Universal SDK Version 7.8.1

ReadMe File for rf IDEAS SOFTWARE and HARDWARE - rf IDEAS' pcProx® USB Software Developer's Kit, and Proximity Reader.

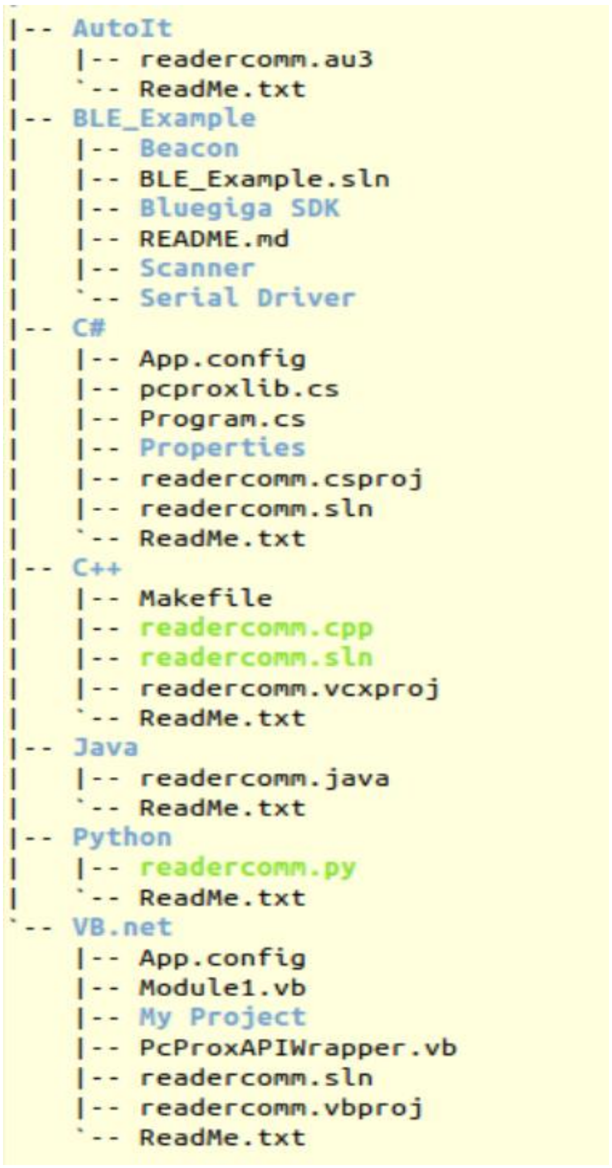
Version 6.x of the DLL Library supported a maximum of 16 devices, version 7 supports up to 127 USB devices and has been ported to Linux and Mac OS X.

We recommend new users use version 7 which is downward compatible with version 6.x. Version 7 will scan for all three products (pcProx, pcSwipe, pcProx Sonar) by default, version 6 will only connect with pcProx devices.

This installation file contains following examples:

1. C/C++
2. C#
3. Java
4. Python
5. VB.NET
6. AutoIt.
7. BLE_Example

Directory tree for examples



```
|-- AutoIt
|    |-- readercomm.au3
|    `-- ReadMe.txt
|-- BLE_Example
|    |-- Beacon
|    |-- BLE_Example.sln
|    |-- Bluegiga SDK
|    |-- README.md
|    |-- Scanner
|    `-- Serial Driver
|-- C#
|    |-- App.config
|    |-- pcproxlib.cs
|    |-- Program.cs
|    |-- Properties
|    |-- readercomm.csproj
|    |-- readercomm.sln
|    `-- ReadMe.txt
|-- C++
|    |-- Makefile
|    |-- readercomm.cpp
|    |-- readercomm.sln
|    |-- readercomm.vcxproj
|    `-- ReadMe.txt
|-- Java
|    |-- readercomm.java
|    `-- ReadMe.txt
|-- Python
|    |-- readercomm.py
|    `-- ReadMe.txt
`-- VB.net
     |-- App.config
     |-- Module1.vb
     |-- My Project
     |-- PcProxAPIWrapper.vb
     |-- readercomm.sln
     |-- readercomm.vbproj
     `-- ReadMe.txt
```

Note: This is the structure of examples folder for windows. For Linux and Mac OS X, the examples folder contain the examples for following language: C/C++, Java and Python only.

Note: GetActiveID() can be called every 250ms (4 times per second) to sample for cards in the reader. Typically the card data will be present for one second as this is the factory default setting.

Note: The 32 bit and 64 bit DLL's are the same name but can be distinguished from one another by the Property Detail as seen when right clicking on the DLL file.

The functions within this DLL are not "name managed" and are compatible with all languages capable of loading a DLL and calling its functions. This include VB .Net, C#, C++, Java, Python, AutoIt and more. The Linux shared library is compiled under Ubuntu 20.04 (32 & 64-bit) with g++ 9.4.0 on a 5.4.0 kernel for x86 platform.For ARM boards shared library compiled under Raspberry Pi 3B+ with g++ 9.4.0 on a 5.4.0 kernel. The Mac dynamic library is compiled under Monterey 12.6.2 using Xcode with Apple clang version 14.0.0.

Cavities :

1. **Why pcproxAPI SDK does not work on Linux kernel below then 2.6.39?**

Solution: HIDRAW driver which comes by default with Linux kernel does not support feature report communication before version 2.6.39 which rfIDEAS used to communicate with the readers.

2. **Why the following errors: Ioctl(SFEATURE) : Broken Pipe Error Ioctl(SFEATURE) : Protocol Error comes in Linux while developing console application?**

   Solution: These are the messages that are coming from 3rd party library HIDAPI on failure of device communication and there is no way to silent these messages. Do not treat them as errors.

3. **Why we receive an error message "error while loading shared libraries: libudev.so.0: cannot open shared object file: No such file or directory" on linux?**

   Solution: This error will come because libudev.so.0 is not supported in some newer kernel versions. They support libudev.so.1.

   To fix this error: link libudev.so.1 to libudev.so.0 using the command:

```
sudo ln -sf <location of libudev>/libudev.so.1  <location of libudev>/libudev.so.0
```

   Note: Creating a soft link is one of the solution that we tried and worked. However, user can choose any other method.

4. **Why we receive a message "Reader not Connected "even when the device is attached physically on linux?**

Solution: Whenever rfIDEAS readers are connected on Linux machine, User have to give appropriate permission to readers to communicate with them. However giving permission manually is not a hard task but when user have to connect and disconnect various readers frequently, this becomes a quite frustrating thing to do.

To handle such situations, there is another approach to give permission to readers, which is a onetime setup. After this process the user will never have to give permission to any rfIDEAS reader on that Linux machine.

This step requires creating a rules on Linux systems (works with any Linux based machines like Raspberry Pi machines too).

Steps:

- Open the terminal and type the following command:

  sudo vi /etc/udev/rules.d/rfideas.rules

- Type the following lines in the rfideas.rules file and save it.

```
KERNEL=="hidraw*", ATTRS{idVendor}=="<vendor_id>", MODE="0666"
KERNEL=="hidraw*", ATTRS{idVendor}=="<vendor_id>", ATTRS{idProduct}=="<product_id >",
MODE="0666"
SUBSYSTEM=="tty", ATTRS{idVendor}=="<vendor_id>", MODE="0666"
SUBSYSTEM=="tty", ATTRS{idVendor}=="<vendor_id>", ATTRS{idProduct}=="<product_id>",
MODE="0666"
```

  Note:

  Replace <vendor_id> and <product_id> with the actual value of vendor id and product id respectively. For standard pcProx reader, vendor_id = 0c27 and product_id = 3bfa.

- Type the following command to activate the newly created rules for rfIDEAS devices:

  sudo udevadm trigger

AFTER FOLLOWING STEPS, WHENEVER rfIDEAS READERS ARE CONNECTED, THEY WILL HAVE APPROPRIATE PERMISSIONS FOR COMMUNICATION.