FXP20 KEY INJECTOR

TRUDU, Laurent

Revision History

Date	Author	Version	Comments
17th March 2025	Laurent Trudu	1.0	Initial Version
22nd May 2025	Laurent Trudu	1.1	Added GUI description, Config.xml file description and more details on the Setup section of this document.
27th May 2025	Laurent Trudu	1.2	Added EPC reading using GPI port Configuration information in Config.xml file description.
28th May 2025	Laurent Trudu	1.3	Added ClipboardPasteDelay description in Config.xml section.
8th July 2025	Laurent Trudu	1.4	Added EULA
8th September 2025	Laurent Trudu	1.5	Added RemoveDuplicates description in Config.xml section
9th September 2025	Laurent Trudu	1.6	Add a section to explain how to start automatically the application

1. EULA

Before using the application, you must read and accept the End User License Agreement (EULA).

An EULA.txt file is included in the software distribution.

A copy of the EULA is available on the Zebra website:

https://www.zebra.com/us/en/support-downloads/eula/restricted-eula.html

Or in the following text file:

EULA.txt

Using the FXP20KeyInjector software means that you accept the terms and conditions defined in the EULA document.

2. Overview

The FXP20 Key Injector application provides integration into the Windows desktop environment without the need of using the Zebra SDK.

It extends the capabilities of the FXP20 by implementing:

- Keyboard wedge using key injection
- Keyboard wedge using copy/paste injection
- MQTT protocol for data sending and hardware control

The application is provided as an executable that can be run on Windows.

If setup in Keyboard Wedge mode, the application will send the EPC read to all windows that will match a name filter.

If setup in MQTT mode, the application will send the EPC read to a defined topic, and it will be controlled through a specified control topic.

1. Availability

Please not that this software is still in the process of review and must not be shared with partners without authorization.

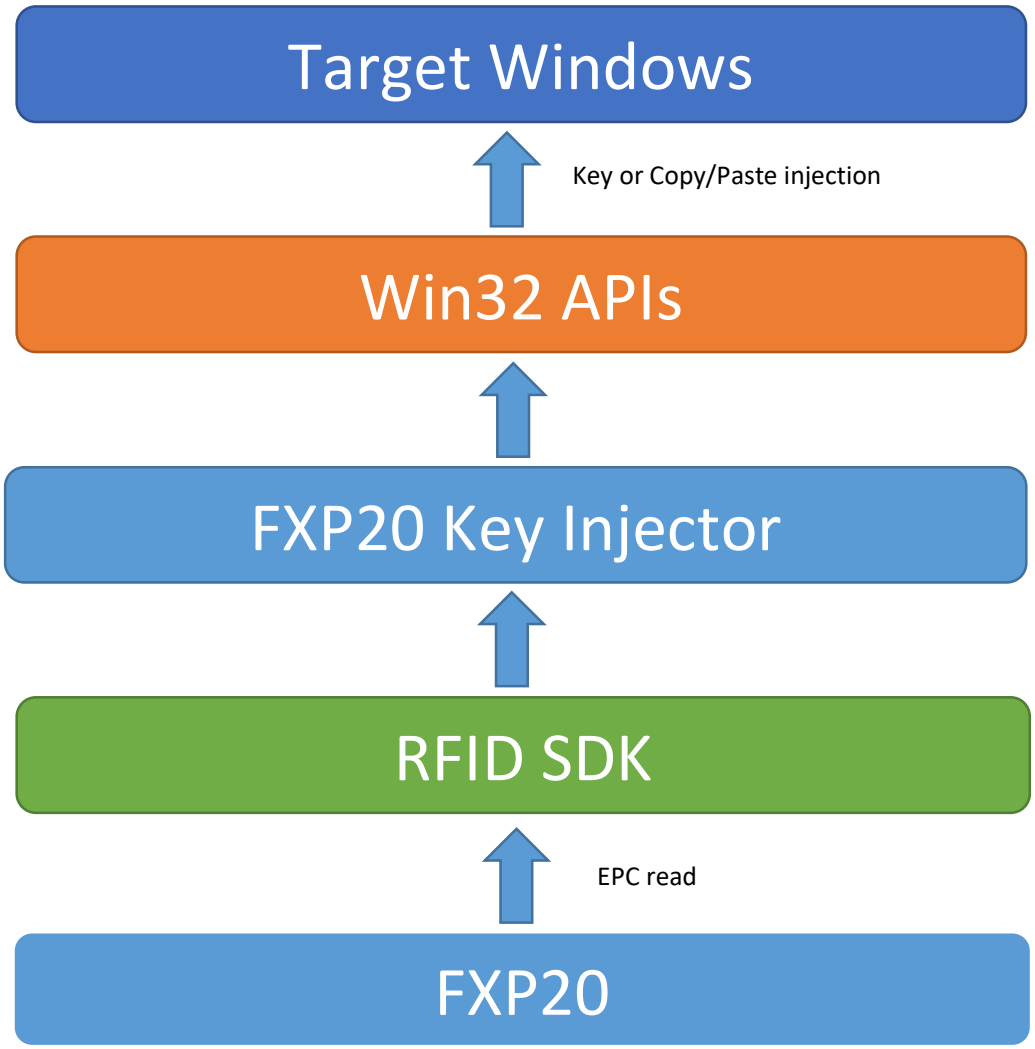
Contact me for more information: laurent.trudu@zebra.com

2. Architecture

When setup in Keyboard wedge mode, the application will directly broadcast EPC read to all the windows that matches a specified filter in the configuration.

For better performance, it is advised to limit the key injection to a specific window.

If many windows are needed, it is advised to use the copy/paste injection method that provides better performance and stability than the key injection method.



When setup in MQTT Mode, the FXP20 injector will need a MQTT Broker to connect to.

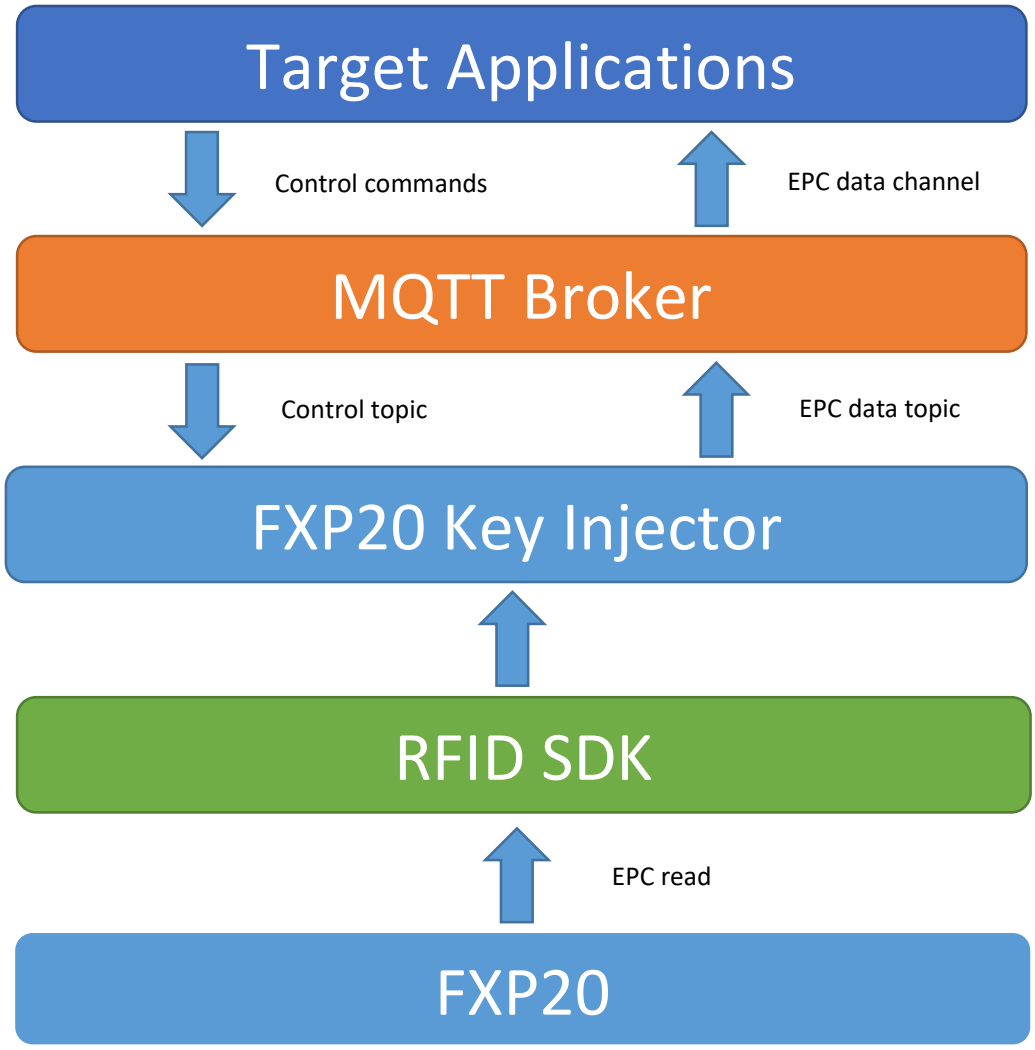
The address and the login/password are set inside the configuration of the FXP20 Key Injector.

The FXP20 Key injector will register itself to a specified control topic and will send all the EPC read to another specified data topic.

These topics are set inside the FXP20 Key injector configuration file.

The MQTT Broker will dispatch the EPC read to the registered application and will dispatch the commands received from the applications to the FXP20 Key Injector app.

The system can work with multiple apps but it is advised to use a single app for FXP20 control.



3. Pre-requisites

Hardware:

- A PC with usb >= 2.0 ports
- A FXP20

Software:

- The application has been tested on Windows 10 and Windows 11 and should work on Windows server as well.
- .NET Framework runtime >= 4.8
- MQTT Broker if necessary. The Mosquito MQTT Broker version 2.0.20 has been tested successfully

4. Installation

Install the .NET Framework 4.8 to the target PC.

The Framework runtimes can be found here:

https://dotnet.microsoft.com/en-us/download/dotnet-framework/net48

Unzip the FXP20 Key Injector archive in a folder on the PC that will run the application.

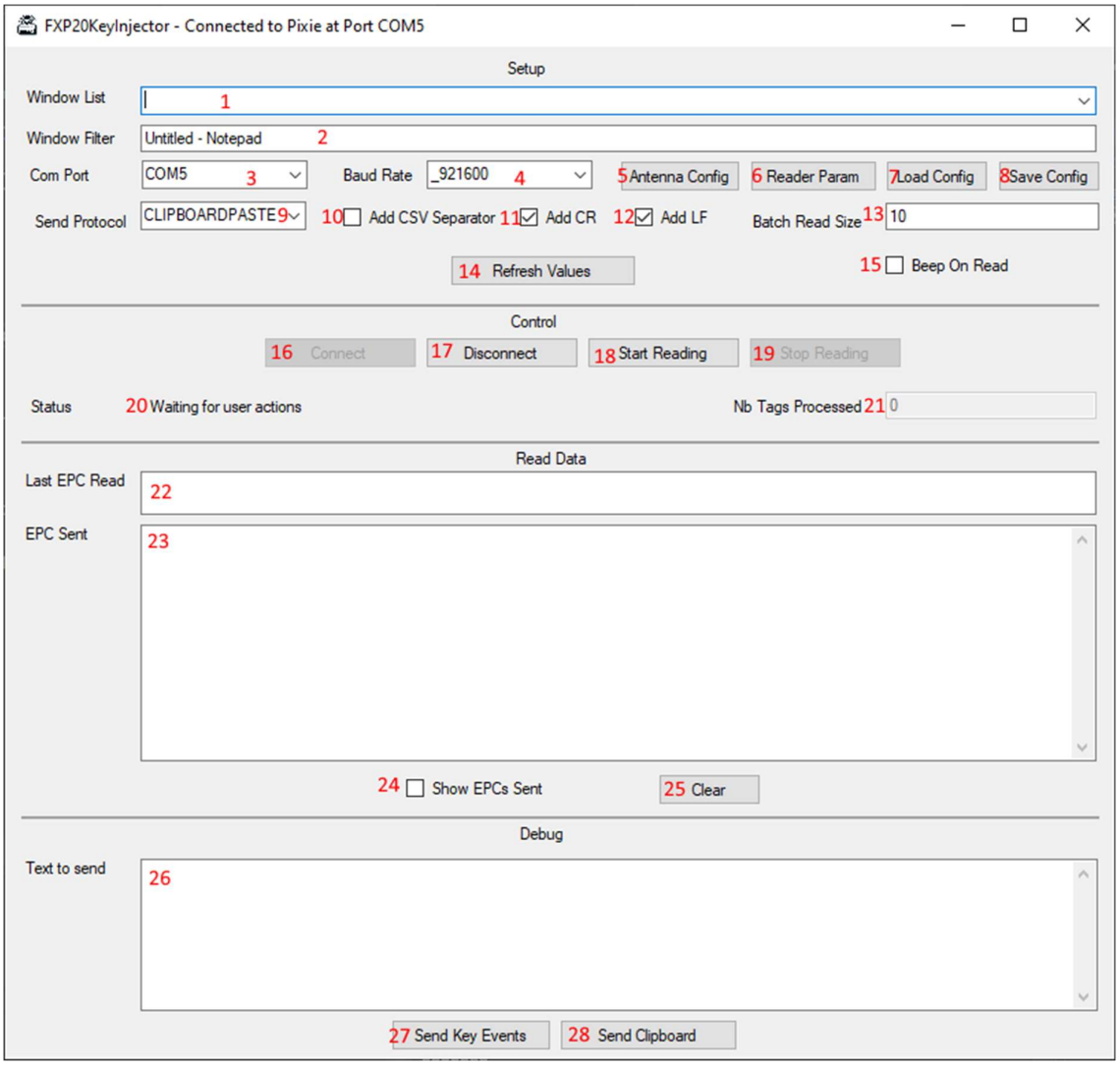
No more steps are needed if you want to use Key or Copy/Paste injection.

If necessary, install the MQTT broker on the target PC or on a separate server depending on your needs.

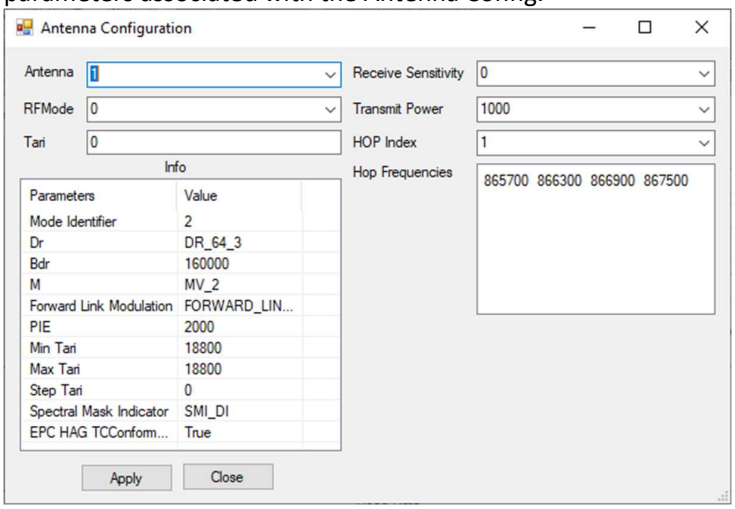
Update the target PC system path to the folder that contains the broker binary to make its launch easier.

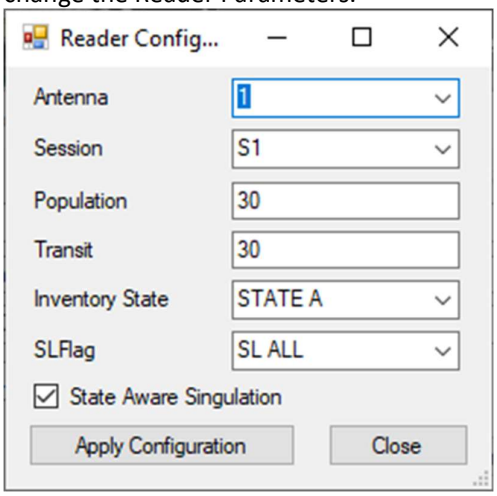
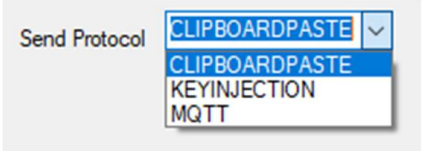
Sample config file and launcher for Mosquitto MQTT Broker are provided in the appendice section of this document.

3. Graphical User Interface



1	A drop-down list that contains all the names of the windows that are actually opened in the operating system. Clicking the button “Refresh Value” (14) will re-fill this list with all the windows names (useful if you opened an application after running FXP20KeyInjector). When you select a window name, it will automatically fill the filter input field (2). This drop-down list only serves to list the available windows and fill the filter input field. It is a helper to see what the name of the currently opened windows are.
2	An input field that contains a filter parameter. When using the Key injection or the Clipboard Paste protocol, the software needs to know where to redirect the EPCs that have been read. To find the correct windows, it will check the names of all the opened windows. If it finds the content of this input field inside a window name, it will force the focus on this window and send the EPC as Keys or with a Clipboard Paste.

	Ex: filling this field with “Notepad” will make FXP20KeyInjector to inject EPCs to all the windows which names contains “Notepad”. This can be the original Notepad app, or even Notepad++ app since they share a common sequence of characters in their titles which is “Notepad”. To make it easier to fill this input field, you can rely on the Windows Names Drop-Down List (1) by selecting a window name directly in this list. It will fill the filter parameter accordingly.
3	A drop-down list that contains all the COM ports associated with a FXP20. If many FXP20 are connected to the same computer, the list will contains all the COM ports associated with each FXP20. Since we can’t name the FXP20, it is advised to do the old school try and error method by selecting a COM port, then Connect to the reader using the Connect button (16), then check the LEDS on the FXP20s to see which one was connected.
4	The speed of the connection wit the FXP20. This can be: • 230400 • 460800 • 115200 • 921600 The default speed is 921600. To get the best performance and stability it is advised to “Not Change” this parameter, unless you know what you are doing 😊
5	Antenna config button, this will open a modal window that will allow you to change the parameters associated with the Antenna Config.  Parameters are: • RFMode • Tari • Receive Sensitivity • Transmit Power • Hop Index Please note that the reader MUST be connected to be able to change its antenna configuration with the button Connect (16). When changing a parameter on a specific antenna, user must Apply the configuration BEFORE modifying the parameters associated with another antenna for them to be take into consideration.

	The user MUST apply the configuration before closing the window if he wants it to be applied. When the user changes the antenna, the application will present its specific parameters. The parameters are specific to the antenna that is currently selected.
6	Reader Parameters button, this will open a modal window that allows the user to change the Reader Parameters.  The parameters are: • SESSION type • Population • Transit • Inventory state • SLFlags • State Aware Singulation The parameters are specific to each antenna. Please note that the reader MUST be connected with the Connect Button (16) to be able to change these parameters. When the user changes a parameter, he must Apply the configuration for it to be considered before trying to change parameters of another antenna or closing the modal window. When the user changes the antenna, the specific parameters of this antenna will be presented to the user.
7	Load config button. This button loads the config from the Config.xml file that is located in the folder of the FXP20 application (see Config.xml section for more details).
8	Save config button. This button will overwrite the Config.xml file located in the folder of the application with the parameters that have been defined using the GUI. Please note that some parameters are not available in the GUI and must be changed manually by editing the Config.xml file (see Config.xml section for more details)
9	Send Protocol drop down list. This control allows the user to choose how the EPCs will be processed once read.  Three options are available: • Clipboard Paste: this is the best option for keyboard emulation. It simulates a Paste (CTRL+V) of the EPC read in all the windows detected thanks to the

	window filter parameter (2). Since it is using the clipboard it has better performances and stability than using the Key Injection mode. By default, a delay of 500ms is applied between each clipboard paste to prevent data loss. You can try lower values as 100ms should be enough, but not lower. This parameter can be changed in the Config.xml file. See ClipboardPasteDelay for more information. • Key Injection: Instead of pasting the whole EPC in one call, the Key Injection protocol will send keystrokes to the target windows. It is slower and less stable than the Clipboard Paste method, but some application requires this keystroke emulation to work properly. • MQTT : This is the best protocol to retrieve EPCs read by the FXP20, see the MQTT messaging section of this document for more information.
10	Add CSV Separator. A semicolon will be added after each EPC read to match CSV formatting.
11	Add CR. A carriage return character will be added after each EPC read.
12	Add LF. A line feed character will be added after each EPC read.
13	Batch Read Size. The number of EPC that the FXP20 can process at every read cycle can be configured with this input field.
14	Refresh Value: This button will refresh the content of the Window List (1) and Com Port (3) drop down lists.
15	Beep On Read. The reader will emit a beep every time it reads an EPC.
16	Connect. This button allows the user to connect to the reader that has been selected using the Com Port drop down list (3).
17	Disconnect. This button will disconnect the reader from the FXP20 application. If the reader was in reading mode, it will be stopped before being disconnected.
18	Start Reading. This button will start the reading tags process.
19	Stop Reading. This button will stop the reading tags process.
20	Status. A label that indicates the current status of the application.
21	NB Tags Processed. This label indicates the number of tags that have been processed since the application has been launched. It can not be reset (you have to quit the application to reset it).
22	Last EPC Read. In this label, you'll find the last EPC data that has been read by the application.
23	EPC Sent. Displays a list of EPCs that have been sent through keyboard emulation or MQTT.
24	Show EPC Sent. If checked, every time an EPC is read, the app will show it in the Last EPC Read (22) and add it to the list of EPC Sent (23).
25	Clear. This will clear the list of EPCs Sent (23).
26	Text To Send. This window is for debug purposes. You can put some text in it and test if the Send Key Events (27) or Send Clipboard (28) methods works properly with the Window Filter (2) that you defined in the application.
27	Send Key Events. This button will send all the text contained in the Text To Send field (26) using Key Injection method. Use this for debugging purposes.
28	Send Clipboard. This button will send all the text contained in the Text To Send field (26) using Clipboard Paste emulation.

4. Setup

1. Configuration file

The configuration of the application can be done from the GUI or by manually changing the config.xml file found in the folder of the exe file.

If this file is available when the application is launched, it is read automatically.

If the file is not available, it will be created with default values.

If the configuration is changed, it has to be saved manually to update the file using the button “save config”.

If the file is updated manually, it can be re-loaded using the “Save Config” button.

Each time the reader is connected using the “Connect” button, the current configuration is applied to the reader.

All the elements are explained in a section later in this document.

2. Connection configuration

When the application is launched, it will try to discover all the readers connected to the host PC.

If no reader is found, it will display the following error message:

If one or many readers are found, the COM Port drop down list fill be filled with the ports that are used by the connected readers.

The reader port that will be used is setup by selecting the right port in the COM Port drop down list.

If the settings are saved with a selected COM Port using the Save Config button, the application will select this port automatically when relaunch if it finds a reader connected to it.

If a reader is connected to the PC after launching the application, the COM Port list can be updated by using the “Refresh Values” button.

The COM port can be set manually inside the config.xml file with the element <ComPort>.

The COM port is specified using the format COMx, where x is the number of the port used by the FXP20.

By default, the communication speed is set to 921600 bauds.

It is advised to let this setting to this value for better performances. It can be modified if necessary.

If can be set manually inside the Config.xml file by changing the <BaudRate> element.

Once these parameters have been set, the FXP20 Key Injector can be connected to the hardware manually with the button “Connect”.

The FXP20 Key Injector can automatically connect to the reader when launched if the <AutoConnect> element is set to true in the configuration file.

It will automatically start reading tags after connection if the <AutoStartReading> element is set to true in the configuration file.

3. Send protocol

The “Send Protocol” drop down list allow to select the way the FXP20 Key Injector application will interact with the client application**.**

The standard sending protocols are keyboard emulation.

Three settings are available for this behaviour:

- KEYINJECTION
- CLIPBOARDPASTE
- MQTT

1. KeyInjection

The Key Injection will send all characters of the EPC data sent by the FXP20 one by one sequentially.

If “add CSV Separator” option is selected, a “;” separator will be added between every EPC data send.

If “add CR” is selected, a carriage return will be added after each EPC read.

If “add LF” is selected, a line feed will be added after each EPC read.

The batch size defines the number of data that will be read at the same time, all the read EPC will be send sequentially to the target window.

2. CLIPBOARDPASTE

The difference between KEYINJECTION and CLIPBOARDPASTE is that in this mode, the content of the EPC is sent to the clipboard, and a paste command is then injected by the FXP20 Key Injector.

This method provides the best performance and is the most stable.

If “add CSV Separator” option is selected, a “;” separator will be added between every EPC data send.

If “add CR” is selected, a carriage return will be added after each EPC read.

If “add LF” is selected, a line feed will be added after each EPC read.

A delay is added after each paste to prevent loss of data by default it last 500ms.

This delay can be configured with the element ClipboardPasteDelay in the Config.xml file.

See Config.xml section for more information.

3. Target Window

When using keyboard emulation, the window that will receive keyboard events has to be specified.

If the user clicks on the “Refresh Value”, the application will list all opened windows inside the Window List drop down.

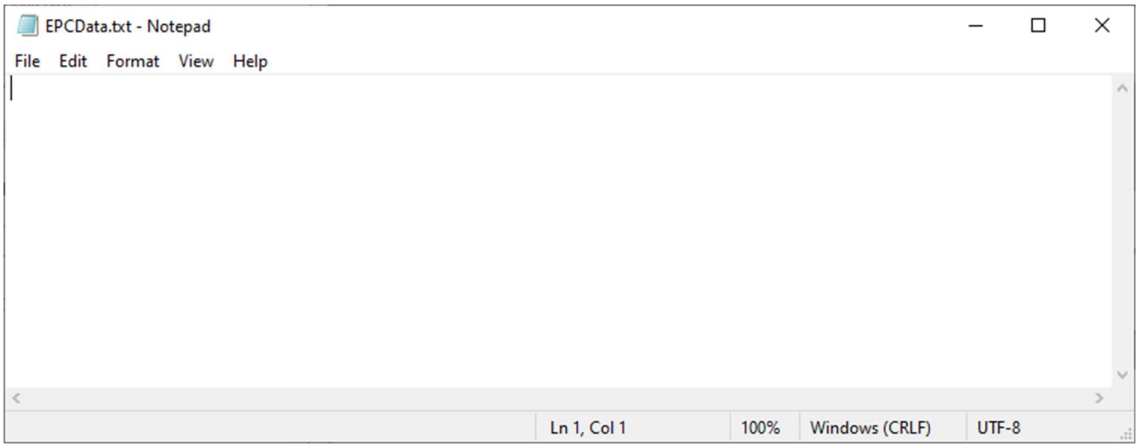
If the user selects one entry in this drop-down list, the “Window Filter” edit text will be filled with the selected window title.

The filter value will be used to find the windows that will receive the key or clipboard injection.

When the FXP20 will read a tag, the application will list all the available windows and send injection event to all the windows that contains the value in the “Window Filter” text edit inside their window title.

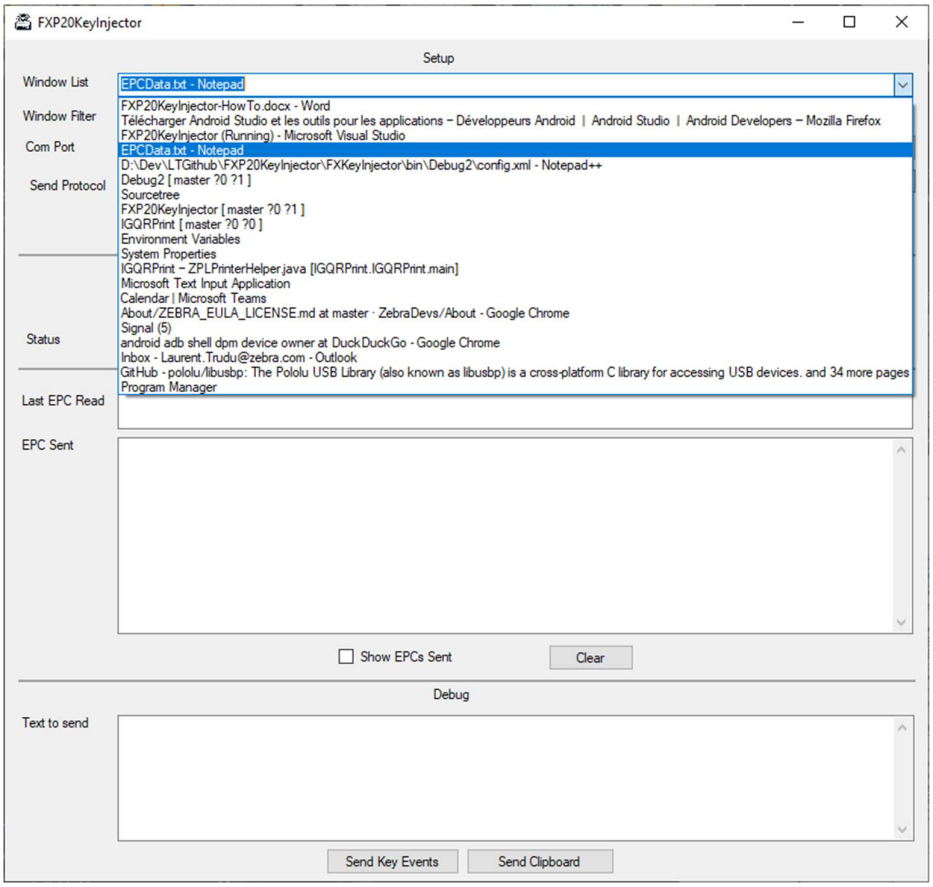
Ex:

Let’s assume we have a notepad window that need to receive the data read by the FXP20.



If we want to explicitly send data to this window, we first open the window, then click on “Refresh Values” button.

It will update the “Window List” dropdown with all the opened windows.



Selecting the “EPCData.txt – Notepad” entry will fill the “Window Filter” with the name of the window.

Now all the EPC read will be sent to this specific window.

If we have two different notepad windows, we must change the filter to a less specific value that targets all the notepad windows for ex: “Notepad”.

The filter search is case-sensitive and culture-insensitive.

4. MQTT

The MQTT protocol is available for more complex use cases.

To work properly, it needs a MQTT Broker that can be launched on a separate server, or on the hardware that will host the FXP20 Key Injector.

The MQTT broker needs to be launched before the FXP20 Key Injector.

The connection to the MQTT Broker must be secured using a username and a password.

When MQTT protocol is selected or if it was set in the config file, the FXP20 application will connect itself to the MQTT Broker using the parameters defined in the config.xml file.

If the application is disconnected from the Broker and an EPC is read and must be sent, the application will try to reconnect to the broker before trying to send the received EPC.

The MQTT configuration is done using the config.xml file, as there is no GUI to do it inside the application.

The elements are:

- MQTTServer: the IP address of the MQTT Broker server
- MQTTPort : the port of the server
- MQTTUser : the user name to connect on the MQTT server
- MQTTPassword : the password of the MQTT server
- MQTTSendTopic : the topic that will be used to send the EPC read data
- MQTTControlTopic : the topic that will be used to control the application behaviour

4. MQTT messaging.

1. Receiving EPC data

To receive EPC readings, the client application must connect to the Broker, then register itself to the topic defined in the MQTTSendTopic configuration.

All the EPC that the FXP20 will read will be sent in a separate message on the user defined topic as a text payload.

2. Controlling the FXP20

Once launched the FXP20 Key Injector application will register itself to the MQTTControlTopic topic defined in the config.xml file.

The FXP20 Key Injector will listen to the MQTTControlTopic for the following text payloads:

- Connect : this will launch a FXP20 connection command. It will act as if the user click on the “Connect” button from the GUI.
- Disconnect : this will disconnect the FXP20 Key Injector from the FXP20 hardware.
- StartReading : this will set the FXP20 to read EPC tags. The FXP20 Key Injector must be connected to the FXP20 to start reading tags.
- StopReading : this will stop the user from reading Tags.
- Beep : will perform a Beep using the configuration defined in the Config.xml file (see appendix for the elements that configure the beep)
- UnBeep: will force the reader to stop beeping.

5. Config.xml file

In this section you'll find a description of all the elements that are available in the Config.xml file.

All parameters are mandatory.

The best way to obtain this file is to launch the application without any Config.xml

A default values Config.xml will be created by the application in the folder it has been run.

Element Name	Format	Values	Description
Configuration	N/A	N/A	The root element of all the configuration.
TargetWindowFilter	String	Text String	Contains the filter that will be used to find the target window(s) that will receive injected keys or pasted EPC values.
ComPort	String	COMx with x an integer number	Contains the value of the COM port that will be used to connect to the reader. It is the USB virtual com port where the reader is connected. Ex: COM5 represents the fifth virtual COM port.
BaudRate	Integer	• 230400 • 460800 • 115200 • 921600	Contains the speed of the serial connexion with the FXP20 Reader. The default speed is 921600. To get the best performance and stability it is advised to "Not Change" this parameter, unless you know what you are doing 😊.
MQTTServer	String	An ip address.	Contains the IP address of the MQTT Server. Default value is 127.0.0.1 (localhost). Host name resolution has not been tested but should be supported (by RFID SDK). If not, contact the SE who gave you access to this software in order to report the issue to the developer of the software.
MQTTPort	Integer	A connexion port.	Contains the port that will be used to connect to the MQTT Server. Default value is 1883 which is the default port of standard MQTT server.
MQTTUser	String	A user name / login	Contains the user name that will be used to connect to the MQTT server. It is mandatory that the server is setup with a user / password requirements.
MQTTPassword	String	A password	The password that will be used to connect to the MQTT Server.

			It is actually displayed in plain text. For security reasons, a future version may store this value in encrypted format.
MQTTSendTopic	String	A MQTT topic name in the form of a full path with subtopics if necessary.	The MQTT topic on which the application will send any EPC read. Your client application must register to this topic in order to receive the EPC that are read by the FXP20.
MQTTControlTopic	String	A MQTT topic name in the form of a full path with subtopics if necessary.	This will contain the MQTT topic that can be used to control the FXP20 application behaviour. This topic listens to specific messages. See the "Controlling the FXP20" section of this document for more information.
Protocol	String	• CLIPBOARDPASTE • KEYINJECTION • MQTT	Contains the protocol that will be used to send the EPC read by the FXP20.
ClipboardPasteDelay	Integer	A value >=0	The delay in milliseconds that will be applied between each clip board paste to let the system process the "Ctrl+V" key event sent to paste the data. This delay is necessary to prevent loss of data. By default, its value is 500ms. Depending on your system performances, you can lower this value up to 100ms. A lower value is not recommended but if your PC is fast enough, you can even disable it by putting it at 0ms.
AutoConnect	Boolean	• true • false	If set to true, the application will automatically connect to the reader when launched using the port defined in the ComPort element. This will work only if a reader is found at the specified port and is available for connexion.
AutoStartReading	Boolean	• true • false	If set to true, the reader will automatically start reading tags when the application gets connected to the reader. This behaviour is executed when the application is launched with auto connect, and when the user manually hits the connect button on the application.

StartMinimized	Boolean	• true • false	If set to true, the application will start minimized in the form of an icon in the Windows toolbar. If AutoConnect is set to true, you may see the application form appears for a limited time (this is a SDK limitation), don't worry, it will minimize itself once the connexion is made.
AddCSVSeparator	Boolean	• true • false	If set to true, a semicolon ";" will be added after each EPC read. This can be useful to create CSV files.
AddCR	Boolean	• true • false	If set to true, a carriage return is added after each EPC read.
AddLF	Boolean	• true • false	If set to true, a line feed is added at the end of each EPC read. This can be useful when dealing with Linux file formats for example.
BeepOnRead	Boolean	• true • false	Set if the reader should beep when a tag is detected.
nbBeeps	Integer	Anything >= 1	Set the number of beeps that the reader should do when a tag is read.
beepSleepTime	Integer	milliseconds	Set the duration used to make the reader beep. Half of this value will be used to make the reader beep. The remaining half will be used to keep the reader quiet. Only one beep can occur at the same time. If the beep sleep time is too long and many read occurs, the application may not beep as much as it should. But since the method is synchronized, normally each call should be placed in a call pipe FIFO and each beep should occurs. Contact the SE who gave you access to this application if you find any problems with this feature.
BatchSize	Integer	A value >= 1	The number of tags read that are extracted from the reader at each read events. Default value is 10. This can dramatically change the performances of the application. Change this value only if you know what you are doing.
HookKeyboardToStartReading	Boolean	• true • false	If set to true, a Keyboard Hook mechanism will take place on the host computer.

			The application will listen to all the keys that are hit by the user. If it intercept a key corresponding to the value of the element HookKeyboardKey, it will start reading tags for the duration defined in the element HookKeyboardReadingDurationInMs.
HookKeyboardKey	Integer	A value >= 0	This parameter contains the ASCII value of printable characters that will be checked by the HookKeyboardToStartReading mechanism. See appendix for more information on ASCII Printable Characters values. The default value is the F12 key.
HookKeyboardReadingDurationInMs	Integer	Milliseconds	The duration of the reading that is executed when the user hit the HookKeyboardKey when the HookKeyboardToStartReading is set to true.
AntennasConfig	Array of antennaConfig	N/A	The start element that contains an array of antennaConfig elements. This will hold the configuration for each antenna. It is mandatory to have one configuration for each antenna with the right Antenna_ID element value. These composite elements are automatically created by the application when it is launched without a Config.xml file. It is the best approach if you want to manually configure the antennas.
antennaConfig	Composite element.	N/A	A composite element that contains the configuration of one antenna. See further elements to understand what is configurable for one antenna.
Antenna_ID	Integer	A number between 1 and 4.	The ID of the antenna that will receive the configuration. FXP20 has 4 possible antennas: • 1: The internal antenna • 2 to 4 : the external antennas.
RFMode_Tari	Integer	A value >= 0	The tari parameter for the selected antenna.
RFMode_TableIndex	Integer	0, 1, 2 or 3	The RFMode value for the selected antenna.
ReceiveSensitivityIndex	Integer	0	The receive sensitivity index.

TransmitPowerIndex	Integer	Value >= 0 and <= 170	See appendix section Transmit Power Index value to match the index with the power value.
TransmitFrequencyIndex	Integer	N/A	This parameter depends on the hardware you are using. On the FXP20 there is only one index reported by the SDK : 1 This index contains the frequencies : • 865700 • 866300 • 866900 • 867500
SingulationControl_Session	String	•SESSION_S1 •SESSION_S2 •SESSION_S3 •SESSION_S4	The Session type used for the inventory.
SingulationControl_TagPopulation	Integer	Value >= 1	The tag population parameter.
SingulationControl_TagTransitTime	Integer	Value >= 1	The tag transit time
SingulationControl_ PerformStateAwareSingulationAction	Boolean	• true • false	If true, the reader will perform State Aware Singulation Action
SingulationControl_ Sl_Flag	String	• SL_ALL • SL_FLAG_ASSERTED • SL_FLAG_DEASSERTED	Set the SL Flags behaviour of the tags.
SingulationControl_ InventoryState	String	• INVENTORY_STATE_AB_FLIP • INVENTORY_STATE_A • INVENTORY_STATE_B	Set the Inventory State behaviour of the tags.
PerformReadingWithGPIO	Boolean	• true • false	If true, the GPI can be used to trigger reading tags. Note that a StartReading must have been performed for this mechanism to work by clicking on its button or sending a MQTT command. No reading will be done until the setup PerformReadingGPIOPort port will be activated. Once the GPI port activated, the reading will last for PerformReadingDuration milliseconds. To quit this mode, the StopReading must be clicked or sent through MQTT.
PerformReadingGPIOPort	Int	A value >= 1	The GPI port that will be used to trigger a reading.
PerformReadingDuration	Uint	A value >= 1	The duration of the reading triggered by the GPI port in milliseconds.

RemoveDuplicates	Boolean	• true • false	This option works for the HookKeyboardToStartReading and PerformReadingWithGPIO mode. If this value is set to true, during the time of the reading session launched by a keyboard hook or a GPI, the EPCs will only be processed once. It means that if you have set the reader in Session S0, when the same tag is read multiple time, only one EPC will be processed for the time of the reading session defined by HookKeyboardReadingDurationInMs or PerformReadingDuration

6. Appendix

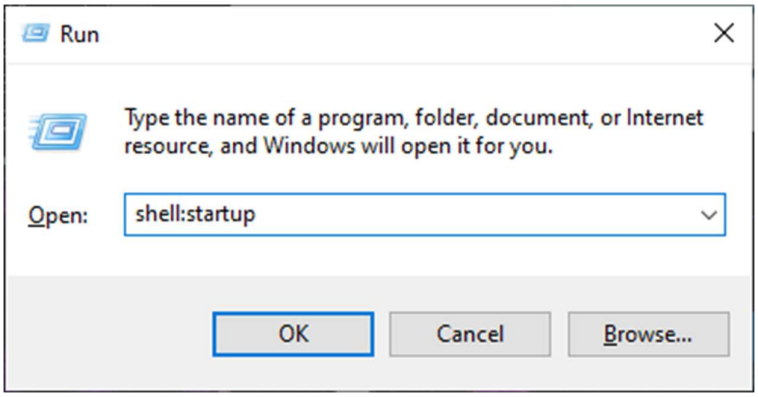
1. Auto-Start application

You can make the application start when the user logs on or when the system starts, connect and start reading automatically.

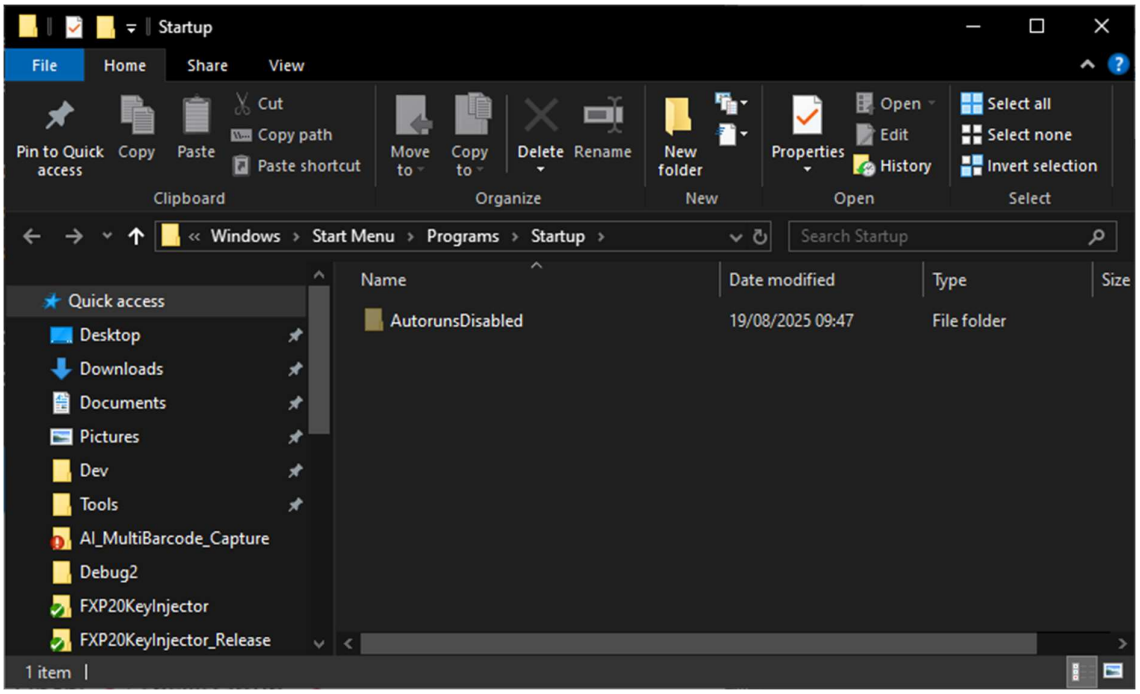
1. Start the application when the user logs on

Hit Windows + R and type the following:

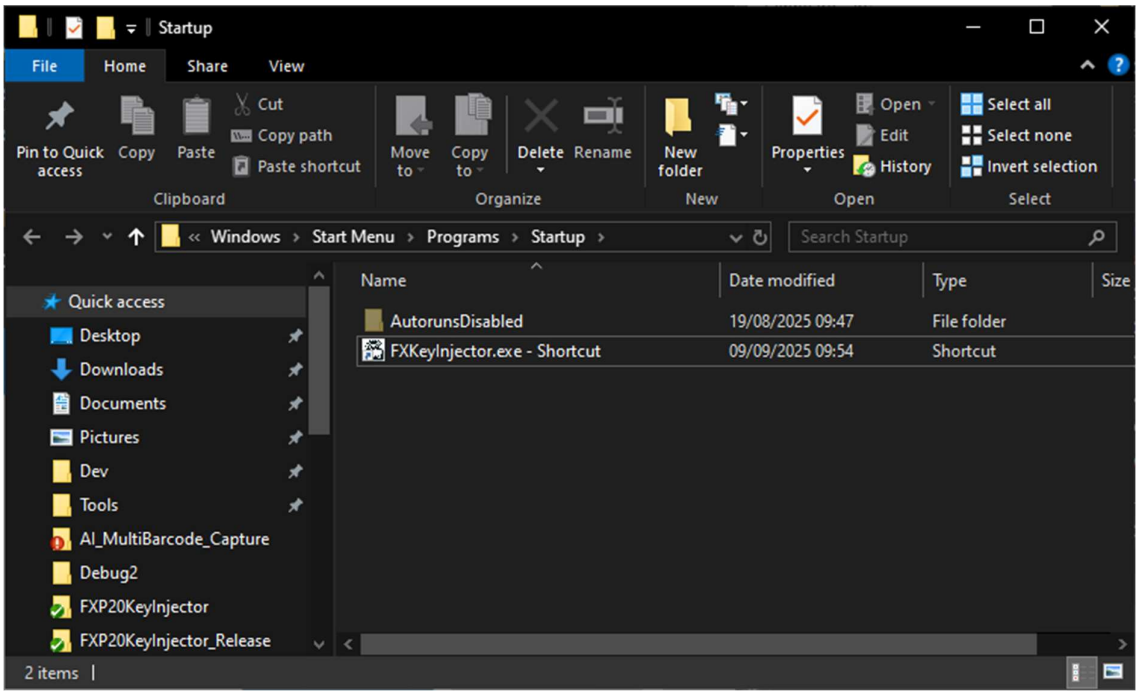
shell:startup



This will open your startup menu folder:

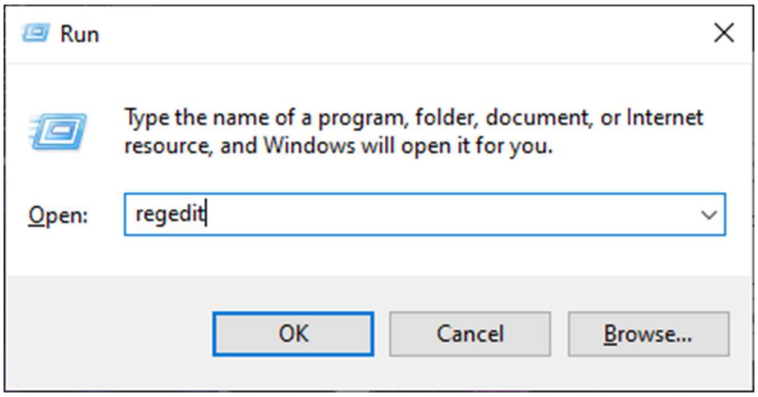


Simply add a shortcut to FXP20KeyInjector in this folder to make it run automatically when the user log on.



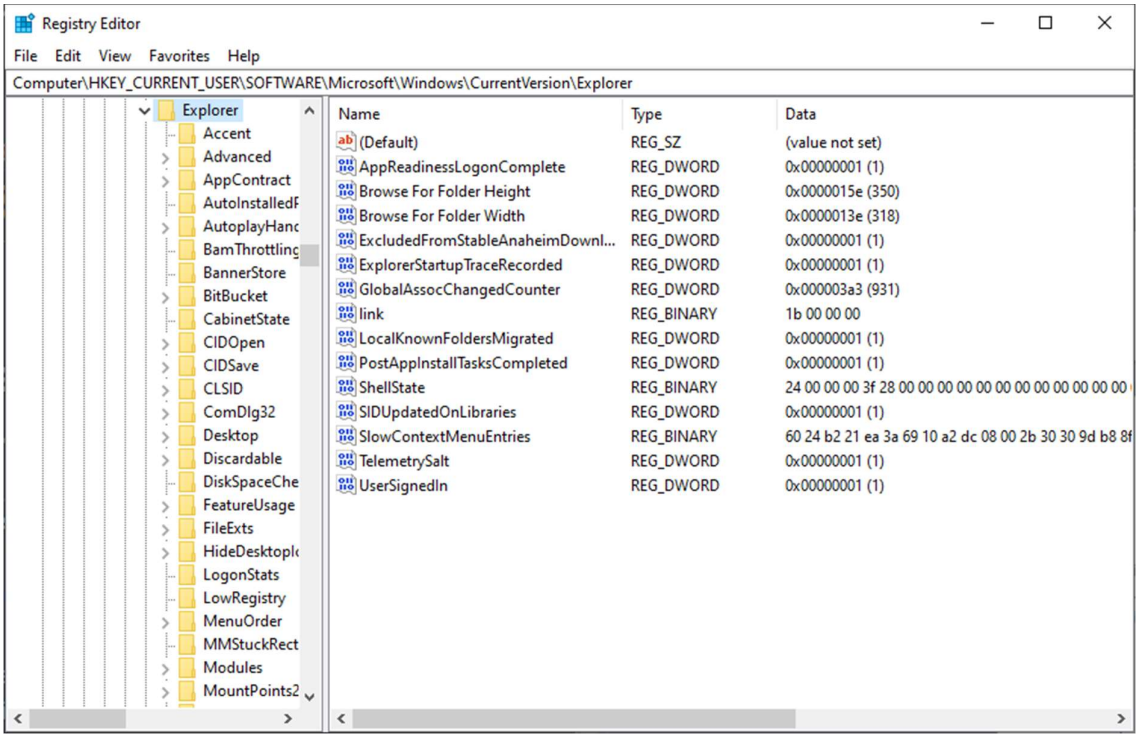
If you use this method you may want to reduce the startup delay of windows.

Hit Windows + R, and start the registry editor.



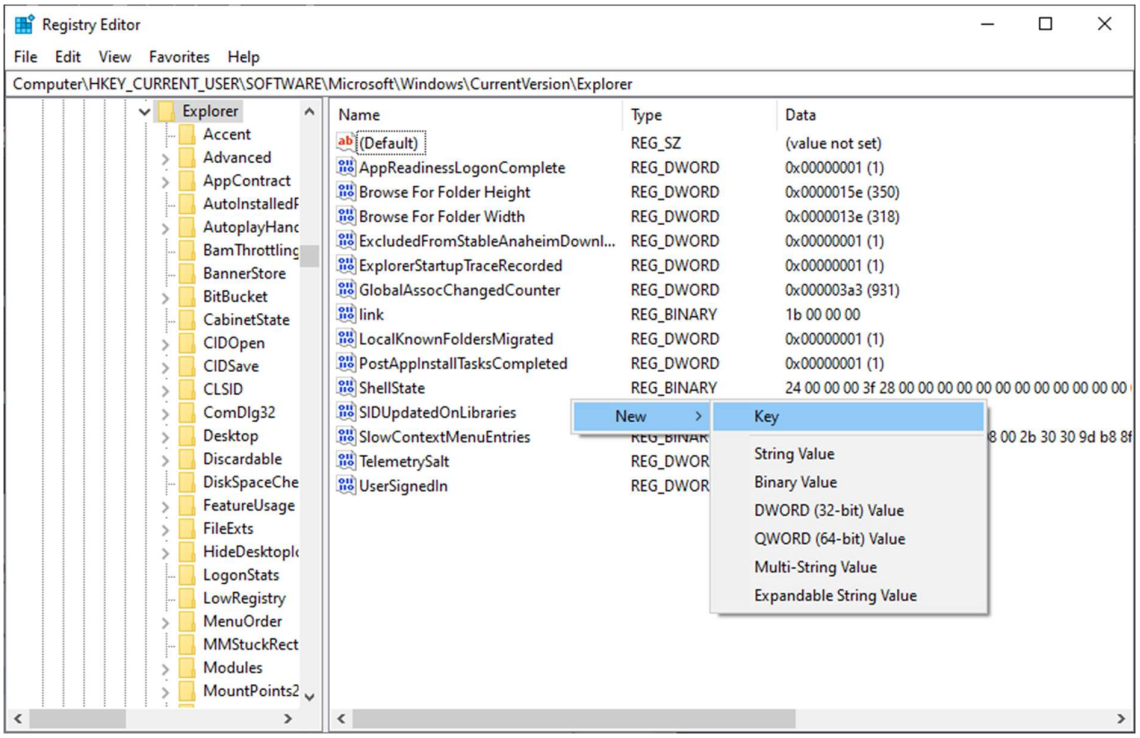
Copy/Paste this path into the textbox below the File/Edit/View… menu:

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\

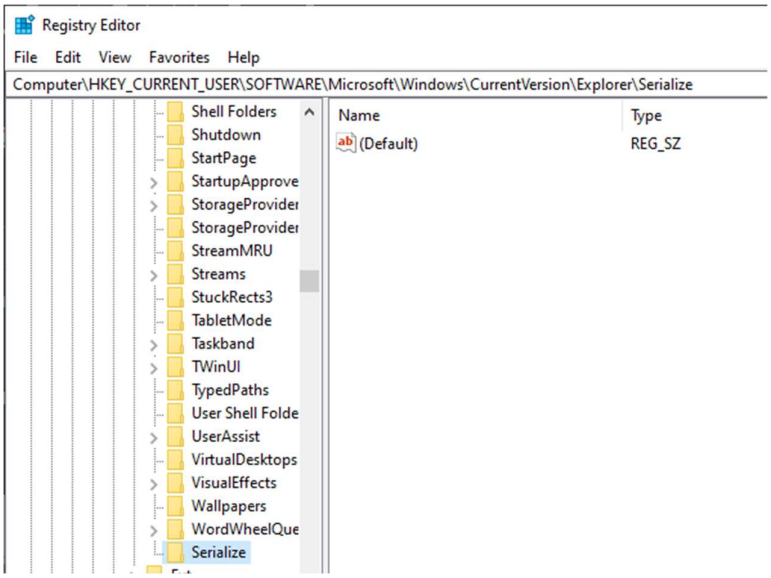


If there is no Serialize Key, create it.

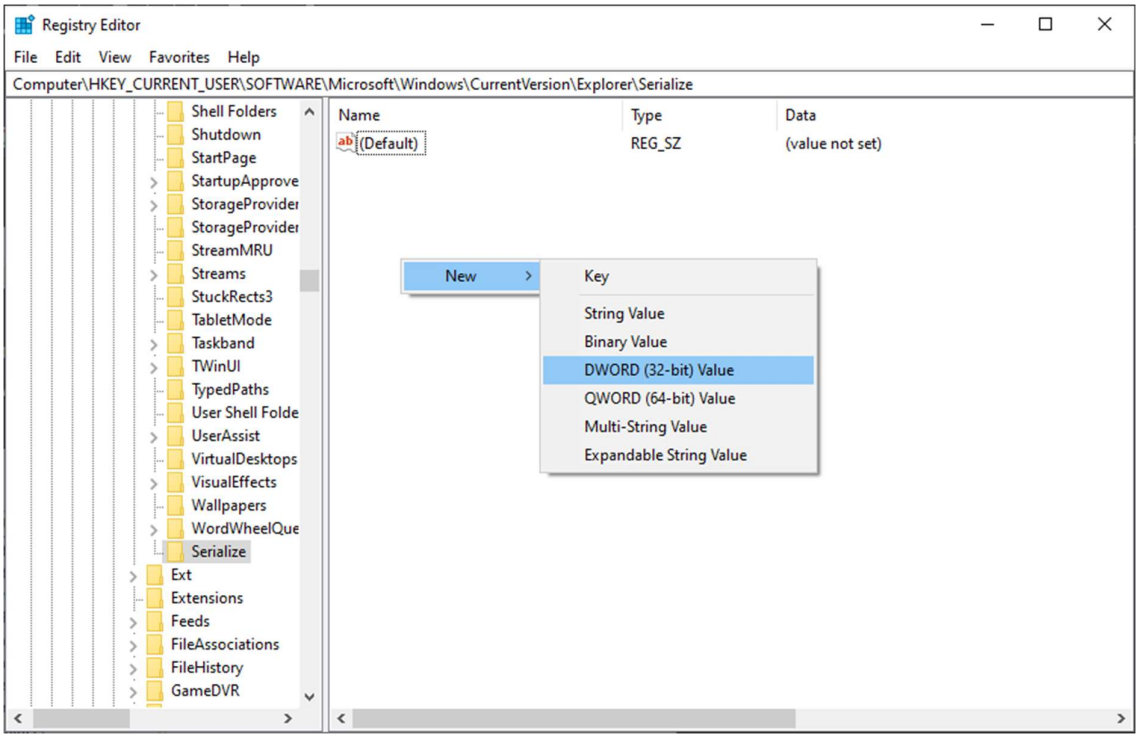
Right click on the list and select key in the popup menu:



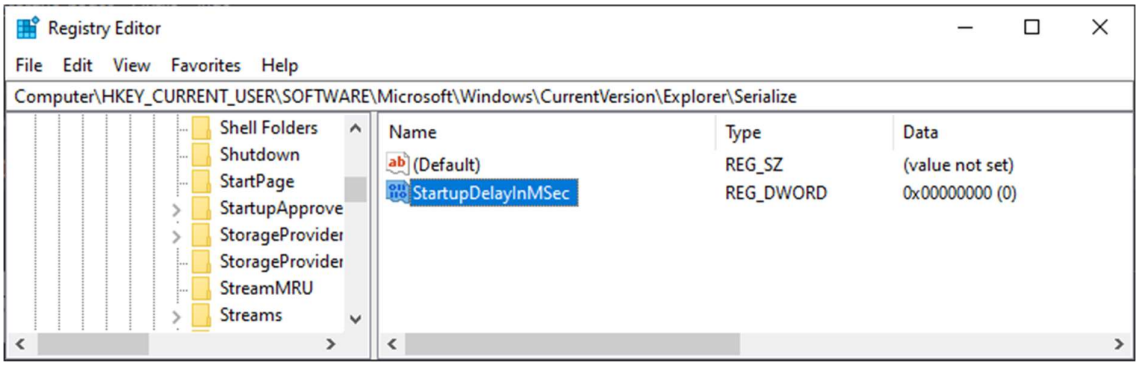
Name the key: Serialize



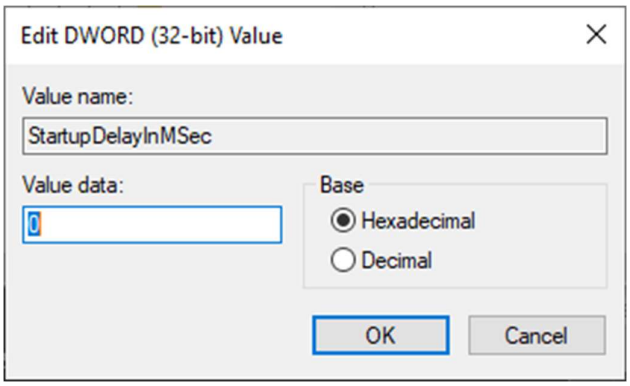
Create a new DWORD 32 bits value:



Name it : StartupDelayInMSec



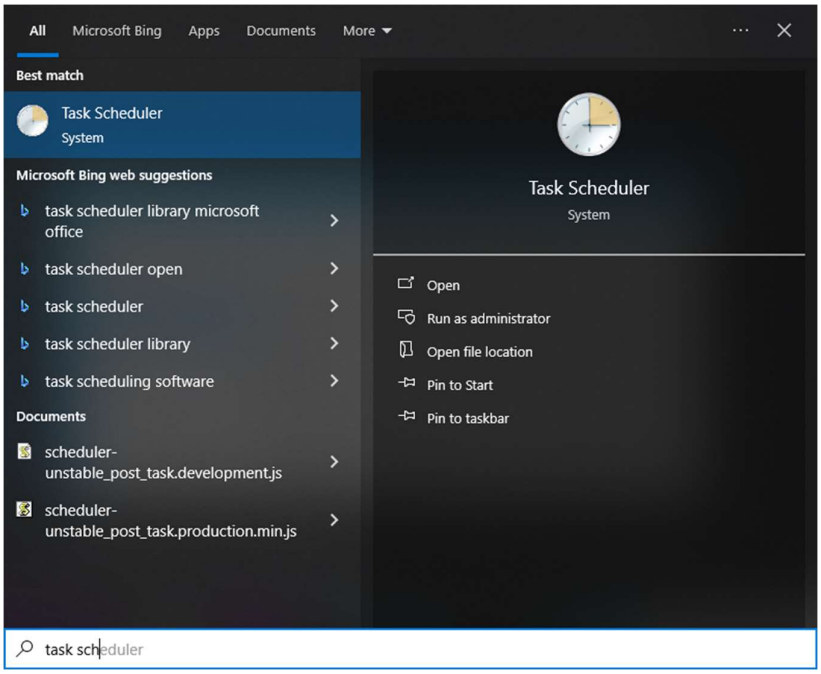
Double click on the newly created key and set its value to 0



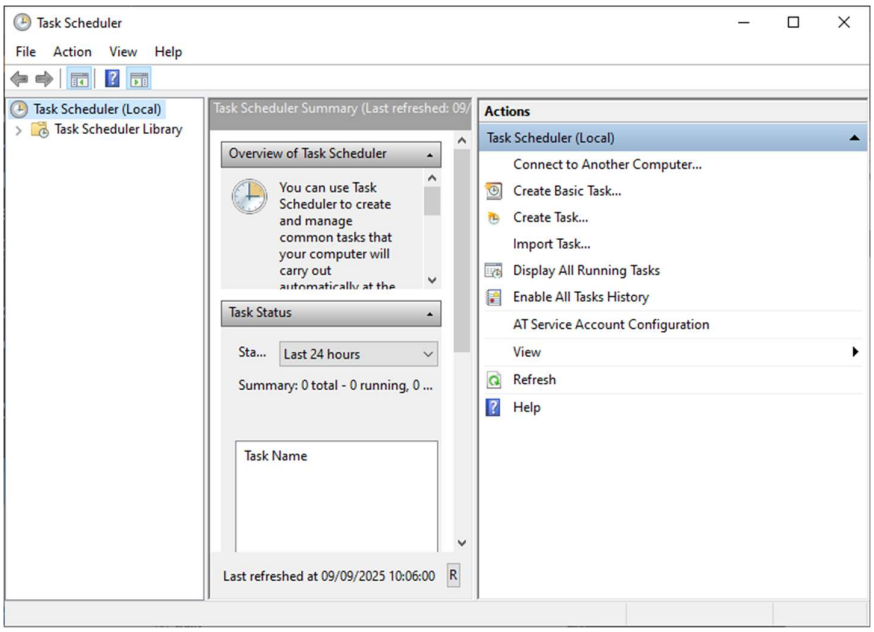
2. Start the application with Task Scheduler

You can use the Task Scheduler to start the application automatically.

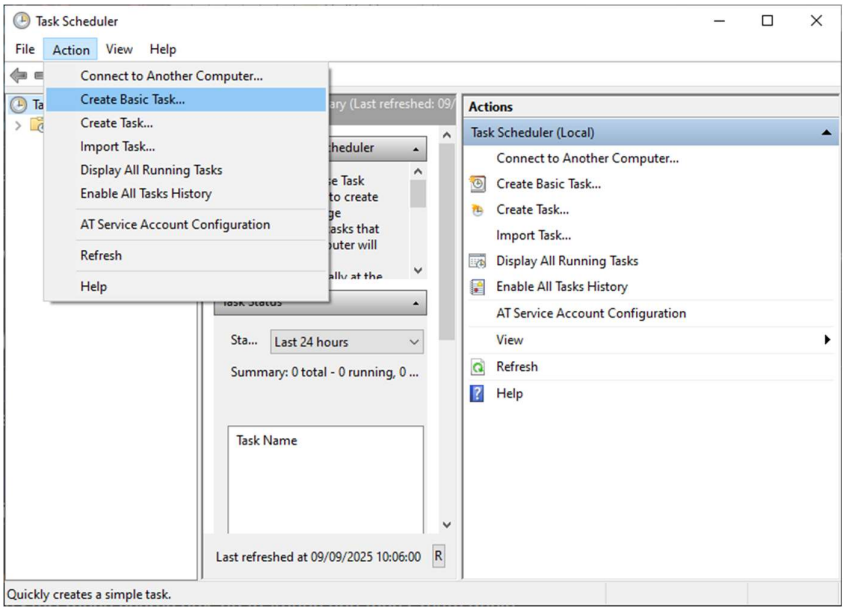
Use the search box to find the Task Scheduler.



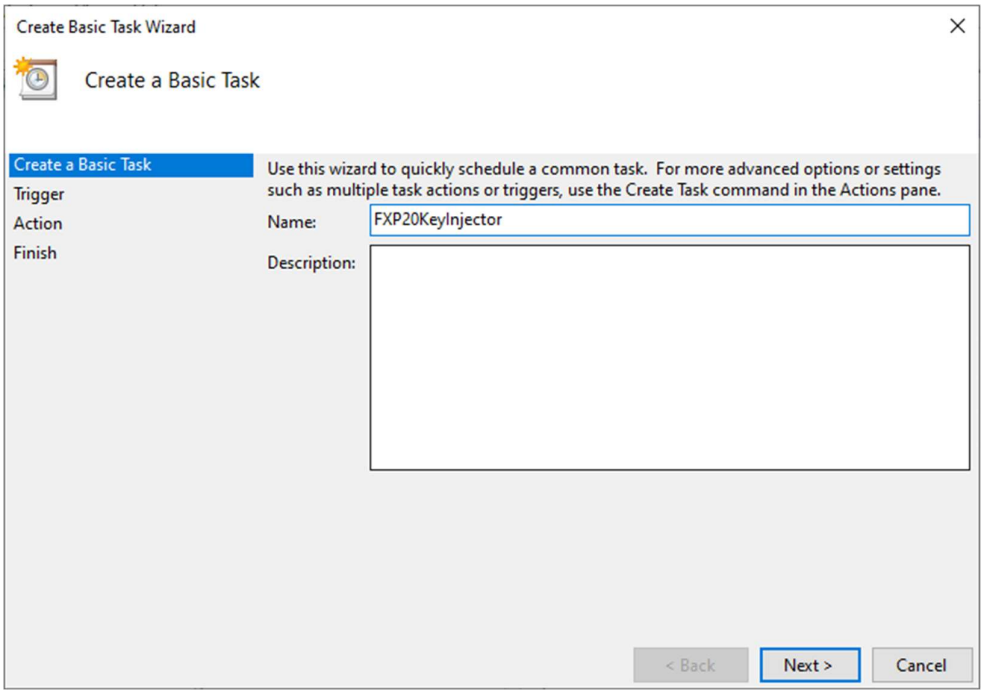
Open it.



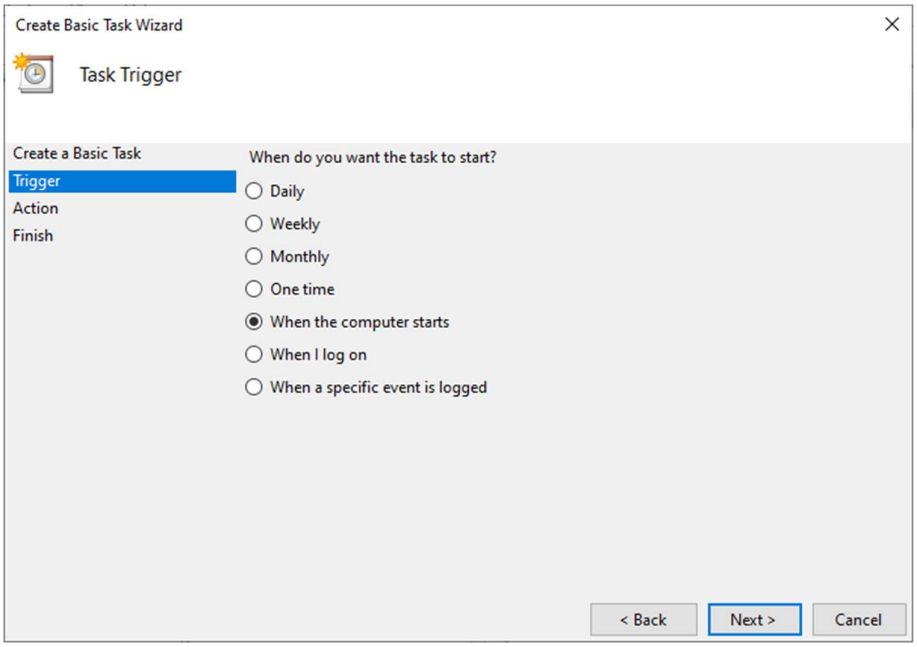
Go to Action and select Create Basic Task:



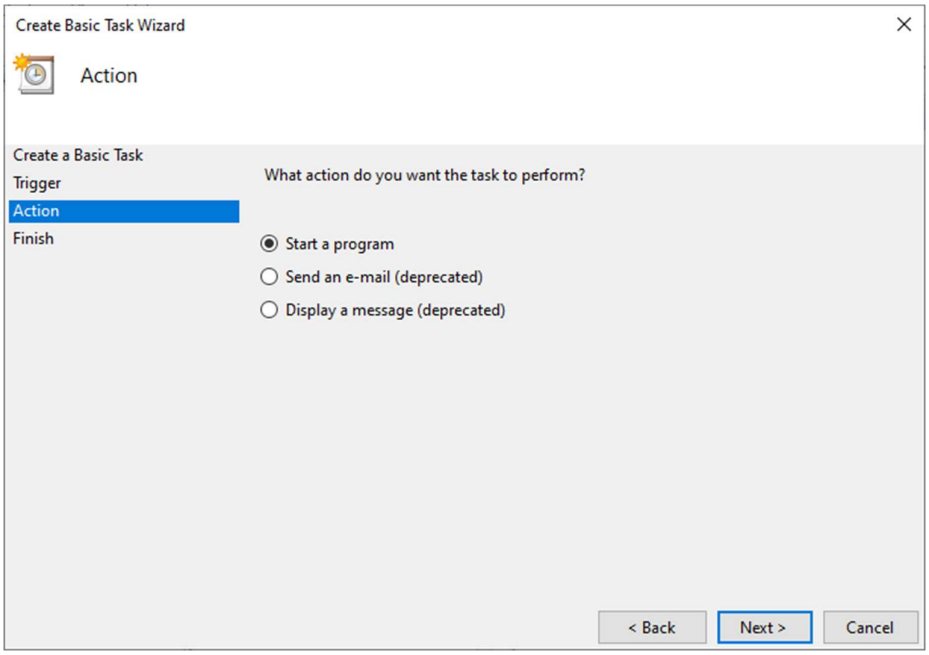
Follow the wizard, first enter the name of the task:



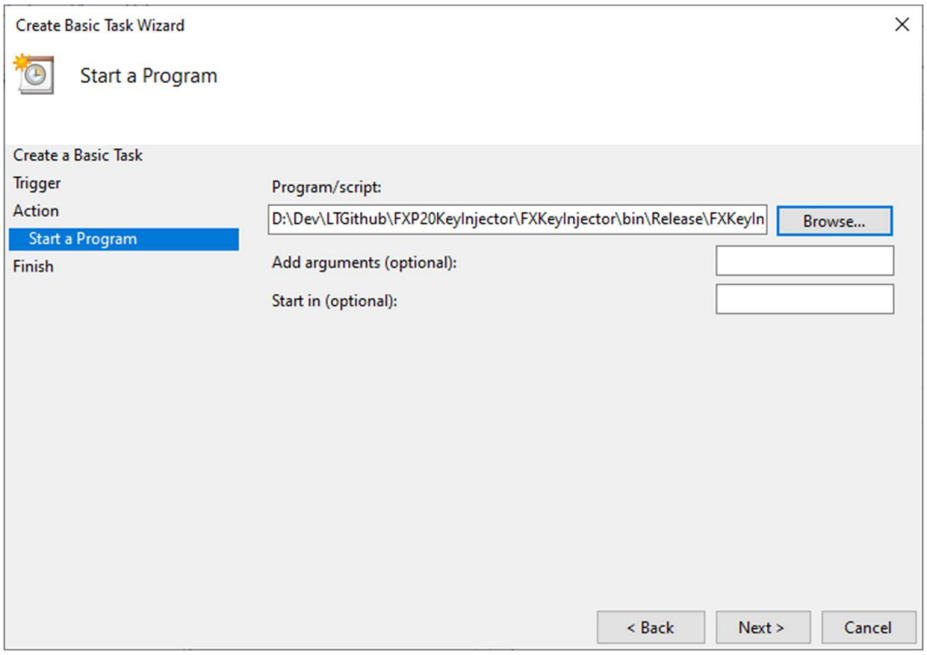
Set a trigger for the task:



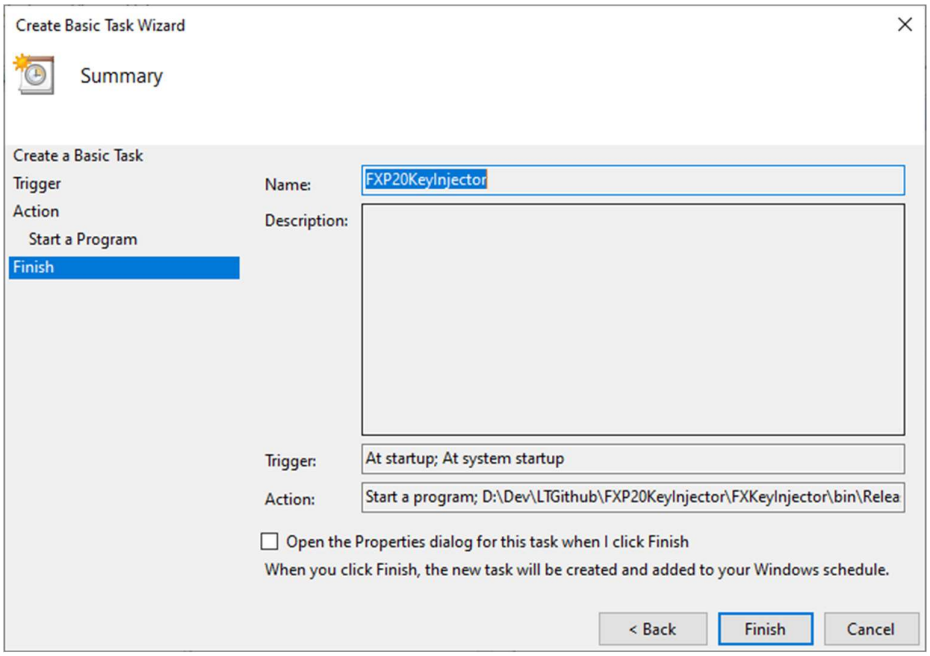
Then set an action:



Then select the location of the FXP20 executable:



Review the parameters and click on finish:



3. Manage the FXP20 Startup parameters

You have many options when starting the application that can be configured using the Config.xml file.

- StartMinimized element will make the application start in minimized form <StartMinimized>true</StartMinimized>
- AutoConnect will connect the app to the reader automatically using the COM port defined in the Config.xml. Please ensure that you configured this option correctly otherwise the

application will not connect to the reader.
<AutoConnect>true</AutoConnect>

- You can start reading automatically on connection to with the element: AutoStartReading. Used in with Autoconnect, this will ensure that the reader automatically start reading when the application is launched.
<AutoStartReading>true</AutoStartReading>

With all these parameters, you can automatize the launch of the application and the reading of tags.

2. ASCII Printable Characters

- **0-9**: '0' to '9' are 48 to 57
- **A-Z**: 'A' to 'Z' are 65 to 90
- **a-z**: 'a' to 'z' are 97 to 122
- **Common Symbols**:
    - Space: 32
    - Exclamation Mark (!): 33
    - Double Quote (''): 34
    - Hashtag (#): 35
    - Dollar Sign ($): 36
    - Percent (%): 37
    - Ampersand (&): 38
    - Single Quote ('): 39
    - Left Parenthesis (() : 40
    - Right Parenthesis ()): 41
    - Asterisk (*): 42
    - Plus (+): 43
    - Comma (,): 44
    - Minus (-): 45
    - Period (.): 46
    - Slash (/): 47
    - Colon (:): 58
    - Semicolon (;): 59
    - Less Than (<): 60
    - Equals (=): 61

- Greater Than (>): 62
- Question Mark (?): 63
- At (@): 64
- Left Bracket ([): 91
- Backslash (): 92
- Right Bracket (]): 93
- Caret (^): 94
- Underscore (_): 95
- Grave Accent (`): 96
- Left Brace ({): 123
- Vertical Bar (|): 124
- Right Brace (}): 125
- Tilde (~): 126

4. Virtual-Key Codes:

In Windows, keys can also be represented by virtual-key codes, which are used for identifying keys in system-level programming. Here's a list of some of the common ones:

- **Function Keys**:
  - F1: 112
  - F2: 113
  - F3: 114
  - … up to …
  - F12: 123
- **Control Keys**:
  - Enter: 13
  - Backspace: 8
  - Tab: 9
  - Shift: 16
  - Control: 17
  - Alt: 18
  - Escape: 27
  - Delete: 46

- Arrow Keys:
  - Left Arrow: 37
  - Up Arrow: 38
  - Right Arrow: 39
  - Down Arrow: 40

3. Transmit Power Index

Index	Power Value
0	1000
1	1010
2	1020
3	1030
4	1040
5	1050
6	1060
7	1070
8	1080
9	1090
10	1100
11	1110
12	1120
13	1130
14	1140
15	1150
16	1160
17	1170
18	1180
19	1190
20	1200
21	1210
22	1220
23	1230
24	1240
25	1250
26	1260
27	1270
28	1280
29	1290
30	1300
31	1310
32	1320
33	1330
34	1340
35	1350
36	1360
37	1370
38	1380

39	1390
40	1400
41	1410
42	1420
43	1430
44	1440
45	1450
46	1460
47	1470
48	1480
49	1490
50	1500
51	1510
52	1520
53	1530
54	1540
55	1550
56	1560
57	1570
58	1580
59	1590
60	1600
61	1610
62	1620
63	1630
64	1640
65	1650
66	1660
67	1670
68	1680
69	1690
70	1700
71	1710
72	1720
73	1730
74	1740
75	1750
76	1760
77	1770
78	1780
79	1790
80	1800
81	1810
82	1820
83	1830
84	1840
85	1850
86	1860
87	1870
88	1880

89	1890
90	1900
91	1910
92	1920
93	1930
94	1940
95	1950
96	1960
97	1970
98	1980
99	1990
100	2000
101	2010
102	2020
103	2030
104	2040
105	2050
106	2060
107	2070
108	2080
109	2090
110	2100
111	2110
112	2120
113	2130
114	2140
115	2150
116	2160
117	2170
118	2180
119	2190
120	2200
121	2210
122	2220
123	2230
124	2240
125	2250
126	2260
127	2270
128	2280
129	2290
130	2300
131	2310
132	2320
133	2330
134	2340
135	2350
136	2360
137	2370
138	2380

139	2390
140	2400
141	2410
142	2420
143	2430
144	2440
145	2450
146	2460
147	2470
148	2480
149	2490
150	2500
151	2510
152	2520
153	2530
154	2540
155	2550
156	2560
157	2570
158	2580
159	2590
160	2600
161	2610
162	2620
163	2630
164	2640
165	2650
166	2660
167	2670
168	2680
169	2690
170	2700

4. Mosquitto MQTT setup and sample files

If you plan to use the MQTT feature, it is advised to use Mosquitto MQTT Server as it is one of the best options you'll find on the market.

This guide will help you setup Mosquitto MQTT to work with FXP20KeyInjector when using a PC with Windows 10/11/Server.

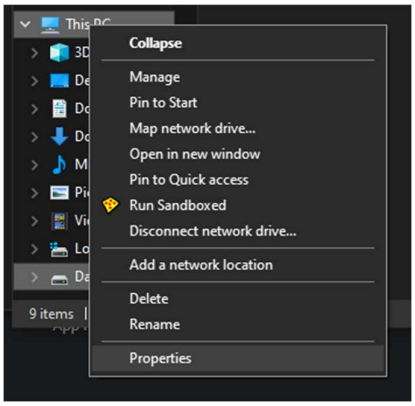
The first step is to download the latest version of Mosquitto MQTT Server:

https://mosquitto.org/download/

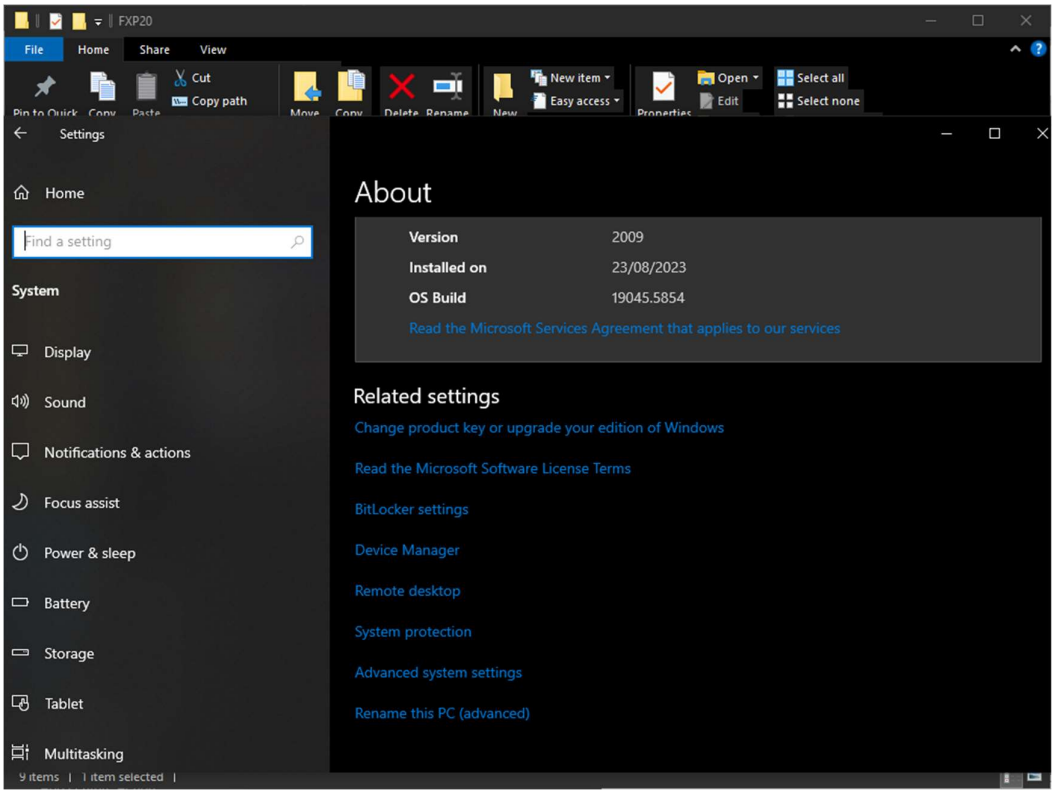
Then install it on your computer.

It is strongly recommended to add the Mosquitto binary location to the path of your computer.

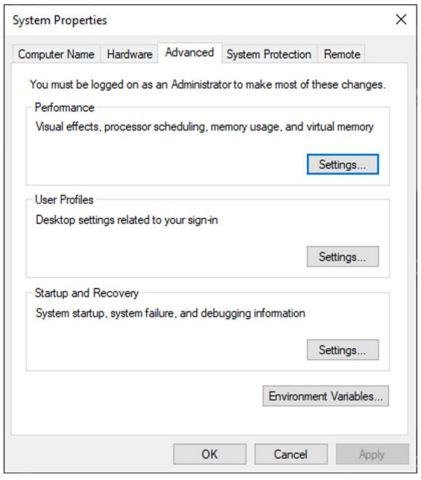
For that, right click on your PC icon and select Properties.



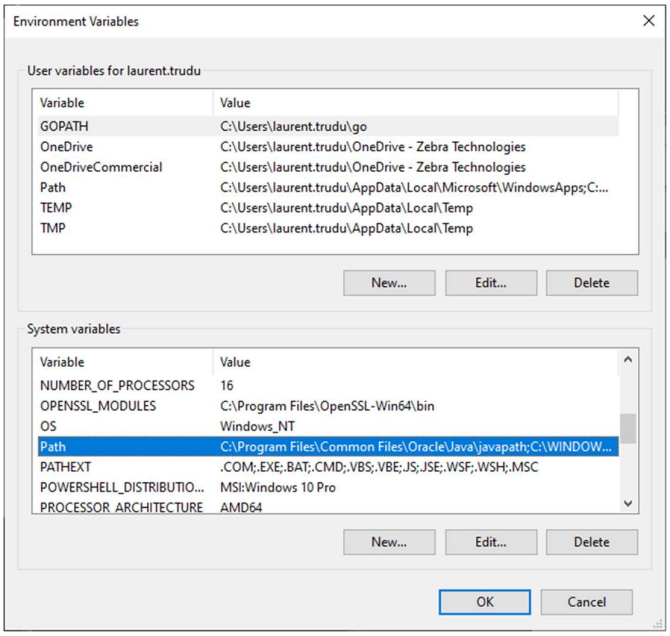
Then open the Advance System Settings:



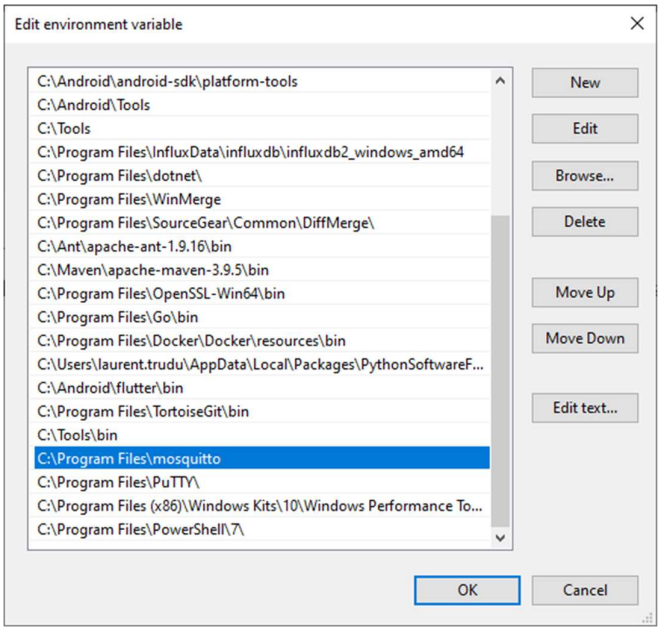
Select Environment Variables:



Then double click on the Path entry in system variables to edit it:



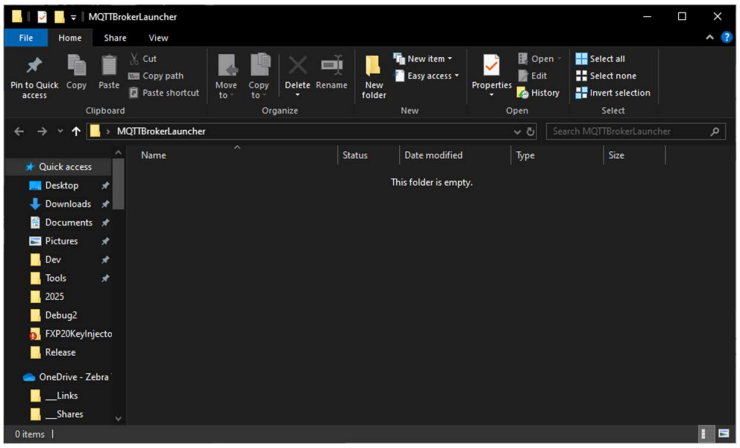
Now you can add the path of your Mosquitto installation to the system Path:



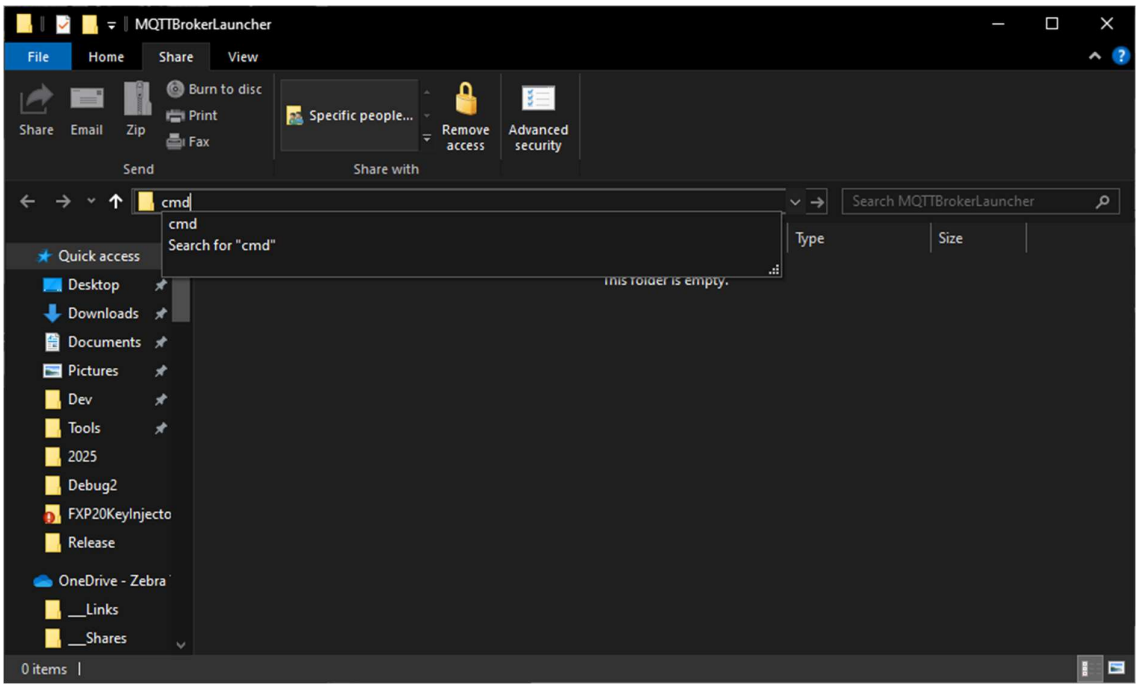
Once the path has been added, create a new folder on your computer.

Let’s name it MQTTBrokerLauncher.

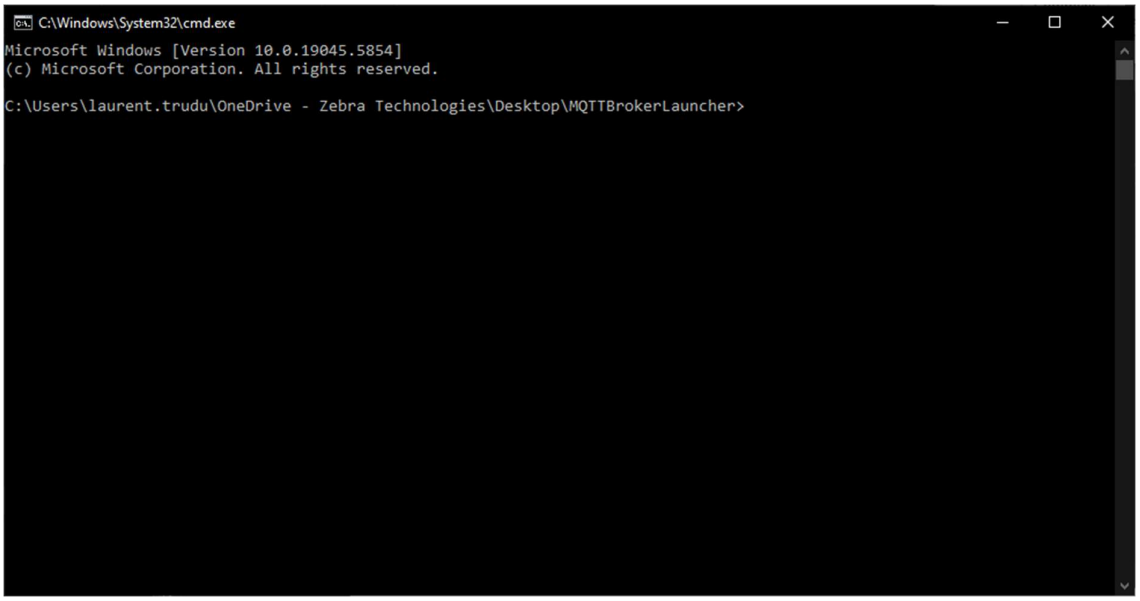
Open it with explorer.



Then type cmd in the file path input control :



Hit the key “enter” and it will open a command prompt directly at the path of the folder.

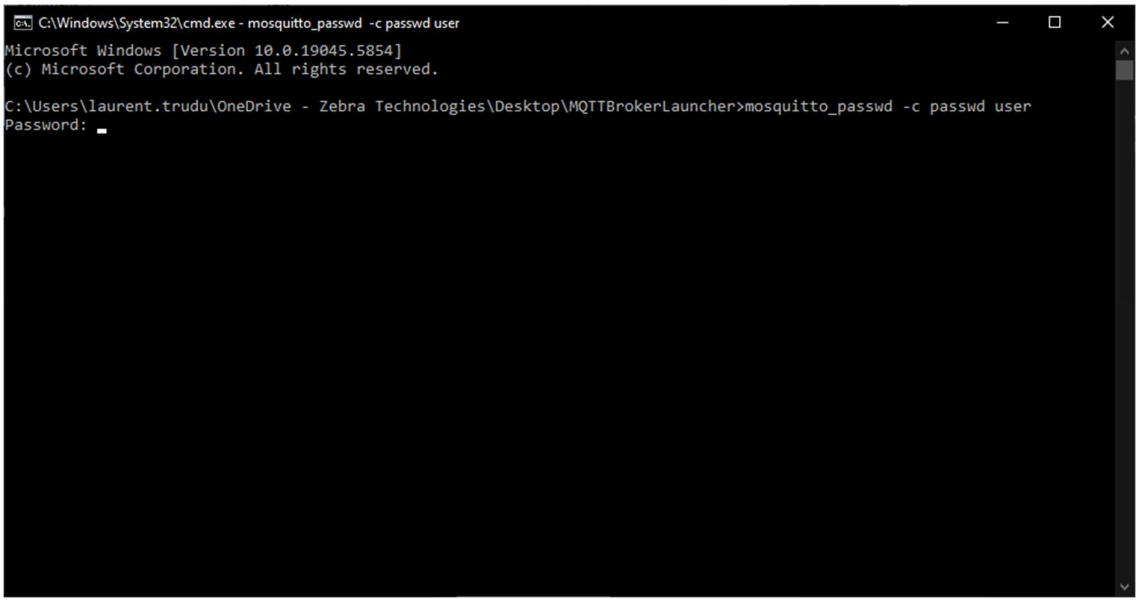


Then you can create a Mosquitto passwd file containing a user login and password.

To create this file for a the username “user” just enter the following command:

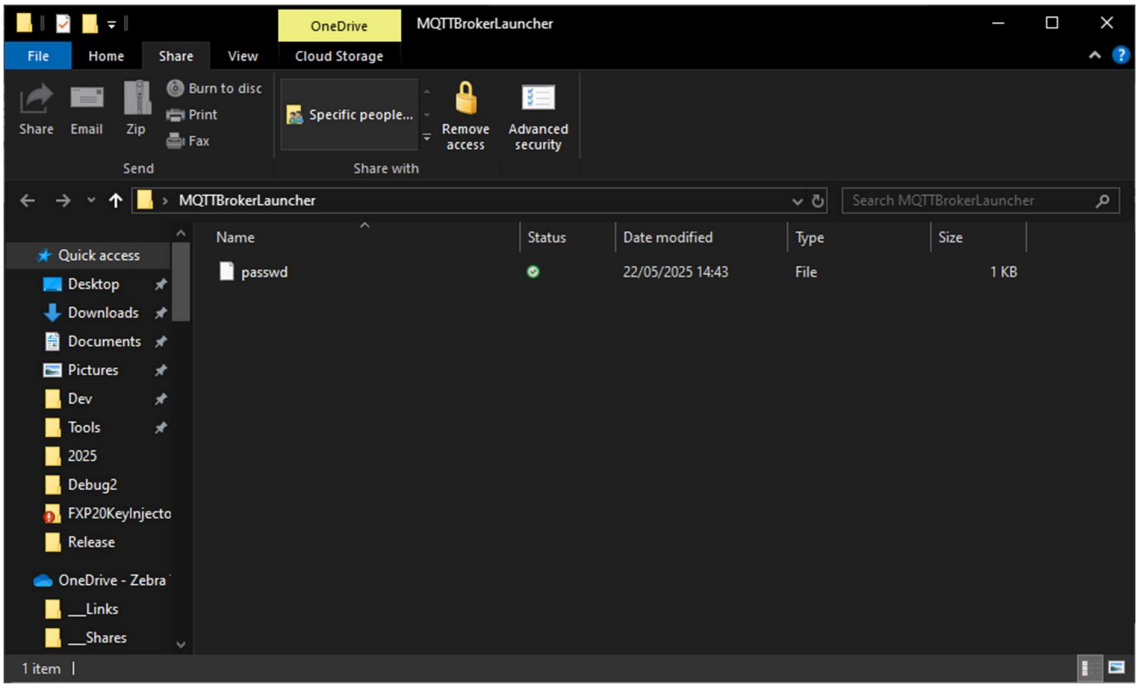
mosquitto_passwd -c passwd user

This will ask you for a password:



Let’s use the password “sko” (as it is the default password for the config.xml file)

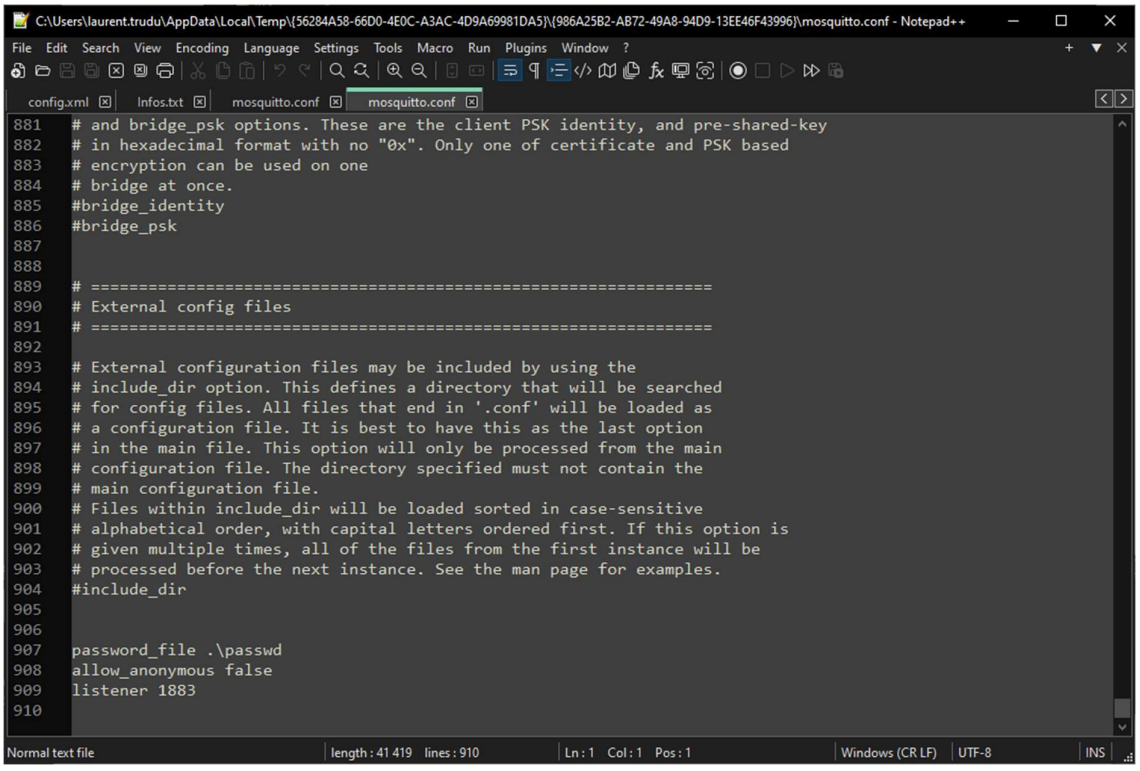
Once you have entered the password twice, the file passwd will be created in the folder.



You can now add a mosquito.conf file to your folder.

mosquitto.conf

This file contains the necessary configuration to run Mosquitto server with username and password.



This file will setup Mosquitto server to use the passwd file you created previously and set the listening port to 1883.

You can directly add this file to your folder as we will use it to run the Mosquitto server.

To run the Mosquitto server, just create a batch file (ex: runMosquitto.bat) and copy/paste the following command into it :

mosquitto -v -c "mosquitto.conf"

runMosquitto.bat

You have now a batch file that will run the Mosquitto server with the specified login/password : user/sko in localhost mode.

This server can be used to relay the EPC read by the FXP20 to your application.

You can use it to manage the reader through the control topic.