Rosetta® IoT SDK

Hardware Getting Started Guide

Document number 513-325001-01

SPYRUS Product Design Information License Agreement

PLEASE READ THIS! This is a legal agreement between SPYRUS, Inc. ("SPYRUS") and the recipient of this document, whether an individual or an entity ("You"). BY ACCESSING, USING, REVIEWING OR READING THIS DOCUMENT OR PROVIDING FEEDBACK ("this document"), YOU AGREE TO BE BOUND BY THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS STOP READING AND DESTROY THIS DOCUMENT, AND DO NOT PROVIDE ANY FEEDBACK.

1. **This document is SPYRUS confidential information under Your most recent Non-Disclosure Agreement with SPYRUS.** However, Your only rights to use this document are as described in Paragraph 2 below. You are being granted a non-transferrable, defeasible license to review the material in this document only if You comply with the terms herein.
2. You may review the material in this document only (a) to provide feedback to SPYRUS; or (b) as a reference to assist You in planning and designing Your product, service or technology ("Your Product") to interface with a SPYRUS product, technology or service ("SPYRUS Product") as described in this document. All other rights are retained by SPYRUS; You have no other rights to use the intellectual property in this document. You may not (i) duplicate any part of this document, (ii) remove this Agreement or any notices from this document, or (iii) give any part of this document, or assign or otherwise provide Your rights under this Agreement, to anyone else.
    - **You have no obligation to give SPYRUS any suggestions, comments, or other feedback.** If You do give SPYRUS feedback on any version of this specification, You agree that SPYRUS may freely use, disclose, reproduce, license or otherwise distribute, and exploit Your feedback in its products, services, technologies, specifications and other documentation ("SPYRUS Offerings"), without any intellectual property restrictions, payments or other obligations.
3. This document contains preliminary information that may change prior to release of any associated SPYRUS Product, and is provided entirely "AS IS." To the extent permitted by law, SPYRUS MAKES NO WARRANTY OF ANY KIND EXPRESS OR IMPLIED, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND SHALL HAVE NO LIABILITY TO YOU FOR ANY DAMAGES, IN CONNECTION WITH THIS DOCUMENT OR ANY INTELLECTUAL PROPERTY IN IT.
4. If You are an entity and are acquired, or if more than a 20% of Your ownership changes, or if You are an individual and change employment, this Agreement terminates with no further notice and You must destroy this document.
5. This Agreement is governed by the laws of the State of California. Any dispute involving it must be brought in the federal or state courts located in Santa Clara County, California, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party's reasonable attorneys' fees, costs, and other expenses. If any part of this Agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall remain in effect. This Agreement is the entire agreement between You and SPYRUS concerning this document; it may be changed only by a written document signed by both You and SPYRUS.

Table of Contents

SPYRUS Product Design Information License Agreement ii
Table of Contents iii
Introduction 1
Setup on Windows 2
Setup Rosetta USB 2
Setup Rosetta microSD 3
Install SPYRUS CCID Devices in Linux 5
Open a root terminal, su root or run the following commands as a sudoer 5
Connect reader to system 5
Add Reader VID:PID and Description to CCID Configuration 6
Setup Rosetta microSD on Linux 9

Introduction

The SPYRUS Rosetta IoT SDK is designed to integrate the SPYRUS Rosetta USB and the SPYRUS Rosetta microSD into the Microsoft Azure IoT SDK.

The primary reason for this guide is to provide a series of steps for verifying that your Rosetta device is properly configured on the desired platform. If you are using the Rosetta USB, this involves verifying that you have a PC/SC driver on your host and that the driver is configured to recognize the Rosetta USB. If you are using the Rosetta microSD, this involves verifying that the device is recognized and mounted on the platform and that the communication with the SMARTIO file is established.

The Rosetta IoT SDK has been tested on Windows, Linux (Ubuntu 14.04 LTS & Ubuntu 16.04 LTS), and Raspberry Pi (Raspbian).

Supported Rosetta devices at this time are:-

- Rosetta Series III USB (SPYCOS® 3.0 FIPS & Non-FIPS)
- Rosetta Series III microSDHC (SPYCOS 3.0 FIPS & Non-FIPS)
- Linux2Go™ WorkSafe Pro™ USB 3.0

Setup on Windows

Setup Rosetta USB

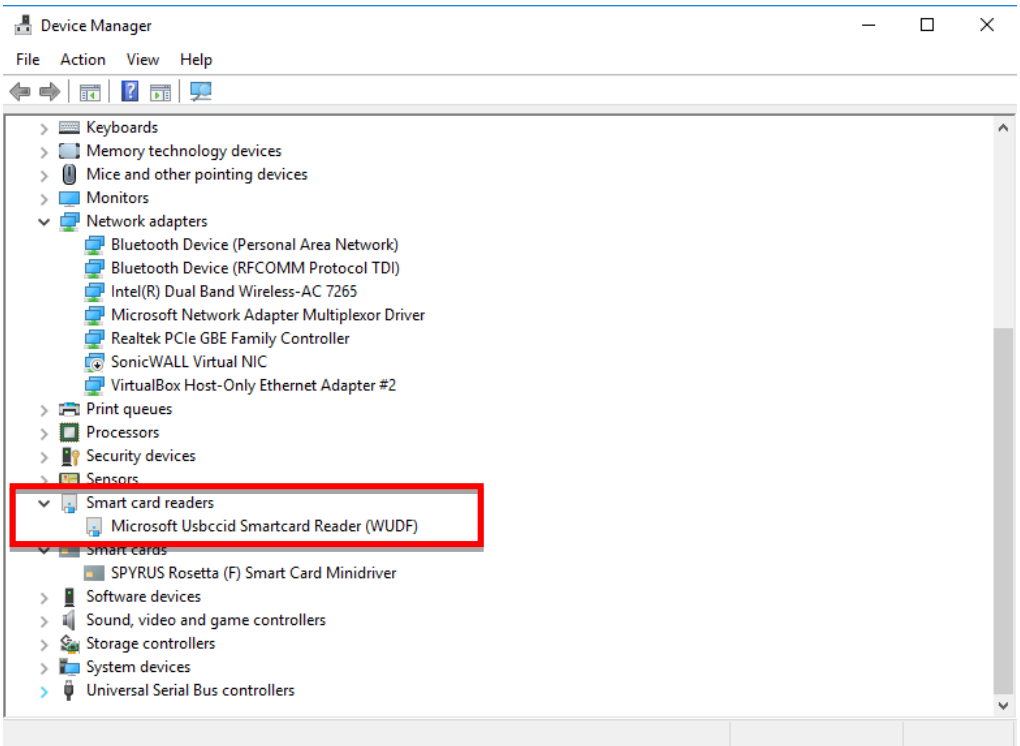
Windows has a USB CCID class driver that reduces the need for hardware vendors to create their own smart card reader specific driver. This driver is compatible with the Rosetta USB for all versions of Windows. If this driver is not installed out of the box, it is available from Microsoft through Windows Update as well as through the Windows Update Catalog.

To use the Rosetta USB with Windows, verify that the CCID driver recognizes the Rosetta USB. The can easily be accomplished using Windows Device Manager.

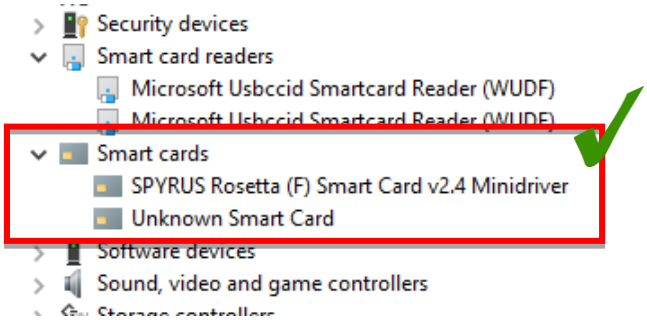
1. Log on to Windows
2. Open the Device Manager by navigating to 'Control Panel\Hardware and Sound\Device Manager' or searching for Device Manager with Cortana (Windows 10).
3. Insert the Rosetta USB.
4. It should be detected automatically, if not then click the button with the 'Scan for hardware changes' tooltip.

NOTE: You will require Administrator privileges if the Windows CCID device driver is not already installed on the system.

5. If you see an item under 'Smart card readers', your Rosetta USB is ready for use.



There is also an entry in the device manager for the Rosetta Smart Card. This entry is for the Rosetta Smart Card module embedded within the USB. If the SPYRUS Smart Card Minidriver is installed the device manager entry will look as shown below either without any errors or as an unknown smart card. It is okay if it shows up as unknown for the purposes of the IoT SDK since a minidrver is not required. This simply means that the SPYRUS Minidriver is not installed.



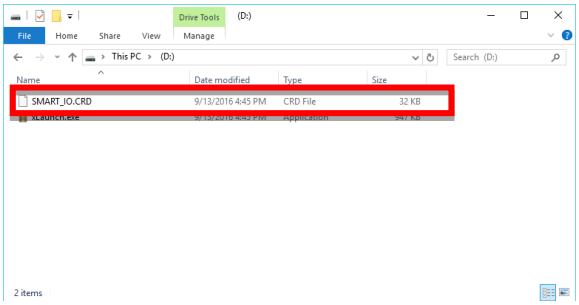
Beginning with Windows Vista, applications can use the Microsoft Cryptography API: Next Generation (CNG) for smart card–based cryptographic services. These SPYRUS minidriver enables access to the Rosetta USB through these Windows cryptographic services. The SPYRUS Minidriver is available both through Windows Update as well as directly from SPYRUS support.

NOTE: The SPYRUS Rosetta minidriver is not required by all applications to use Rosetta USB on Windows.

Setup Rosetta microSD

The Rosetta microSD is not a USB CCID device and does not use Windows smart card services. In fact Rosetta microSD does not require any special drivers at all. To function properly, Windows only needs to recognize and mount the file system. Verify that the device is recognized and mounted using Windows Explorer.

1. Log onto a Windows account.
2. Insert the Rosetta microSD.
3. The drive should automatically mount.
4. Open Windows Explorer and navigate to the mounted drive partition.
5. You may see a 'SMART_IO.CRD' file on the drive. If you do not see this file, it may be hidden. You can check for this file by enabling hidden items in the Windows Explorer settings. It is recommended that this setting be returned to not show hidden files for most uses after you have verified that the SMART_IO.CRD file exists.



The Rosetta SD can be tested using a tool provided with the sdk, smartio_test.exe. Smartio_test is used to display version information about your Rosetta microSD. It is a simple way to verify that the device is functioning and communications with the crypto-module within it is working. An example of the expected output is shown below. The most critical piece of information to verify is a successful reset. After a cold reset, Rosetta SD should respond with the hex sequence similar to what is highlighted below. If it is blank, communications with the cryptomodule are not operating properly.

```
=====================
  microSDHC Version
=====================
WR  :
RD  :  82 02 4A 01 68 44 42 01 57 54 02 00 2C 44 44 4B
       08
==============
  Cold Reset
==============
WR  :
RD  :  3B FB 18 00 00 40 78 80 59 53 50 59 52 55 53 0B
       00 03
==================
  SPYCOS Version
==================
WR  :  90 0C 00 00 06
RD  :  03 00 02 0C 00 0E 90 00
===============
  Select Root
===============
WR  :  00 A4 00 00 02 00 00
RD  :  61 17
```

Install SPYRUS CCID Devices in Linux

Linux has a similar smart card service to the one found on Windows. The PCSC lite project is an open source project that implements middleware to access a smart card using the SCard API (a.k.a. PC/SC). Part of this open source platform is a CCID device driver for Linux that is compatible with Rosetta USB. However, on some systems, configuration of the device driver may be necessary for it to recognize the Rosetta USB.

The first thing you will need to do on a Linux host is install open source CCID device driver. This setup is required for using the Rosetta USB with the Azure IoT SDK. The driver can be installed from your Linux distributions' repository or it can be built from the sources hosted on Github. The specific instructions for installing from your distributions repository may vary depending on our distribution. Here is one example that should work on Ubuntu.

Open a root terminal, su root or run the following commands as a sudoer

```
apt-get install libusb-dev
apt-get install libccid
apt-get install pcscd
apt-get install libpcsclite1
apt-get install pcsc-tools
```

Connect reader to system

Connect the reader to the system, insert the card if it is a removable card reader. Use the lsusb command to locate the reader information.

NOTE: the instructions that follow illustrate installing the SPYRUS PocketVault P-3X. The instructions are the same for the Rosetta USB, however, use the VID and PID found for the Rosetta USB not the P-3X in this example.

`lsusb -v -d 08df:` ← include the colon

```
# lsusb -v -d 08df:
Bus 001 Device 015: ID 08df:3201 Spyrus, Inc.
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.10
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x08df Spyrus, Inc.
  idProduct          0x3201
  bcdDevice            0.00
  iManufacturer           1 Spyrus Inc
  iProduct                2 PocketVault P-3X
  iSerial                 3 000002000000B2000EE4
```

If the reader is already in the CCID configuration file it will work now. Use the pcsc_scan utility to see if the reader is already setup.

```
pcsc_scan
```

If the reader and ATR show up then installation was successful; done, do not continue. If not then the reader needs to be added to the CCID configuration file; continue with the steps that follow.

Successful Installation

```
# pcsc_scan
PC/SC device scanner
V 1.4.10 (c) 2001-2007, Ludovic Rousseau <ludovic.rousseau@free.fr>
Compiled with PC/SC lite version: 1.3.3
Scanning present readers
0: PocketVault P-3X (000002000000B2000EE4) 00 00

Thu Sep 10 09:54:16 2015
 Reader 0: PocketVault P-3X (000002000000B2000EE4) 00 00
  Card state: Card inserted,
  ATR: 3B FB 18 00 00 40 78 80 59 53 50 59 52 55 53 0B 04 02

PCSC library does not contain all the required symbols at
/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Chipcard/PCSC.pm line
259.
Compilation failed in require at /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-
thread-multi/Chipcard/PCSC/Card.pm line 35.
Compilation failed in require at /usr/bin/ATR_analysis line 47.
BEGIN failed--compilation aborted at /usr/bin/ATR_analysis line 47.
ATR_analysis '3B FB 18 00 00 40 78 80 59 53 50 59 52 55 53 0B 04 02': Success
```

Not Successful (continue below – Add reader)

```
# pcsc_scan
PC/SC device scanner
V 1.4.10 (c) 2001-2007, Ludovic Rousseau <ludovic.rousseau@free.fr>
Compiled with PC/SC lite version: 1.3.3
Scanning present readers
SCardListReader: Cannot find a smart card reader. (0x8010002E)
```

Add Reader VID:PID and Description to CCID Configuration

Locate Info.plist, e.g. /usr/local/lib/pcsc/drivers/ifd-ccid.bundle/Contents/Info.plist

```
find / -name Info.plist
```

```
# find / -name Info.plist
/usr/lib64/pcsc/drivers/ifd-ccid.bundle/Contents/Info.plist
```

Edit Info.plist and add the VID, PID and reader description found using the lsusb command (Concatenate iVendor | iProduct for the reader description. E.g. Rosetta USB or Spyrus Inc. PocketVault P-3X). These

are three consecutive sections in the Info.plist XML file. Add the new reader to the beginning of each section.

```
# nano /usr/lib64/pcsc/drivers/ifd-ccid.bundle/Contents/Info.plist
```

Scroll down to the ifdVendorID key and add oxo8DF, SPYRUS's USB vendor id.

```
        Default value: 0
        -->

        <key>ifdManufacturerString</key>
        <string>Ludovic Rousseau (ludovic.rousseau@free.fr)</string>

        <key>ifdProductString</key>
        <string>Generic CCID driver</string>

        <key>ifdVendorID</key>
        <array>
                <string>0x08DF</string>
                <string>0x072F</string>
```

Scroll further down to the ifdProductID key and add the product id shown using the lsusb command for the reader being installed.

```
                <string>0x062D</string>
        </array>

        <key>ifdProductID</key>
        <array>
                <string>0x3201</string>
                <string>0x0004</string>
                <string>0x3115</string>
                <string>0x90CC</string>
                <string>0x0013</string>
```

Scroll further down to the ifdFriendlyName key and add a description for this reader. For this string concatenate the iVendor and iProduct strings found using the lsusb command.

```
                <string>0x2007</string>
                <string>0x0001</string>
        </array>

        <key>ifdFriendlyName</key>
        <array>
                <string>SPYRUS Inc. PocketVault P-3X</string>
                <string>ACS ACR 38U-CCID</string>
                <string>ActivIdentity USB Reader V3</string>
                <string>ActivIdentity Activkey_Sim</string>
```

Save and close the file. Restart the smart card service, pcscd.

```
/etc/init.d/pcscd restart
```

Re-scan to test the installation

```
pcsc_scan
```

```
# pcsc_scan
PC/SC device scanner
V 1.4.10 (c) 2001-2007, Ludovic Rousseau <ludovic.rousseau@free.fr>
Compiled with PC/SC lite version: 1.3.3
Scanning present readers
0: SPYRUS Inc. PocketVault P-3X (000002000000B2000EE4) 00 00

Thu Sep 10 10:16:00 2015
 Reader 0: SPYRUS Inc. PocketVault P-3X (000002000000B2000EE4) 00 00
  Card state: Card inserted,
  ATR: 3B FB 18 00 00 40 78 80 59 53 50 59 52 55 53 0B 04 02

PCSC library does not contain all the required symbols at
/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Chipcard/PCSC.pm line
259.
Compilation failed in require at /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-
thread-multi/Chipcard/PCSC/Card.pm line 35.
Compilation failed in require at /usr/bin/ATR_analysis line 47.
BEGIN failed--compilation aborted at /usr/bin/ATR_analysis line 47.
ATR_analysis '3B FB 18 00 00 40 78 80 59 53 50 59 52 55 53 0B 04 02': Success
```

Setup Rosetta microSD on Linux

To use the Rosetta microSD on a Linux based host, you will need to verify that the device is recognized and mounted on the platform and that the communication with the SMARTIO file is established.

1. Insert Rosetta microSD.
2. If drive does not auto mount, you may need to configure your system automount settings or you can manually mount it with 'sudo mount /dev/sd(x) /media/(username)/Spyrus' , where x is the SD's assigned drive letter.
3. Check the SD's mount point and for the SMART_IO.CRD file with 'ls –a /media/(username)/Spyrus'.

If the SMART_IO.CRD file is found, you are now ready to run the smartio_test. The smartio_test is included in the SPYRUS PKCS#11 SDK for Rosetta microSD. It is used to display version information about your Rosetta microSD. You may need to set the smart_io binary to executable with 'sudo chmod +x smart_io'. An example of the expected output is shown below.

```
mart_io/Debug$ ./smart_io /media/rr/SPYRUS
SmartIO Test
=========================
 microSDHC Version
=========================
WR  :
RD  :  82 02 4A 01 68 44 42 01 57 54 02 00 2C 44 44 4B
       08
=========================
 Cold Reset
=========================
WR  :
RD  :  3B FB 18 00 00 40 78 80 59 53 50 59 52 55 53 0B
       00 03
=========================
 SPYCOS Version
=========================
WR  :
RD  :  03 00 02 0C 00 0E 90 00
=========================
 Select Root
=========================
WR  :  90 0C 00 00 06
RD  :  61 17
rr@rr-desktop:~/NetBeansProjects/Desktop/Linux2/RosettaSD/PKCS11/linux_eclipse/s
mart_io/Debug$
```