utc_ConvertDate = DateSerial(utc_DateParts(0), utc_DateParts(1), utc_DateParts(2))
+ _
            TimeSerial(utc_TimeParts(0), utc_TimeParts(1), utc_TimeParts(2))
    End If
```

```
End Function

Private Function utc_ExecuteInShell(utc_ShellCommand As String) As utc_ShellResult
    Dim utc_File As Long
    Dim utc_Chunk As String
    Dim utc_Read As Long

    On Error GoTo utc_ErrorHandling
    utc_File = utc_popen(utc_ShellCommand, "r")

    If utc_File = 0 Then: Exit Function

    Do While utc_feof(utc_File) = 0
        utc_Chunk = VBA.Space$(50)
        utc_Read = utc_fread(utc_Chunk, 1, Len(utc_Chunk) - 1, utc_File)
        If utc_Read > 0 Then
            utc_Chunk = VBA.Left$(utc_Chunk, utc_Read)
            utc_ExecuteInShell.utc_Output = utc_ExecuteInShell.utc_Output & utc_Chunk
        End If
    Loop

utc_ErrorHandling:
    utc_ExecuteInShell.utc_ExitCode = utc_pclose(utc_File)
```

```
End Function


#Else


Private Function utc_DateToSystemTime(utc_Value As Date) As utc_SYSTEMTIME
    utc_DateToSystemTime.utc_wYear = VBA.Year(utc_Value)
    utc_DateToSystemTime.utc_wMonth = VBA.Month(utc_Value)
    utc_DateToSystemTime.utc_wDay = VBA.Day(utc_Value)
    utc_DateToSystemTime.utc_wHour = VBA.Hour(utc_Value)
    utc_DateToSystemTime.utc_wMinute = VBA.Minute(utc_Value)
    utc_DateToSystemTime.utc_wSecond = VBA.Second(utc_Value)
    utc_DateToSystemTime.utc_wMilliseconds = 0
End Function


Private Function utc_SystemTimeToDate(utc_Value As utc_SYSTEMTIME) As Date
    utc_SystemTimeToDate = DateSerial(utc_Value.utc_wYear, utc_Value.utc_wMonth,
utc_Value.utc_wDay) + _
        TimeSerial(utc_Value.utc_wHour, utc_Value.utc_wMinute, utc_Value.utc_wSecond)
End Function


#End If
```