afero

Tech Note

Factory Provisioning Guide for nRF52 Products

January 15, 2019 - Version 2.3

Introduction

This document describes the factory provisioning process for devices employing the nRF52 module. Once your product has gone through this process, it will be ready to communicate with the Afero Cloud and the Afero mobile application.

Note that devices programmed with a **production** device profile are no longer considered "development" devices. This means you won't be able to use the Afero Profile Editor to change how the final product ("production device") behaves. Changing a production device's profile requires re-flashing the device or pushing a new profile over-the-air (OTA).

The Afero Customer Enablement (ACE) team will work closely with your designers during the Prototype phase to ensure a smooth transition from Prototype to Production.

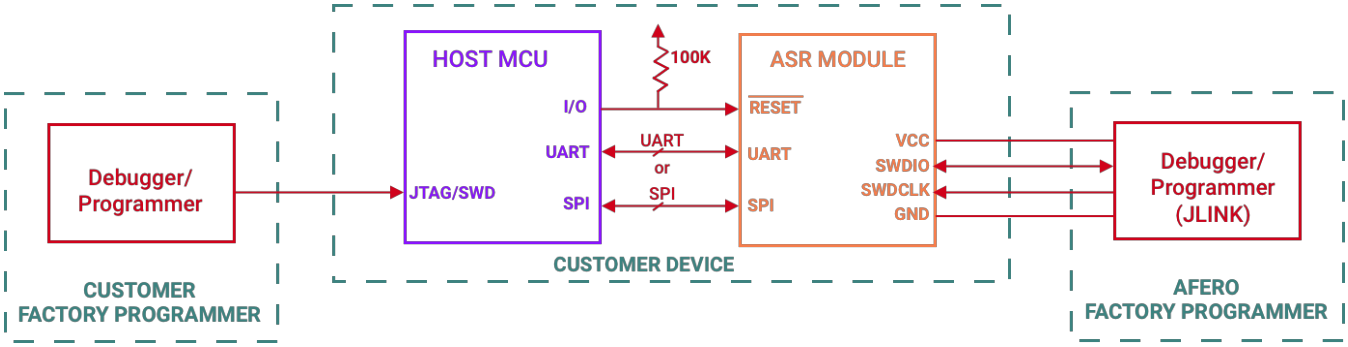
Prerequisites

Prerequisites to the process are as follows:

1. The device profile binary created in the Prototype phase is final and **signed** by Afero. A signed device profile is ready for production and cannot be altered or corrupted; the process uses a cryptographic hash to validate authenticity and integrity.
2. The Afero Firmware Specification, "Preferred Pin Mappings for Nordic nRF52 BLE Chipsets", PN D-LIT-00065-00.
3. An FTP site has been established to facilitate file sharing between your manufacturing team and Afero. Make sure you have received the FTP directory name and account credentials from Afero.

Factory Programmer Block Diagram

The diagram below illustrates the connections for the host MCU, ASR module, and the associated factory programmers:



Step 1. Design Your Board for Mass Production

Your final product board layout must accommodate the Afero provisioning requirements described below:

- The circuit design must expose the nRF52 programming pins used to load the production Afero device profile and latest firmware. The relevant pin functions are shown below; to find the actual pin numbers, please use the Afero Firmware Specification, "Preferred Pin Mappings for Nordic nRF52 BLE Chipsets", PN D-LIT-00065-00 to map the Pin Functions shown below to the actual Pin Locations for your chipset manufacturer:

PIN FUNCTION
VDD
VSS (GND)
UART RX (Optional – Supports Profile Flash and RF Test Mode)
UART TX (Optional – Supports Profile Flash and RF Test Mode)
SWDIO
SWDCLK

- Make sure that the connections (e.g., pads) on your circuit board are physically accessible so your provisioning fixture can connect to them easily in the factory.
- You may decide to expose programming pins on an internal circuit board, or on the outside of your product's enclosure. Your final design should consider the overall cost of production and the service life of the product.
- Deciding to expose test pins outside the enclosure adds expense, but it allows you to program, label, and test the fully-assembled product in the factory. This strategy also supports testing in a service center without disassembling a product that has been returned.

Step 2. First-Time Setup

1. Set up the PC and required software.
   a. You will need a PC running Windows 7 or later.

   > **Important!** Check the name of your PC to make sure it does **not** contain any non-Unicode characters. Only UTF-8 characters are supported by the AFP so if your machine name does not comply with this requirement, please change the name of your Windows PC. You can check and edit the name by clicking the lower-left Windows menu, then selecting (Settings) > System > About > Rename this PC.

   b. Install the latest version of Python 2.7 from https://www.python.org/downloads/. (On the date of this writing the latest is Python 2.7.15 with this direct link: https://www.python.org/downloads/release/python-2715/.)
   c. Add `c:\python27` and `c:\python27\scripts` to the environment path variables. For help on how to do this on Windows, please go to Adding a Path to Windows Environment Variables.
   d. Open Windows Command Prompt with administrator privilege and run these two commands:

   ```
   pip install pip -U
   pip install pyserial
   ```

   e. Install 7-Zip from https://7zip.org.
   f. Install Notepad++ from https://notepad-plus-plus.org.
2. Set up the factory provisioning fixture and debugger/programmer (such as a SEGGER J-Link, which is used as an example in the instructions that follow).
   a. The factory provisioning fixture is a jig that is connected to the PC and that 1) supplies power, and 2) creates a physical connection between the nRF52 and a PC running a Python script. The provisioning fixture must supply 2.4 – 3.5V DC to VDD, and Ground to GND.
   b. The SEGGER J-Link debugger/programmer communicates with the provisioning fixture to flash the firmware images. The J-Link is connected to the SWDIO and SWDCLK pins on the nRF52. The SEGGER J-Link also monitors VDD and GND. (See https://www.segger.com/products/debug-probes/j-link/models/j-link-base/.)

      To install the J-Link driver go to https://www.segger.com/downloads/jlink/#j-linksoftwareanddocumentationpack and download "J-Link Software and Documentation Pack for Windows". After installation, be sure to add J-Link to your environment path variable.

3. Set up the label printer.
   a. You will need to procure a printer that connects to the PC for generating the device labels, which includes the QR code. Standard label sizes are:
      - Labels that are placed on the module are **0.5-in tall by 1-inch wide**.
      - Labels placed on the appliance or device are **30mm tall by 45mm wide**.

      Recommended (supported) label printers are both 300dpi:
      - Zebra Gx430t, or
      - Zebra 105SL Plus

      > **Note!** Be sure to install the correct Zebra print driver on Windows and verify that the printer appears in Settings > Printers and Scanners. Print a test page.

   b. Customize the appropriate `.json` printer label file, included in the unzipped factory package `\targets\`*`ProgrammerName`*`\labels\` subdirectory. The files are named by printer model. Read details about default settings and the values you must customize in Customizing the Label below.

Step 3. Program the ASR Module

> **Important!** Afero recommends that you program the ASR module **first** to ensure that the host MCU is not running code that will hold ASR in RESET or otherwise interfere with the ASR programming step.
>
> Most MCUs are designed with the I/Os floating; as a result, the ASR RESET line between the MCU and ASR will float if the MCU program has been erased or is held in reset. An external pullup on the ASR RESET line is a good practice.

Follow the steps below to program your ASR module:

1. Connect the provisioning fixture to a switchable power supply and your PC. The provisioning fixture will interface with the assembled board, supplying power to and communicating with the nRF52.
2. Use 7-Zip to unzip the encrypted file `afp2-`*`ProgrammerName-BuildNumber`*`.zip` to a convenient location on your C: drive. To unzip, highlight the zip file, then right-click to select 7-Zip> Extract to `afp2-`*`ProgrammerName-BuildNumber`*. You will need a password to unzip the file, which Afero will give you.
3. Customize the `config.json` file, located in the `targets\`*`ProgrammerName`*`\` directory in the unzipped factory package. Use Notepad++ to edit the file.

   Fields that must be customized are marked with the value `CHANGE_ME`. These customized values and required and will appear in the factory database record.
4. Complete the printer setup:
   a. Open a command line window and navigate to the folder in which the programmer package was unzipped.
   b. Within that directory, go to the `core\util` directory by typing:

      ```
      cd core\util
      ```

c. You will find printer configuration scripts (batch files) used to map the Zebra printer to LPT1:
   - `setup-zebra-105SL.bat`
   - `setup-zebra-gx430t.bat`

> **Note:** If you are running the programmer on a non-English PC, you must edit the appropriate batch file to reference the correct path to the `prncntg.vrb` file, given your locale.
>
> For example, this line in the batch file contains the **en-US** locale in the path:
>
> ```
> cscript C:\Windows\System32\Printing_Admin_Scripts\en-US\prncnfg.vbs -t -p "ZDesigner
>   GX430t" +shared -h ZEBRA300
> ```
>
> Make sure your printer's batch file references the locale being used on your PC.

d. Run the appropriate script for the printer that the factory will be using.

> **Note:** If the printer you are using is not one of the two listed, the .bat scripts can be copied and the name changed to match the printer being installed; **however, the label configuration files referenced above in Step 2 > Substep 3b may not be compatible with your printer.**

5. Connect the assembled board to the provisioning fixture.
6. Apply power to the fixture and run the Factory Provisioning batch file script from the command line in the unzipped AFP2 directory:

   `afp-`*`ProgrammerName`*`.bat`
7. You will receive a "PASS" prompt when the programming is completed. You can then power down the provisioning fixture and remove the board.
8. A corresponding QR code will emerge from the QR code printer.
9. The script creates a `.afero` factory file that contains device-specific values. You will find this file in the `_output` directory. Read more about what you will need to do with this file in Step 6 below.

Step 4. Program the Host MCU

Follow the steps below to program the host MCU *after* programming the ASR:

1. Program the host MCU according to the relevant process.
2. Perform factory testing for the host MCU.

> **Important!** If you must program the host MCU before ASR (preference is to program ASR first), the code on the host MCU **must be designed so the ASR RESET line is not held low during ASR programming**.

Step 5. Perform Post-Provisioning Label Validation Test

Perform a post-provisioning test to ensure the QR label on the product is ready for the end user to scan.

The AFP2 script outputs the Association ID and Device ID each time a device is provisioned successfully. The operator should compare the on-screen values to the printed QR code to ensure an accurate device-to-label match.

Step 6. Register the Device with the Afero Cloud

1. The provisioning script will generate a factory file in text format in the `_output` directory named:

   `01_`*`ComputerName_ProgrammerName_YYYYMMDDTimeZone`*`.afero`.

   The 01 designates the Factory File Type and will always be `01` for this output file. This file is comma-separated and contains the fields listed below by default:

   - profile version id
   - profile fw type
   - bootloader version id
   - bootloader fw type
   - app version id
   - app fw type
   - wifi version id
   - wifi fw type
   - device id
   - association id
   - company code
   - ble mac

   Assuming you have customized the `config.json` configuration file as part of Step 3 > Sub-step 3 above, the modified fields will also be included in the file:

   - partNo
   - manufacturer
   - factory
   - station id
   - hw version
2. Post this `.afero` file to an FTP site in a directory specified by Afero.
3. Once the file has been processed by Afero, it is **moved** to a `Completed` directory; the factory is left with a record of devices in a separate FTP folder.

Appendices

Customizing the Label

The QR code contains an Association ID that is used by the Afero mobile app to onboard the device. It is critical that the correct QR code label be physically applied to the nRF52 that generated it.

Every label must include the following two pieces of information:

- QR code, and
- Plain-text version of the Association ID

For more technical QR code information, please refer to the Afero Developer Portal, QR Codes for Afero Products page.

The AFP2 Factory Provisioning package includes `.json` files that define the elements of the QR code + text label. The reference code in these files is compatible with Zebra printers (ZPL format) and is commented for ease of use.

By default, the following configuration is defined in the reference code:

- Standard label sizes: 1.5-inch x 0.5-inch (module label) in 300dpi, and 45x30mm (customer-visible label) in 300dpi
- Supported 300dpi printers: Zebra 105SL Plus or Zebra GX430T
- Module label includes P/N (part number) and FCCID, which must be filled out by the customer/manufacturer

If the print job settings require customization, the operator can do that in the correct `.json` file, found in the `targets\labels\` directory:

Print Components

```
description (select which components are to be printed here)  True or False
aId (association ID) TRUE (Mandatory)
dId (device ID) TRUE (Mandatory)
qr (QR code) TRUE (Mandatory)
fcc (FCC ID) (Optional)
pn (Part Number) (Optional)
blobs (Optional Graphical elements such as lines or logos)
```

Printer

```
description (Printer and Printing Configuration)
printer_dpi (600/300/200)
v_align_dots (Used to fine tune vertical placement. Value from -120 to 120)
h_align_dots (Used to fine tune vertical placement. Value from -120 to 120)
print_darkness (Used to set print darkness. Value from 0-30)
print_rate (Inches per minute. Value from 1-4)
copies (Number of copies to print)
```

label

```
description (Information on label loaded in the printer)
inches
      width_inches (Width of label in inches)
      height_inches (Height of label in inches)
```

qr

```
description (Code information)
magnification (Size of printed QR code. Value from 1-10)
x_coord (Coordinate of the left edge of the QR code in DOTS)
y_coord (Coordinate of the bottom edge of the QR code in DOTS)
```

aid

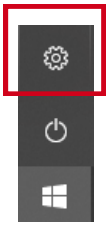
```
description (Association ID information)
font_type (Zebra built in fonts: A,B,D E,F,G,H,O,P,Q,R,S,T,U,V)
font_width (Printed width of the font in DOTS)
font_height (Printed height of the font in DOTS)
line_spacing (Spacing between the two lines of association ID)
char_chunk_size (Number of characters in each printed chunk)
chunks_per_line (Number of character chunks per line)
hyphen_font_type (Zebra built in fonts: A,B,D E,F,G,H,O,P,Q,R,S,T,U,V)
hyphen_font_width (Printed width of each font in DOTS)
hyphen_font_height (Printed height of each font in DOTS)
x_coord: (x coordinate of the left edge of the association ID in DOTS)
y_coord: (y coordinate of the bottom edge of the association ID in DOTS)
```

Adding a Path to Windows Environment Variables

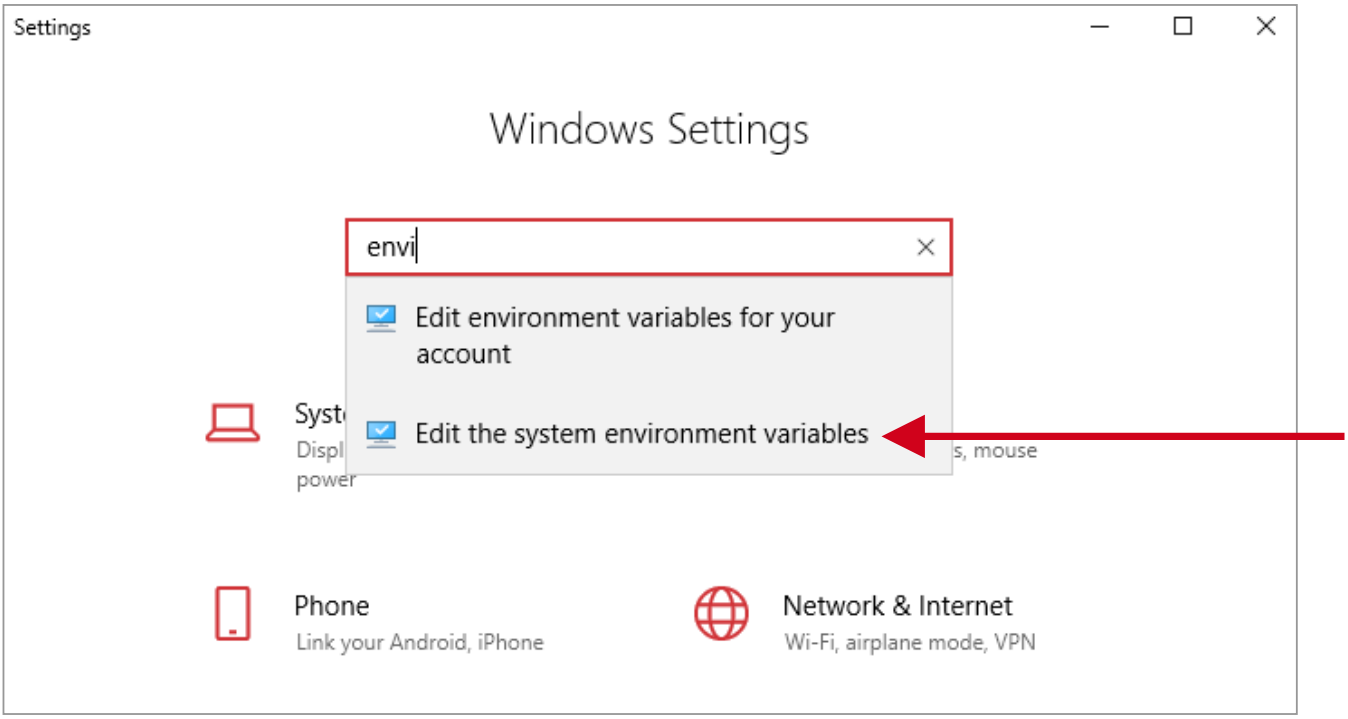
Before programming ASR-1 you must install Python 2.7 and the driver for the SEGGER J-Link. Be sure to install to the default locations on your PC.

Before the provisioning script can run successfully you will need to add to your environment variables the path to these executables. Note that these instructions are for Windows 10, but should be similar for other versions of Windows:

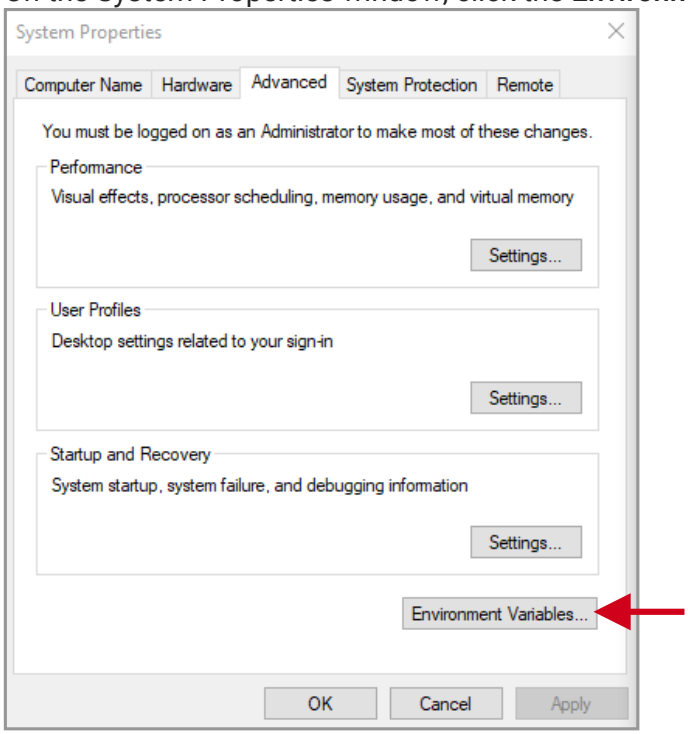
1. Click the Windows icon in the far-lower-right of your desktop, then click the Settings icon to open the Settings window:



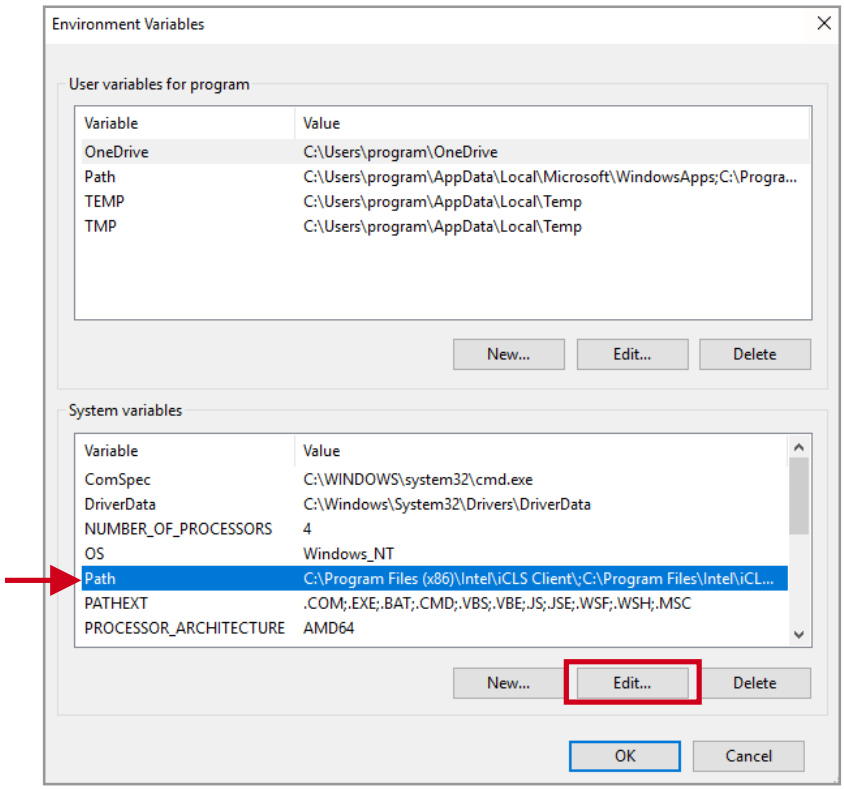
Windows Settings opens, where you should start typing “envi”, as shown. Select “Edit the system environment variables” as indicated above by the red arrow to open the System Properties window.



2. On the System Properties window, click the **Environmental Variables** button:

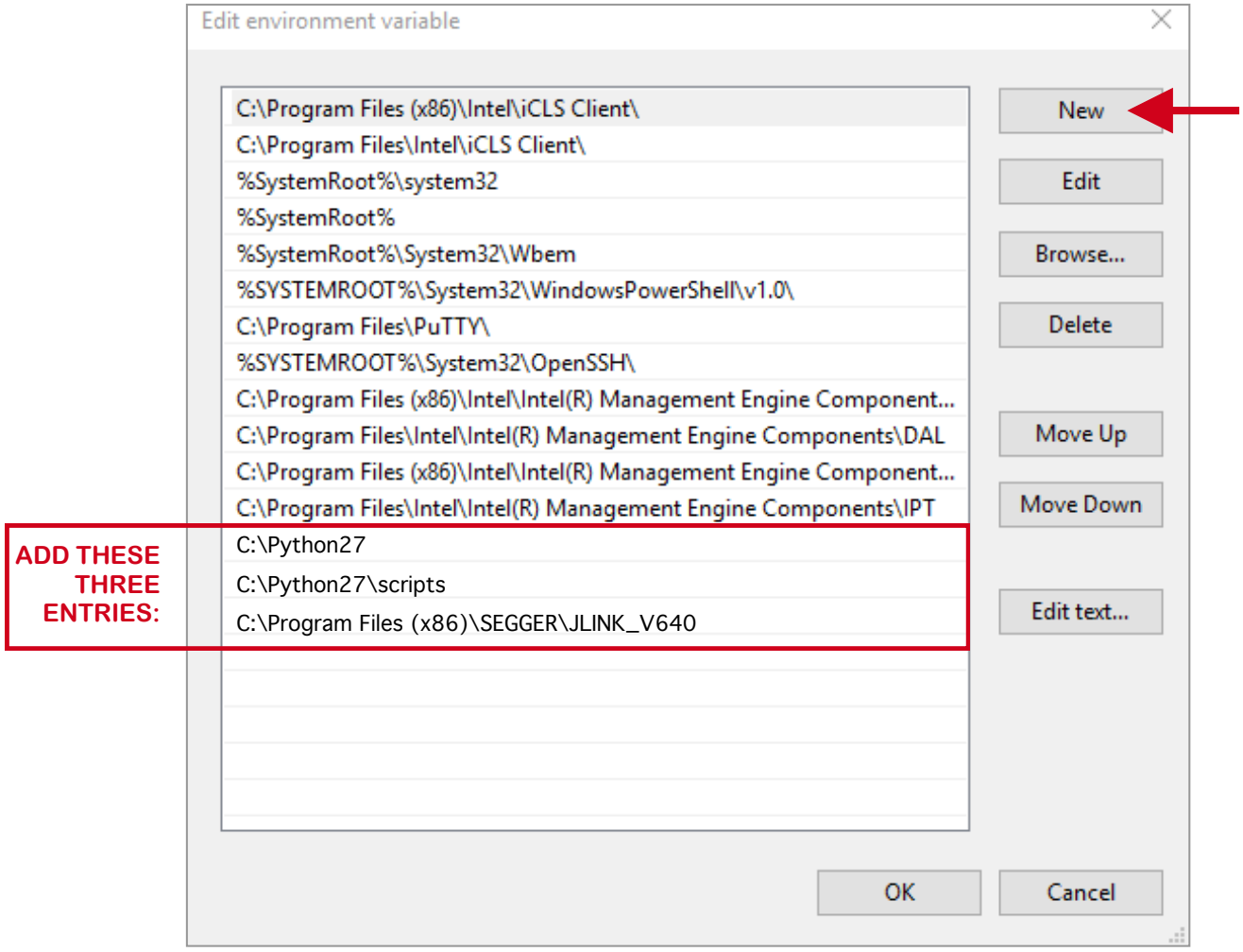


3. On the Environment Variables window, select **Path** to highlight it, then click the **Edit** button:



4. The **Edit environment variable** window, opens. Click the **New** button on the right, then **add** the three entries for Python and J-Link as shown below.

> **Important!** Earlier versions of Windows (<10) may require that you add paths by separating them from existing paths with a semicolon (;). Be care you **DO NOT remove any existing paths** when adding your new variable paths.



5. Click **OK** on each of the open windows to save the new variable paths.
6. You're all set!