ESP from ThingMagic reader into Excel

On a regular basis there is the request to import the ESP from an RFID tags into Excel. This proof of concept (POC) is showing the steps to read data from the RFID tags, store them with a timestamp in daily files in a specific directory, which can then be manually or automatically imported into Excel.

Contents

[Disclosure 1](#_Toc503294450)

[1. C# source file 2](#_Toc503294451)

[2. Create a connection with Excel 2016 / Office365 5](#_Toc503294452)

[3. Refresh or automatic refresh in minutes the connection 7](#_Toc503294453)

[3.1 Manual refresh 7](#_Toc503294454)

[3.2 Automatic refresh 7](#_Toc503294455)

[4. Automatic refresh with a faster refresh time than 1 minute 8](#_Toc503294456)

[Appendix A. Code snippet for “ThisWorkbook” 10](#_Toc503294457)

[Appendix B. Code snippet for Module 11](#_Toc503294458)

[Appendix C. Test Hardware and software information 12](#_Toc503294459)

Excel is a registered trademark of the Microsoft cooperation©.

ThingMagic and its readers are registered trademarks of the Jadak Cooperation©

# Disclosure

This POC contains free sample software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This POC distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this POC. If not, see <http://www.gnu.org/licenses/>.

Regards,

Paul van Haastrecht

January 2018

# C# source file

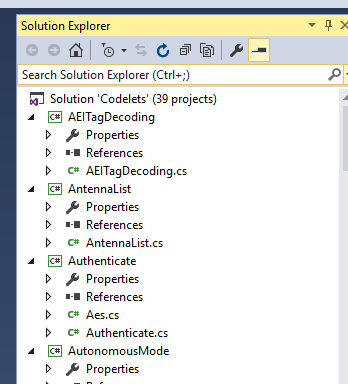
**Download ThingMagic SDK and extract** on a location (e.g. documents) from http://www.thingmagic.com/manuals-firmware

**Prepare first time**. After the API has been downloaded and extracted. In this case it has been extracted to Documents : c:\user\Paul\Documents\mercury-api1.29.4.34.

Go to directory: c:\user\Paul\Documents\mercury-api1.29.4.34\cs\Samples\Codelets.

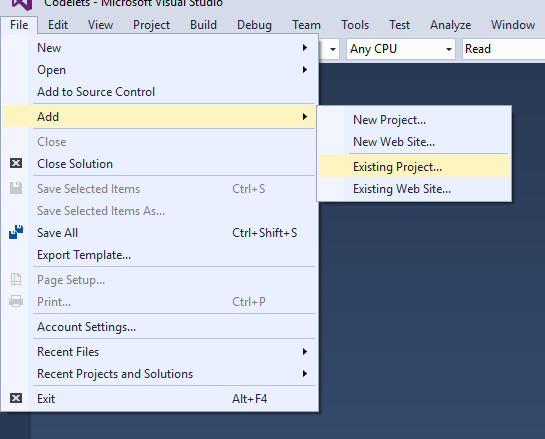
Double click on : Codelets.sln

Visual Studio will now do a one-time update of the files and show at the end a website with errors or warnings detected.

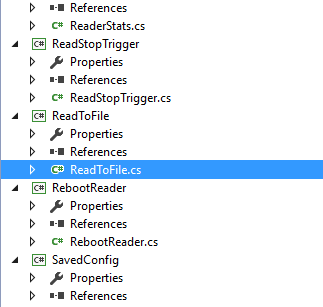
**Download Readtofile.zip and extract** on a location (e.g. documents)

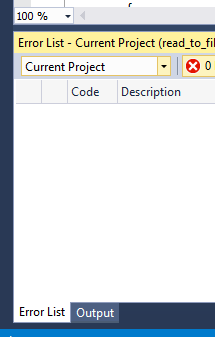
**Install source ReadToFile**

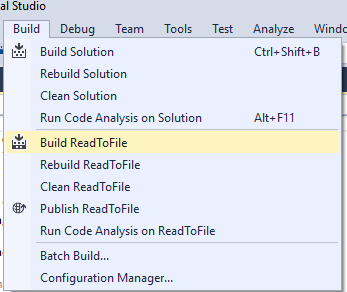
If you don’t see a solution explorer on the right, select in the ribbon “view” and then “solution explorer”. By default the current version has 39 examples.

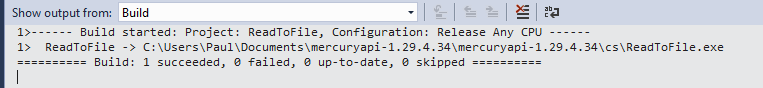
**Copy the downloaded** and extracted directory ReadToFile (& content) to the installed directory of the API. In this case: c:\user\Paul\Documents\mercury-api1.29.4.34\cs\Samples\Codelets

In Visual studio now select File, Add, Existing project. Go to the ReadToFile directory: ThisPc\Documents\mercury-api1.29.4.34\cs\Samples\Codelets\ReadToFile and **double click on the ReadToFile.csproj**

A new entry will have been created in this solution called ReadToFile. D**ouble clicking on the ReadToFile.cs**, will show the source.

**To build only this project** of the solution, at the bottom of page, select “current project” , then select Build & build ReadToFile



A successful build will show like this

**Running the program**

Create a directory in Document called: epcdata

Start a command box for windows

Go to the indicated directory in the picture above. In our case: c:\users\Paul\Documents\mercury-api1.29.4.34\cs\

Connect your reader and select the right port. In my case: ReadtoFile tmr:///com3 - -ant 1,1

The program will continue to loop with reader tries to find an EPC from an RFID tag. If found it will display the EPC and store that in a file.

**Working of the program**

After initialization if has a continue loop where it tries to read EPC from an RFID tag until CNTRL +C. If an ESP has been read, it will display the ESP and call SaveToFile(). In SaveToFile(), it will obtain the current timestamp as well as the extension for the output file. The file path is created by appending the Documents directory + FILE\_NAME + Extension + txt. FILE\_NAME is define at the top as “\epcdata\out”. The extension has the format of YYYYMMDD as such the full path will be C:\Users\Paul\Documents\epcdata\out20180109.txt

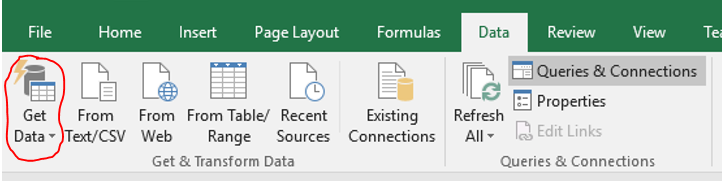
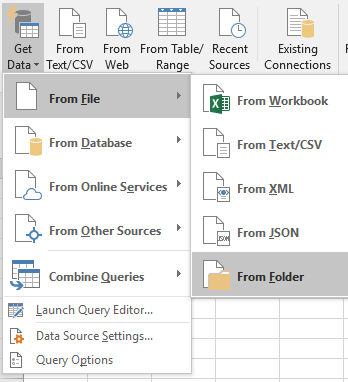
If the filename does not exist a header will be written with :

EPC,year,month,day,hour,minute,sec

The lines that follow have the data stored in that format. The file can be read with any text editor.

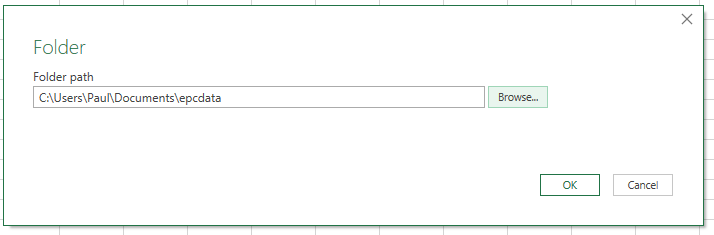
# Create a connection with Excel 2016 / Office365

Excel does not offer a constant, direct connection into a tab (at least I have not found it). You have to select the external data import from a location on the header ”data”:

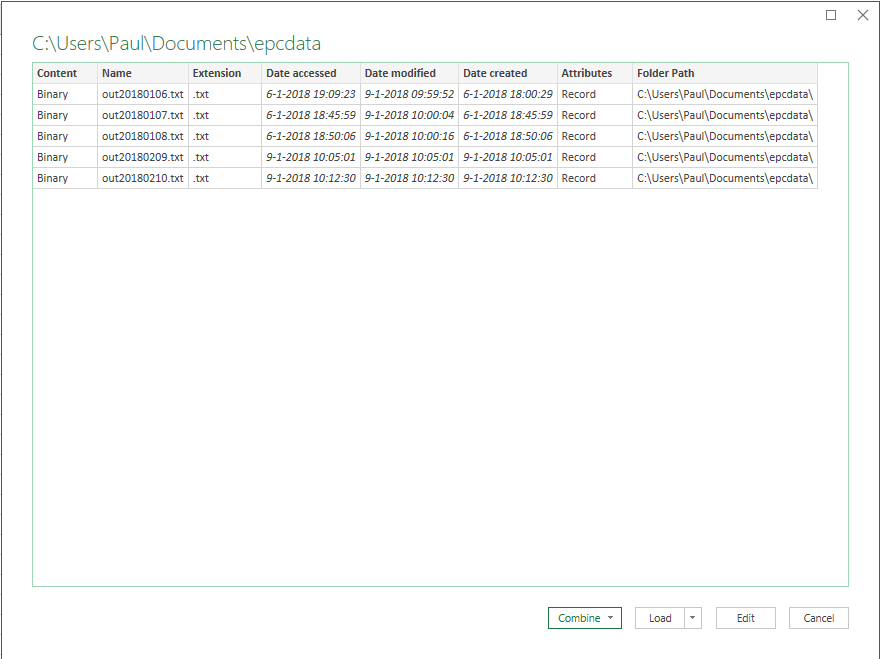


You can select many sources from a single file / web etc. In this POC we create a file for each day, which we store in a single folder, that we want to import: select the “**Get Data**” at the left and in the sub-menus select “**From File”, “From Folder**”

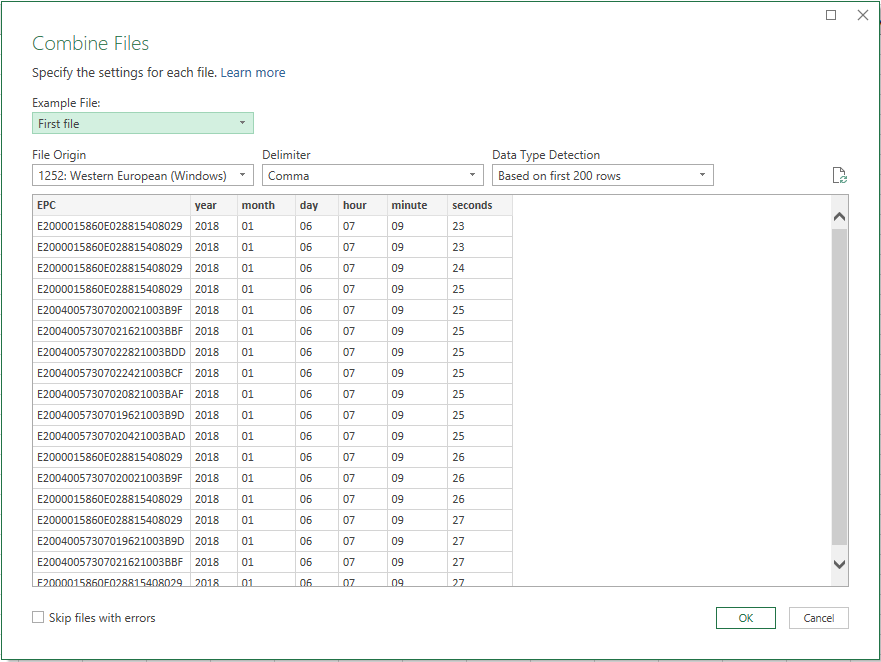
In our example we have used c: \user\Paul\Documents\epcdata. **Browse to select the directory** where the files are stored and select OK



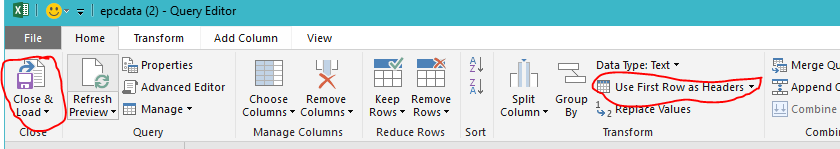
You will get a file menu with files currently in this directory and select **Combine & Edit**



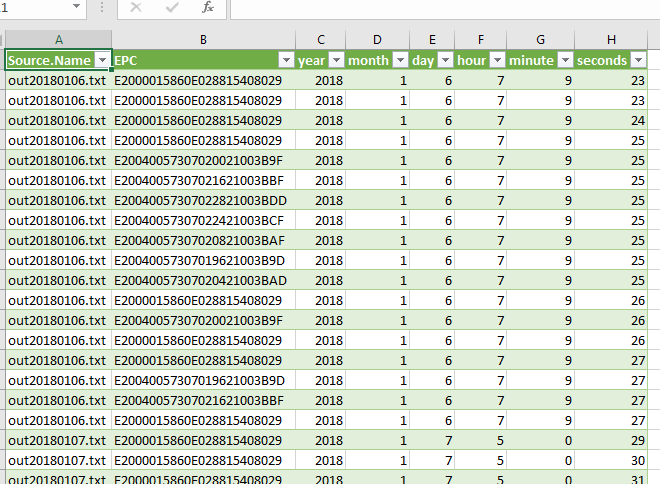
Now something like below should show up. Watch the top row, that should have the headers if not we will change that in the next step. Select **OK**.



Next step is the query editor. Don’t worry, we will only need to edit if the header was NOT showing EPC, year etc, but columns. In that case select in the top: “**Use First Row as Headers**”. We are done and select “**Close & Load**”



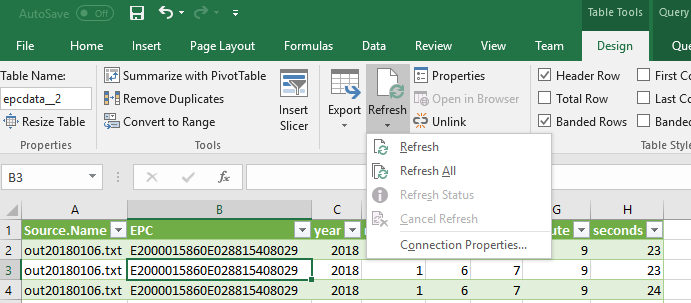
The processing that follows might take some time, but you should get a screen that looks like



# Refresh or automatic refresh in minutes the connection

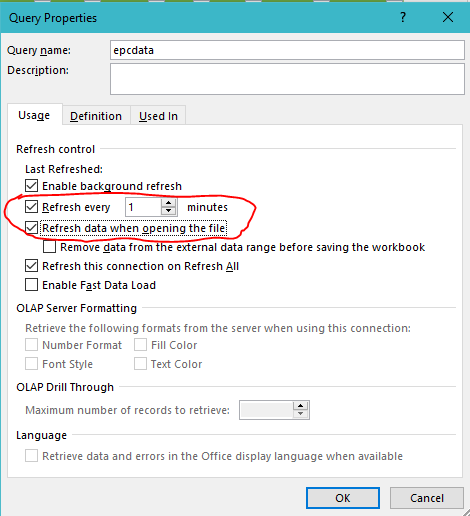
## 3.1 Manual refresh

Select a **field in the imported** data, the ribbon on the top will change, select “**refresh**” then “**refresh**” or “**Refresh All**” if you want to refresh all the connections.



## 3.2 Automatic refresh

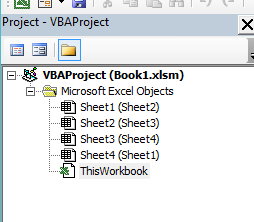
Select a **field in the imported** data, the ribbon on the top will change, select “**refresh**” and “**Connection Properties**”. The follow menu will appear. Select Refresh every x minute. (X = number of minutes in the example below every minute) and select **Refresh data when opening the file**. Select **OK** to close the screen. Now every minute the connection is refreshed. ESP that are added into the file are loaded, ESP and files from c:\user\paul\documents\epcdata that are removed.

.

The only drawback might be that your screen/ cursor blinks every minute when updating

# Automatic refresh with a faster refresh time than 1 minute

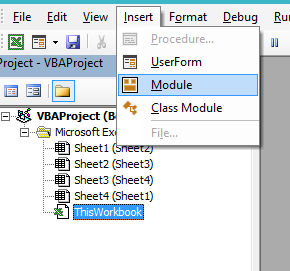
Sometimes it is necessary to get a faster refresh than 1 minute. This can be achieved with VBA macro. It is not as difficult as you might think and you just needs some copy/paste on the code below to the right places.

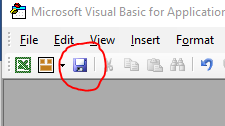
**!!!!! MAKE SURE TO DISABLE THE AUTOMATIC REFRESH OPTION FROM STEP 3.2 !!!!!**

To start the Macro editor: **ALT + F11**

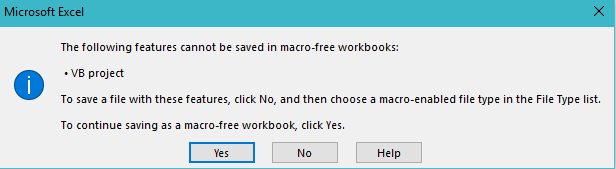
Show the project explorer, either from the ribbon with “View” and then “project explorer” or Keyboard : **CNTRL + R**.

**Double-click** on ThisWorkBook and a box will pop-up. **Copy and paste the code from Appendix A** in the box

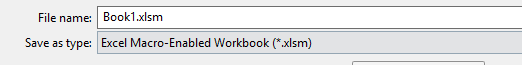
Select from the ribbon : “Insert” and then “Module”. A new box will pop-up**. Copy and paste the code from Appendix B** in the box on the screen.

Now select **save** :

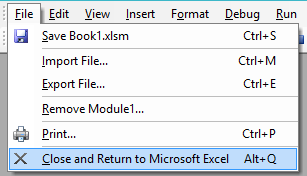
As we are adding a macro, we have to give a different extension to the workbook and as such a message will pop-up :



Now click: **No**. The “save-as” screen will pop-up and select the extension. **xlsm**



We can now leave the macro editor with “**File**”, “**Close and Return to Microsoft Excel**”



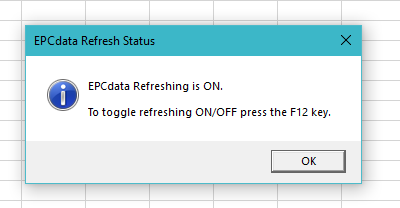
**Close** the workbook and **reopen** it. You might get the security message



Select “Enable Content” in that case

A pop-up will show up. With F12 you can either disable or enable the background refresh.

Press **OK** to continue and the background refresh will start and continue to happen each 15 seconds. This can be suspended or started by pressing F12.



If you want you can **change the interval from 15 seconds** to another timing:.

* Start the macro editor ALT + F11,
* View “project explorer” (CNTRL + R),
* double click module 1 and look for the line : RunWhen = Now + TimeValue("00:00:15")
* Change the 00:00:15 to 00:00:30 for a 30 seconds interval.
* Save and return to the workbook and restart the workbook.

The only drawback might be that your screen/ cursor blinks every minute when updating

# Appendix A. Code snippet for “ThisWorkbook”

Private Sub Workbook\_Open()

'version 1.0 / January 2018 / paulvha

‘Monitor the F12 key and when pressed run the ToggleRefresh procedure

Application.OnKey "{F12}", "ToggleRefresh"

'The refresh process will be off by default when the book is opened

'If you want it on then run this line below.

'Note because the RefreshOn variable is false by default and ToggleRefresh

'changes it to the opposite then RefreshOn will be changed to True and the

'RefreshQuery procedure will be called by the ToggleRefresh procedure.

Call ToggleRefresh

End Sub

# Appendix B. Code snippet for Module

‘version 1.0 / January 2018 /paulvha

‘A public variable that will stay in memory while the book is open (False by default).

'Its used as an indicator so you know whether to refresh or not in the RefreshQuery procedure

Public RefreshOn As Boolean

Public RunWhen As Double

Public Sub ToggleRefresh()

'Toggle the refresh to the opposite value : if its currently false then it will now be true and visa versa

RefreshOn = Not RefreshOn

If RefreshOn = True Then

'If RefreshOn is true you want to start the refresh process

Call RefreshEPC

MsgBox "EPCdata Refreshing is ON." & vbLf & vbLf & \_

"To toggle refreshing ON/OFF press the F12 key.", vbInformation, "EPCdata Refresh Status"

Else

'stop the pending ontime procedure

On Error Resume Next

Application.OnTime RunWhen, "RefreshQuery", schedule:=False

On Error GoTo 0

MsgBox "EPCdata Refreshing is OFF." & vbLf & vbLf & \_

"To toggle refreshing ON/OFF press the F12 key.", vbInformation, "EPCdata Refresh Status"

End If

End Sub

Public Sub RefreshEPC()

Dim cn As WorkbookConnection

‘update ALL connections. This can be changed to specific connection if needed

For Each cn In ActiveWorkbook.Connections

cn.Refresh

Next

'Repeat this procedure every 15 seconds. The false argument should clear

'the Ontime event if its in memory ready to run (ie stop it running twice in

'quick succession). You check to see if RefreshOn is true before repeating

'the procedure again

If RefreshOn = True Then

RunWhen = Now + TimeValue("00:00:15")

On Error Resume Next

Application.OnTime RunWhen, "RefreshEPC"

On Error GoTo 0

End If

End Sub

# Appendix C. Test Hardware and software information

This proof of concept (POC) has been tested and created with the following conditions

Hardware:

* Reader : ThingMagic M6E-NANO (USB connection)

Software version:

* Operating System : Microsoft Windows 10
* ThingMagicAPI : ThingMagic mercuryapi-1.29.4.34
* Development tool : Microsoft Visual Studio Community 2015
* Excel 2016 / Office 365 : Microsoft MSO 16.0.8827.2099 32-bit

The code for the macro’s have been derived from different code-snippets on public sources (forums and helpfiles) from the Internet.

The ReadToFile() program is based on the original Read() example code which is part of the API.