DigitalPersona, Inc.

U.are.U SDK

Version 2

Platform Guide for Linux

DigitalPersona, Inc.
© 2012 DigitalPersona, Inc. All Rights Reserved.

All intellectual property rights in the DigitalPersona software, firmware, hardware and documentation included with or described in this guide are owned by DigitalPersona or its suppliers and are protected by United States copyright laws, other applicable copyright laws, and international treaty provisions. DigitalPersona and its suppliers retain all rights not expressly granted.

U.are.U® and DigitalPersona® are trademarks of DigitalPersona, Inc. registered in the United States and other countries. Java is a registered trademark of Oracle and/or its affiliates. All other trademarks are the property of their respective owners.

This DigitalPersona U.are.U SDK Platform Guide for Linux and the software it describes are furnished under license as set forth in the "License Agreement". Except as permitted by such license, no part of this document may be reproduced, stored, transmitted and translated, in any form and by any means, without the prior written consent of DigitalPersona. The contents of this manual are furnished for informational use only and are subject to change without notice.

Any mention of third-party companies and products is for demonstration purposes only and constitutes neither an endorsement nor a recommendation. DigitalPersona assumes no responsibility with regard to the performance or use of these third-party products.

DigitalPersona makes every effort to ensure the accuracy of its documentation and assumes no responsibility or liability for any errors or inaccuracies that may appear in it.

Technical Support

To access DigitalPersona technical support resources, go to http://www.digitalpersona.com/support.

Phone support is available at (877) 378-2740 (US) or +1 650-474-4000 (outside the US).

Feedback

Although the information in this guide has been thoroughly reviewed and tested, we welcome your feedback on any errors, omissions, or suggestions for future improvements. Please contact us at

TechPubs@digitalpersona.com

or

DigitalPersona, Inc.
720 Bay Road, Suite 100
Redwood City, California 94063
USA
(650) 474-4000
(650) 298-8313 Fax

Document Publication Date: November 28, 2012 release 2.2

Table of Contents

1 Introduction 5
Getting Updated Documentation 5

2 Installation 6
Installing on the Development System 6
Installing on the Target (4500 and 4000B Readers) 8
Installing on Target Hardware (5100 Reader) 10
Uninstalling 11

3 Developing Applications with C/C++ 12
Pre-Requisites 12
System Requirements 12
Kernel 12
Development System 12
Target Runtime Hardware (Linux-Based Device) 12
The Sample Applications 13
Building the Sample Applications 13
Running the UareUSample Application 14
Running the UareUCaptureOnly Application 20

4 Developing Applications with Java 25
Pre-Requisites 25
System Requirements 25
Development System 25
Target Runtime Hardware (Linux Reader) 25
Installing on the Target Hardware 25
The Java Sample Application 26
Selecting a Reader 27
Enrolling a Finger 29
Identifying a Fingerprint 30
Verifying a Fingerprint 31
Using the Capture and Streaming Feature 32

5 Developing Applications with JavaPOS 33
Pre-Requisites 33
System Requirements 33
Linux Kernel 33
Development System 33
Target Runtime Hardware (Linux Reader) 33

Installation Notes 34
Installing on the Target Hardware 34
Enabling and Disabling the USB Hotplug Device 34
Registering the DigitalPersona U.are.U Device Service 34
Upgrading from Previous Versions of the JavaPOS API 35
The JavaPOS Sample Application 36

6 Redistribution 45
Merge Modules 45
Fingerprint Reader Documentation 45
Hardware Warnings and Regulatory Information 45
Fingerprint Reader Use and Maintenance Guide 46

Introduction 1

This manual describes how to use the U.are.U SDK to develop applications that will run on Linux. The U.are.U SDK is available for multiple platforms and this document describes issues specific to developing applications that will run on Linux-based platforms:

Chapter 1, *Introduction* (this chapter) describes how to get the latest version of this documentation.

Chapter 2, *Installation* provides instructions for installing on your development system and on the target (Linux) reader.

Chapter 3, *Developing Applications with C/C++* lists system requirements for developing and running applications in C/C++ and describes the sample application.

Chapter 4, *Developing Applications with Java* lists system requirements for developing and running applications using Java, provides additional installation instructions and describes the Java sample application.

Chapter 5, *Developing Applications with JavaPOS* provides information on using the JavaPOS-compliant API built on the U.are.U framework.

Chapter 6, *Redistribution* describes the merge modules that are provided to help you redistribute applications built using the U.are.U SDK.

For a detailed description of the SDK, consult the U.are.U SDK Developer Guide.

Getting Updated Documentation

If you are viewing this guide from the download package for the U.are.U SDK, you may want to check online at our website for an updated version of this document at

http://www.digitalpersona.com/Support/Reference-Material/DigitalPersona-SDK-Reference-Material/

Installation 2

Except as noted in the platform/language-specific chapters, the installation process is the same for all Linux development.

There are two steps to the installation:

1. Installing on the development system
2. Installing on the device (the target hardware)

These steps are described below.

Installing on the Development System

To install the SDK on your development system:

1. Unpack **`DP-UareU-2.2.0-`**<*`date,build`*>**`.tar.bz2`** into a directory.
2. Go to the directory where you unpacked the files.
3. Run `install`. You must be **root** to install the SDK.

   If the installation scripts are not executable, you might first need to run `chmod +x install`

4. Follow the prompts as they appear. Note that if you are using the U.are.U 4000 or 4500 reader, you must select the option to build and install the driver kernel object for your development system. This step is not required if you are using a 5100 reader or need only the FingerJet Engine for fingerprint recognition.

The installer copies all necessary files to the selected folder (by default, the product folder is `/opt/DigitalPersona/UareUSDK`). The files installed on the developer's machine are located in the following folders within the main product folder:

Folder	Contents
`Include`	Header files for C/C++ API.
`Linux/docs`	End user license agreement (EULA) plus documentation: ■ SDK Developer Guide - describes all APIs ■ Platform Guide for Linux - Linux-specific details ■ C_API - Doxygen for C/C++ API ■ Java_API - Javadoc for Java API
`Linux/drivers`	Drivers for DigitalPersona fingerprint readers
`Linux/lib`	Runtime files for: ■ `arm` - C/C++ development for ARM processors ■ `x64` - C/C++ development for x64 processors ■ `x86` - C/C++ development for x86 processors ■ `java` - Java and JavaPOS JAR files
`Linux/Samples/bin`	**Compiled sample applications:** ■ `Bin` ■ `java` - Java sample ■ `javapos` - JavaPOS sample ■ `linux-arm` - C/C++ sample for ARM processors ■ `linux-x64` - C/C++ sample for x64 processors ■ `linux-x86` - C/C++ sample for x86 processors **Source files for sample applications:** ■ `UareUSample` - C/C++ sample ■ `UareUCaptureOnly` - C/C++ sample that demonstrates only capture ■ `UareUSampleJava` - Java sample ■ `UareUSampleJavaPOS` - JavaPOS sample

Installing on the Target (4500 and 4000B Readers)

To install on the target platform you must

1. Copy the appropriate tarballs to the target device and unpack them.
2. Build and install the kernel object.

1. Copying Files

Depending on the functionality you need, you must copy and unpack the tarballs as shown in the table below. The files will typically be copied to `/usr/lib` or `/usr/local/lib`. To run applications, `LD_LIBRARY_PATH` should point to the correct location or the dynamic linker cache should be remade to include the location with the library files.

Platform	Functionality	Required file
ARM	Capture	**DigitalPersona-fpdriver4k-2.1.0.7-1.arm.tar.gz** **DigitalPersona-fpcapture-2.1.0-1.arm.tar.gz**
	Feature extraction (FingerJet Engine)	**DigitalPersona-fpengine-2.1.0-1.arm.tar.gz**
x64	Capture	**DigitalPersona-fpdriver4k-2.1.0.7-1.x64.tar.gz** **DigitalPersona-fpcapture-2.1.0-1.x64.tar.gz**
	Feature extraction (FingerJet Engine)	**DigitalPersona-fpengine-2.1.0-1.x64.tar.gz**
x86	Capture	**DigitalPersona-fpdriver4k-2.1.0.7-1.x86.tar.gz** **DigitalPersona-fpcapture-2.1.0-1.x86.tar.gz**
	Feature extraction (FingerJet Engine)	**DigitalPersona-fpengine-2.1.0-1.x86.tar.gz**

The folder

/opt/DigitalPersona/UareUSDK/redist

contains the redistributable files for all supported target architectures, including

DigitalPersona-fpkrndrv4k-source-2.1.0.7-1.noarch.tar.gz

which contains the source code for the kernel object for the 4000B and 4500 readers.

2. Building and Installing the Driver Kernel Module

The U.are.U SDK includes a kernel module source package that contains the driver for the U.are.U 4000/4500 fingerprint reader. The kernel is required on the development machine (for testing) and on the target machine at runtime.

You would normally build the kernel on your development machine during installation. However if you did not build the kernel during installation OR your target platform is different than your development platform, you can build the kernel manually by following the instructions below.

To build the kernel module, the following packages/components must be installed on your system:

- kernel sources or kernel headers for the target platform
- `binutils`
- `gcc`, preferably the same version which was used to build the kernel for the target platform
  OR
  if the target platform is a different architecture than the development platform, the toolchain for the cross-compiling binaries for the target platform.

To manually build the kernel module for a target platform that is the same as your development platform:

1. Change to the `/opt/DigitalPersona/UareUSDK/drivers/source/usbdpfp` directory.
2. Run `make`

To manually build the kernel module for a target platform that is the same as your development platform BUT with a different kernel:

1. Change to the `/opt/DigitalPersona/UareUSDK/drivers/source/usbdpfp` directory.
2. Run `KDIR=<directory with the kernel sources for the target platform> make`

To manually build the kernel module for an ARM target platform on an x86 or x64 development platform:

1. Change to the `/opt/DigitalPersona/UareUSDK/drivers/source/usbdpfp` directory.
2. Run `ARCH=arm CROSS_COMPILE=<cross-toolchain prefix> KDIR=<directory with the kernel sources for the target platform> make`

To install the kernel module

1. Copy the `/opt/DigitalPersona/UareUSDK/drivers/source/usbdpfp/mod_usbdpfp.ko` file to the kernel module directory `/lib/modules/‘uname -r‘/kernel/drivers/biometric`. You might need to create the target directory first.
2. If your system uses `udev`, copy the `40-usbdpfp.rules` file to its proper location in the `/etc/udev/rules.d` directory.

   ```
   cp /opt/DigitalPersona/UareUSDK/redist/40-usbdpfp.rules /etc/udev/rules.d
   ```

   If your system does not use `udev`, you may need to manually change the group and user permissions for the files /dev/usbdpfp0 and `/dev/usbdpfpPnp` to not require user applications to be run with root privileges. You may need to make the following change:

```
sudo chmod 666 /dev/usbdpfp*
```

3. Run `/sbin/depmod`.
4. Load the driver by issuing the following command from the biometric directory.

```
su -c "insmod ./mod_usbdpfp.ko"
```

To verify that the kernel module is loaded:

- Run `lsmod | grep mod_usbdpfp`

Installing on Target Hardware (5100 Reader)

Depending on the functionality you need, you must copy and unpack the tarballs as shown in the table below. The files will typically be copied to `/usr/lib` or `/usr/local/lib`. To run applications, `LD_LIBRARY_PATH` should point to the correct location or the dynamic linker cache should be remade to include the location with the library files.

Platform	Functionality	Required file
ARM	Capture	**DigitalPersona-fpdriver5000-2.0.0.7-1.arm.tar.gz** **DigitalPersona-fpcapture-2.0.0-1.arm.tar.gz**
	Feature extraction (FingerJet Engine)	**DigitalPersona-fpengine-2.0.0-1.arm.tar.gz**
x64	Capture	**DigitalPersona-fpdriver5000-2.0.0.7-1.x64.tar.gz** **DigitalPersona-fpcapture-2.0.0-1.x64.tar.gz**
	Feature extraction (FingerJet Engine)	**DigitalPersona-fpengine-2.0.0-1.x64.tar.gz**
x86	Capture	**DigitalPersona-fpdriver5000-2.0.0.7-1.x86.tar.gz** **DigitalPersona-fpcapture-2.0.0-1.x86.tar.gz**
	Feature extraction (FingerJet Engine)	**DigitalPersona-fpengine-2.0.0-1.x86.tar.gz**

The folder

/opt/DigitalPersona/UareUSDK/**redist**

contains the redistributable files for all supported target architectures, including

DigitalPersona-fpkrndrv5000-source-2.0.0.7-1.noarch.tar.gz

which contains the source code for the kernel object for the 4000B and 4500 readers.

Uninstalling

If you need to uninstall the SDK, run

```
./uninstall
```

Note that you must be **root** to uninstall.

Developing Applications with C/C++ 3

Pre-Requisites

This chapter assumes that you have a working knowledge of C/C++ and that you know how to develop for applications that will run on Linux platforms and devices.

System Requirements

Kernel

The Linux kernel that is supported for target and development systems is:

- **5100 reader:** tested on kernel version 2.6.32 to 3.2.25.
- **4500 reader:** will compile correctly only for kernel version 2.6.18 to 2.6.41.

The FingerJet Engine is supported on all versions of the Linux kernel.

Development System

The development system can be any Intel-based Linux system, as long as it has a kernel supported by the reader (as specified above).

DigitalPersona supports and has fully tested the following development platforms (32-bit and 64-bit):

- Ubuntu 10.04 LTS, 11.04, 12.04
- Fedora 14-17
- OpenSUSE 11.3, 11.4, 12.1

These utilities, packages, and libraries should already be installed on your system:

- `usbutils`
- `binutils` (`lsmod` and `lsusb` utilities)

Target Runtime Hardware (Linux-Based Device)

The Linux-based device that will run the application must be one of the following hardware platforms:

- Linux kernel supported by the reader (as specified above)
- Intel x86 architecture with CPU from 600MHz and at least 16Mb of available RAM;
- Intel x64 (x86-64) architecture with CPU from 600MHz and at least 16Mb of available RAM;
- ARMv4T EABI with CPU from 400MHz and at least 16Mb of available RAM.

The file sizes are (in Kb):

	x86	x64 (x86-64)	ARMv4T
Capture runtime (drivers + SDK layer)	520	650	530
Fingerprint recognition runtime	540	585	260

In addition, the device must also have:

- a USB port
- 16 MB free memory

The SDK works on a variety of hardware and is intended to have a small footprint so that it can run even on minimal hardware. Less capable hardware will work, but response time may not be optimal.

The Sample Applications

U.are.U SDK includes sample applications to demonstrate the features of the SDK. The sample applications are located in subfolders of the `/opt/DigitalPersona/UareUSDK/Linux/Samples` folder:

- The `Bin` subdirectory contains pre-compiled sample applications for ARM, x86 and x64 architectures.
- The `UareUCaptureOnly` subdirectory contains the sources of the sample application that demonstrates fingerprint capture. This application does not have any dependencies on the FingerJet engine libraries.
- The `UareUSample` subdirectory contains the source for the sample application that demonstrates the full functionality of the SDK: fingerprint capture, feature extraction and verification, identification and enrollment of fingerprints. This application has dependencies on the capture and FingerJet Engine libraries.

Building the Sample Applications

`UareUSample` and `UareUCaptureOnly` are built in the same way. The instructions for `UareUCaptureOnly` are provided below.

To manually build the `UareUCaptureOnly` sample application for the same target architecture as your development system:

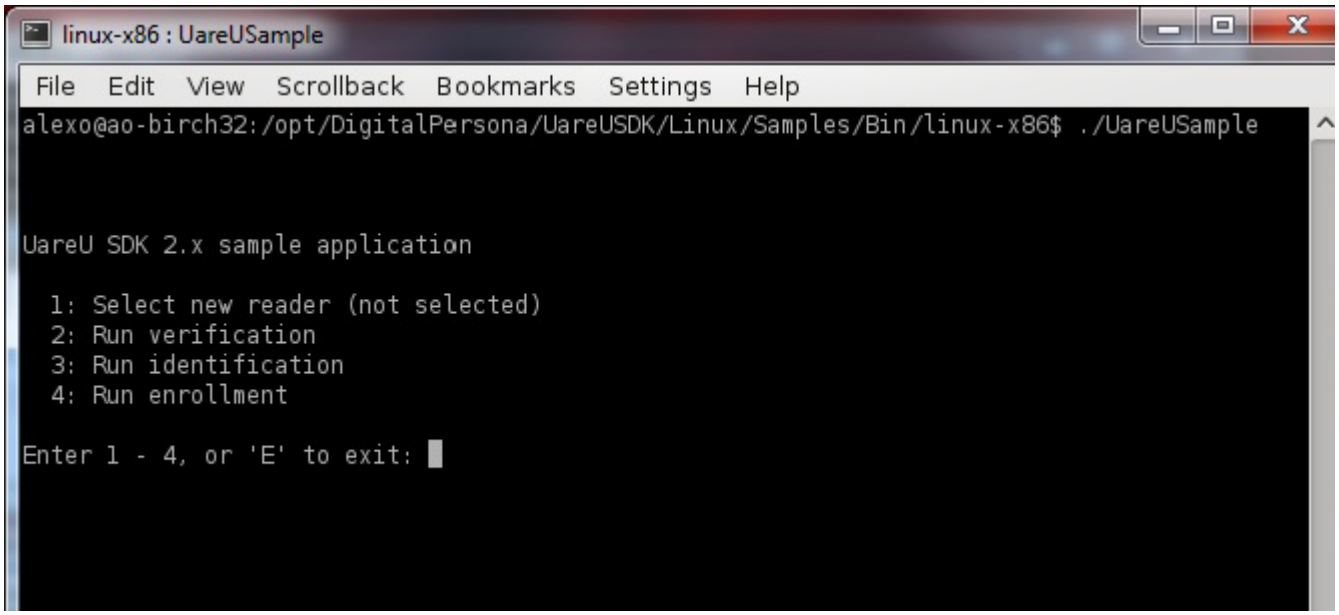
1. Change to the **/opt/DigitalPersona/UareUSDK/Linux/Samples/UareUCaptureOnly** directory.
2. Run **make**

To manually build the `UareUCaptureOnly` sample application for ARM architecture (on a x86 or x64 development system):

1. Change to the **/opt/DigitalPersona/UareUSDK/Linux/Samples/UareUCaptureOnly** directory.
2. Run **CC=<full name of C cross-compiler> CFLAGS="L/opt/DigitalPersona/lib_arm" make**
   (Depending on the cross-toolchain, additional parameters may be required in the CFLAGS parameter, for example, for GCC: **CFLAGS="-march=armv4t -L/opt/DigitalPersona/lib_arm"**

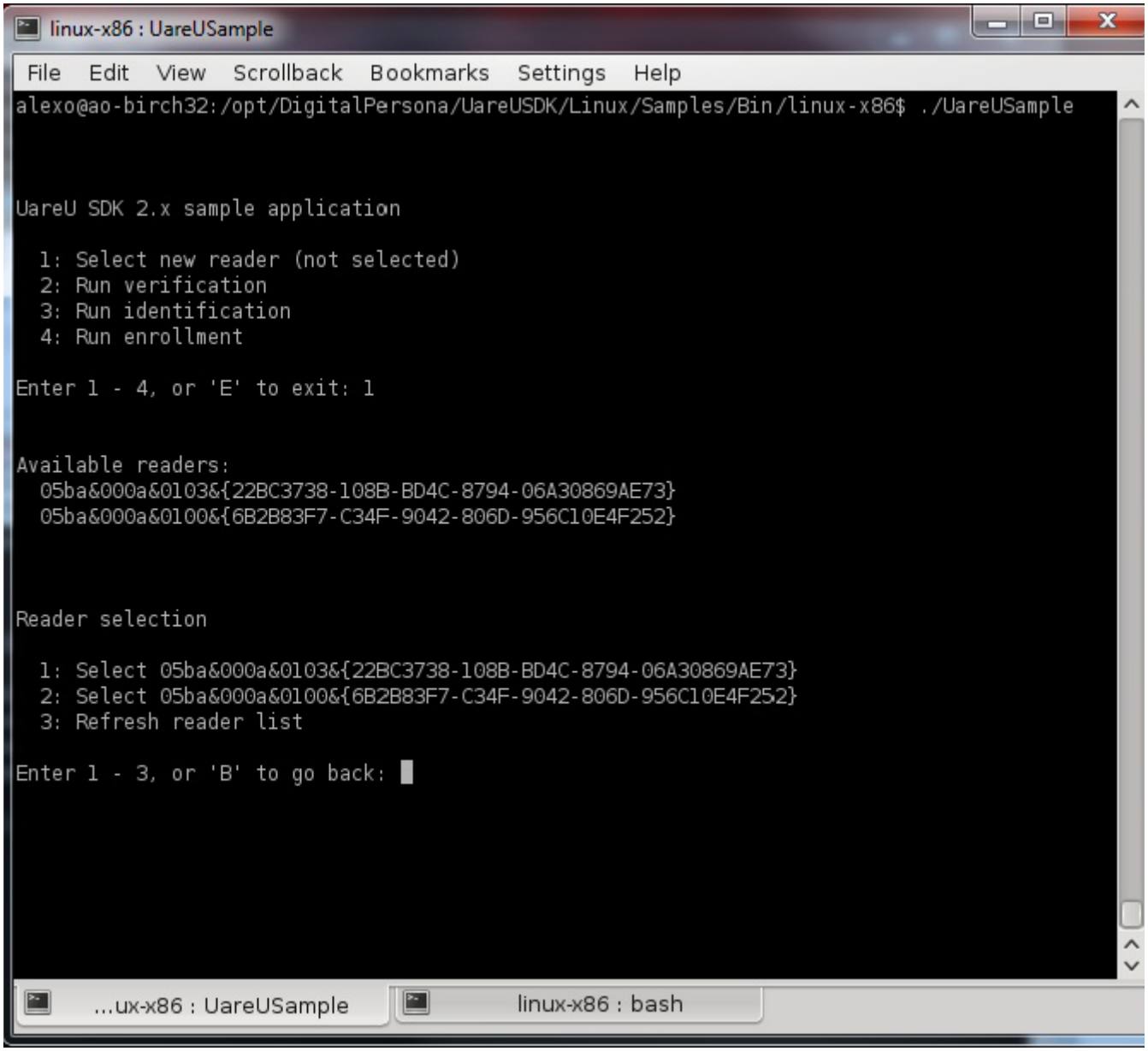
Running the UareUSample Application

The `UareUSample` application demonstrates the Capture API and the FingerJet Engine API. The U.are.U SDK sample application is a console applications -- it does not require any graphic libraries or desktop environment. Be sure that LD_LIBRARY_PATH points to the location of the DigitalPersona libraries. When you launch the application you will see the main menu, as shown below:

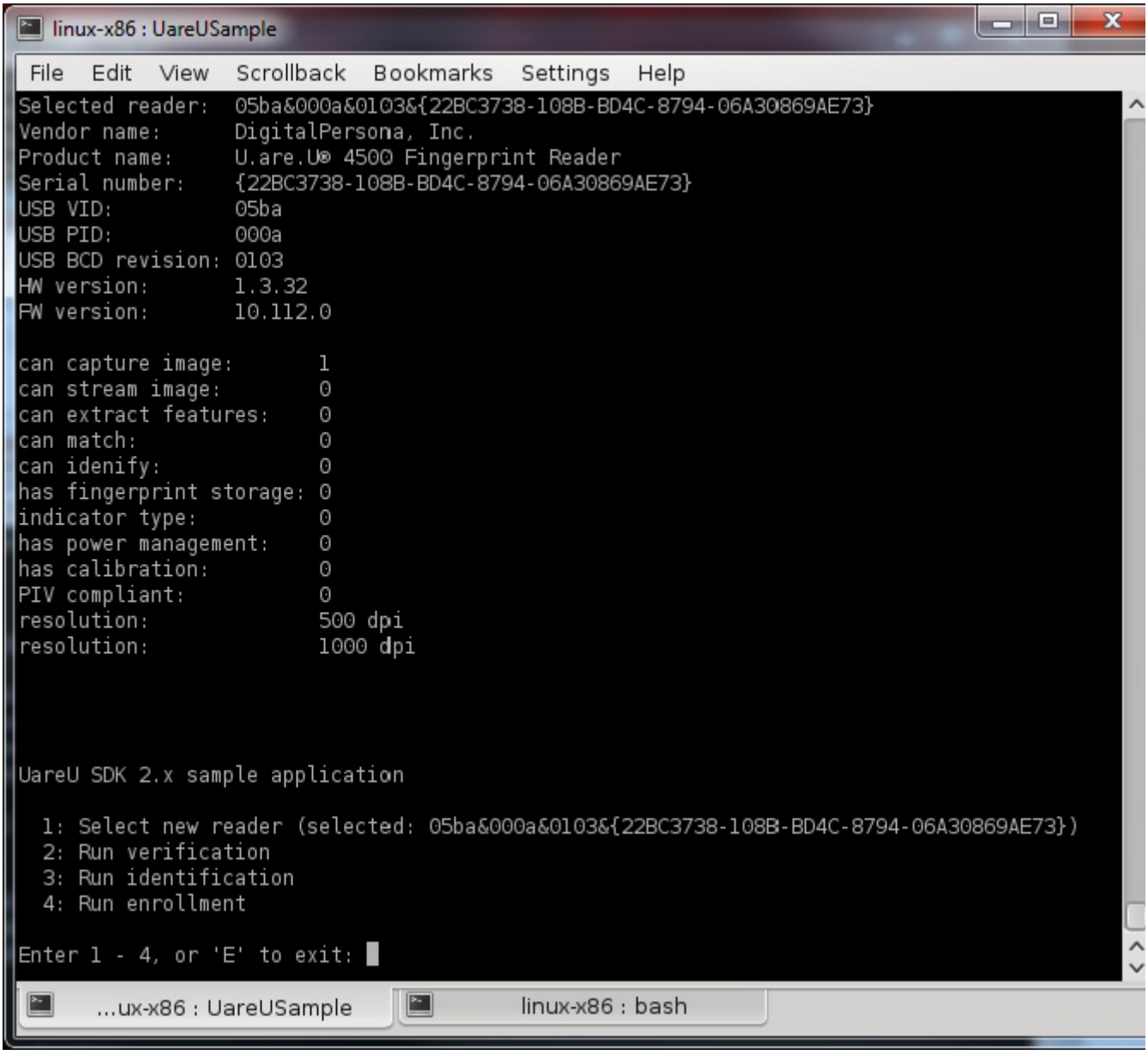


You must begin by selecting a fingerprint reader device to use with the application. After you press **1** and Enter to select the reader, the application will search for any fingerprint readers and display a list of the readers it

finds, as shown below. Choose the desired reader and press Enter to continue.

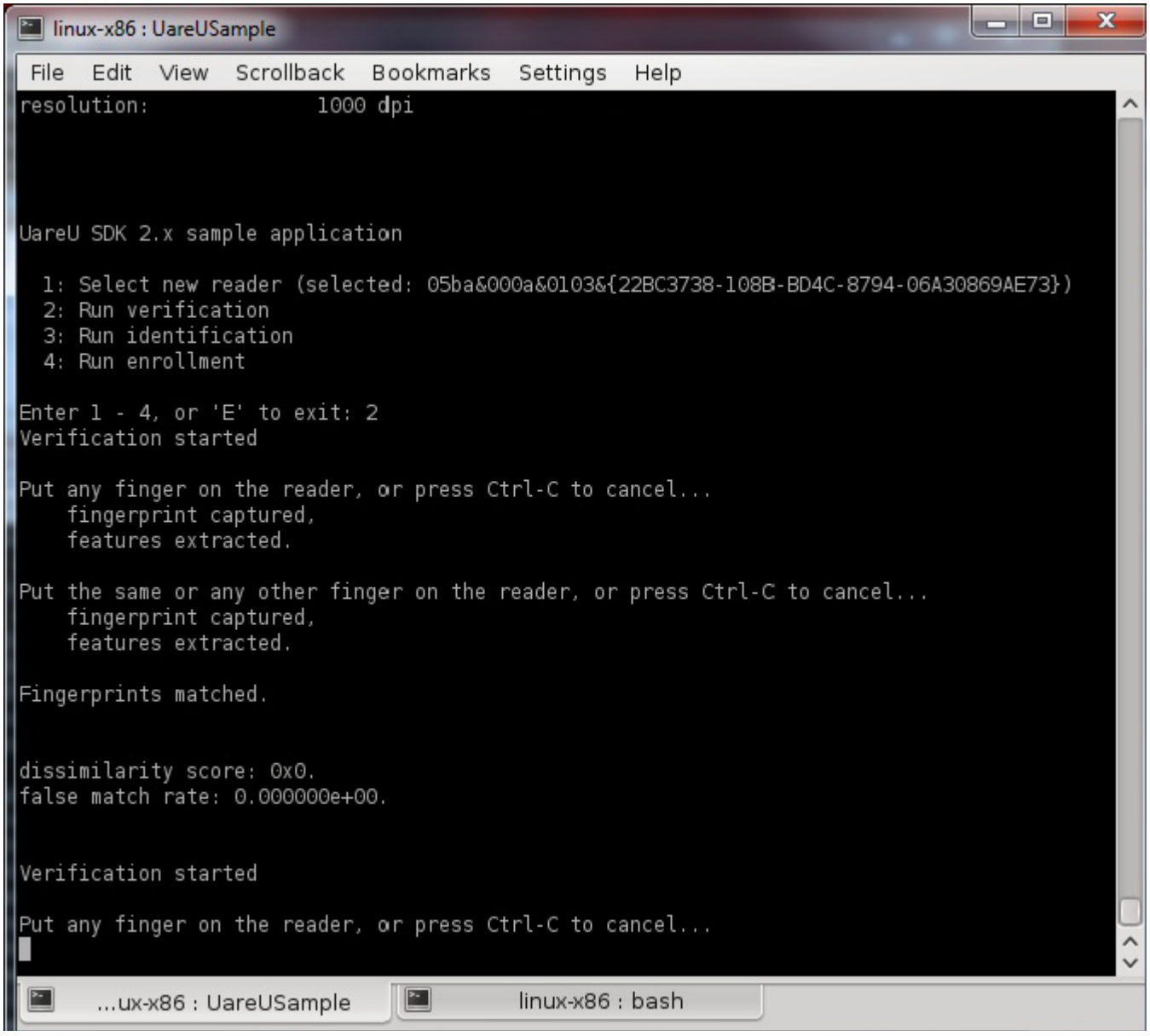


Once a reader is selected, you will see a screen that shows information about the reader and its capabilities, as shown below.



Now let's test verification. To do that, press **2** and press Enter.

Follow the prompts on the screen as shown below and touch your finger on the reader. You will see that the program extracts the fingerprint features. Now touch the reader again and the sample application will extract the features of the second fingerprint and determine whether the fingerprints match or not.

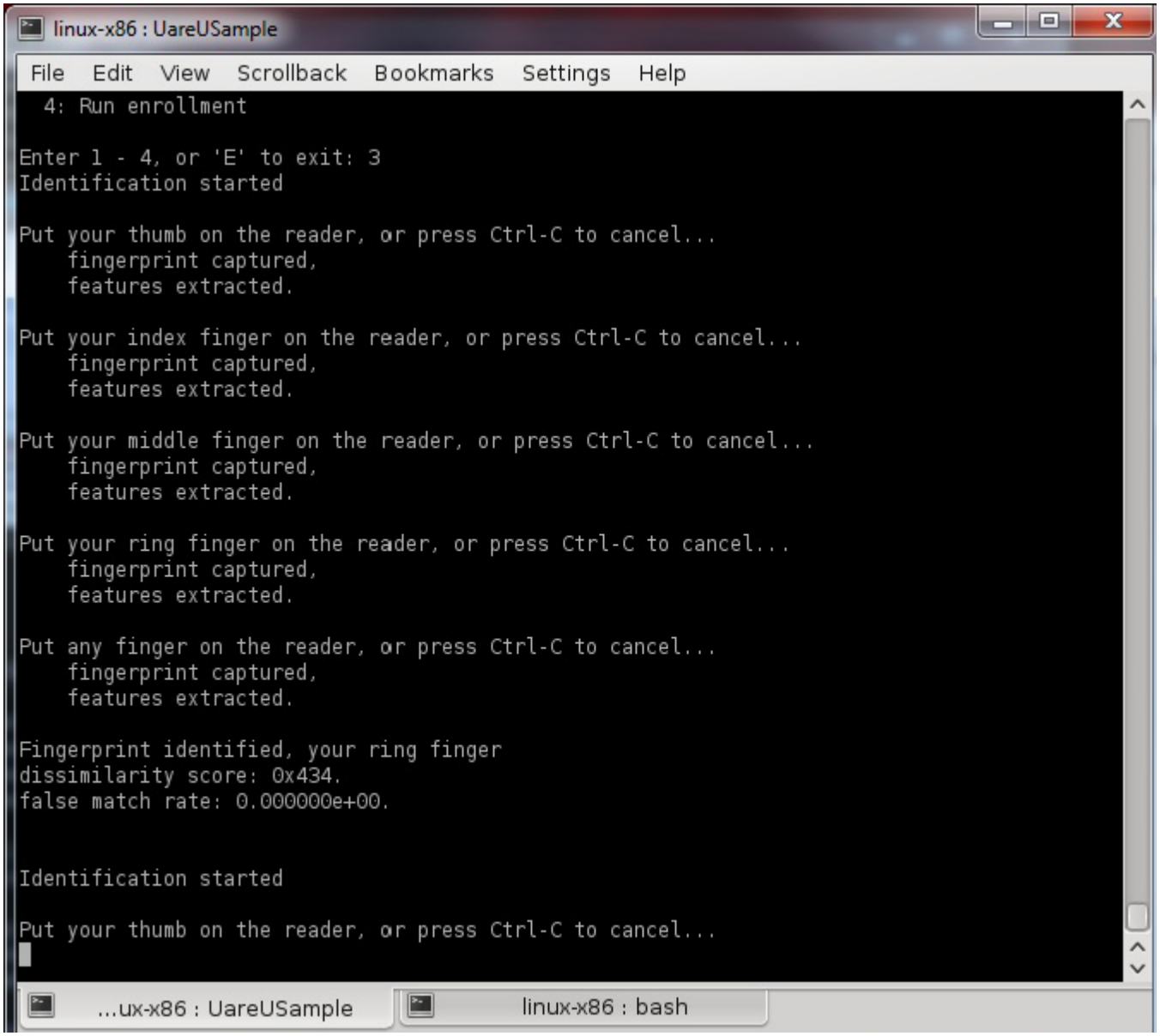


Try touching your index finger as the first fingerprint and your middle finger as the second fingerprint and note that they do not match.

Press the Esc key to return to the main menu.

Now let's test identification. Press **3** and Enter to start.

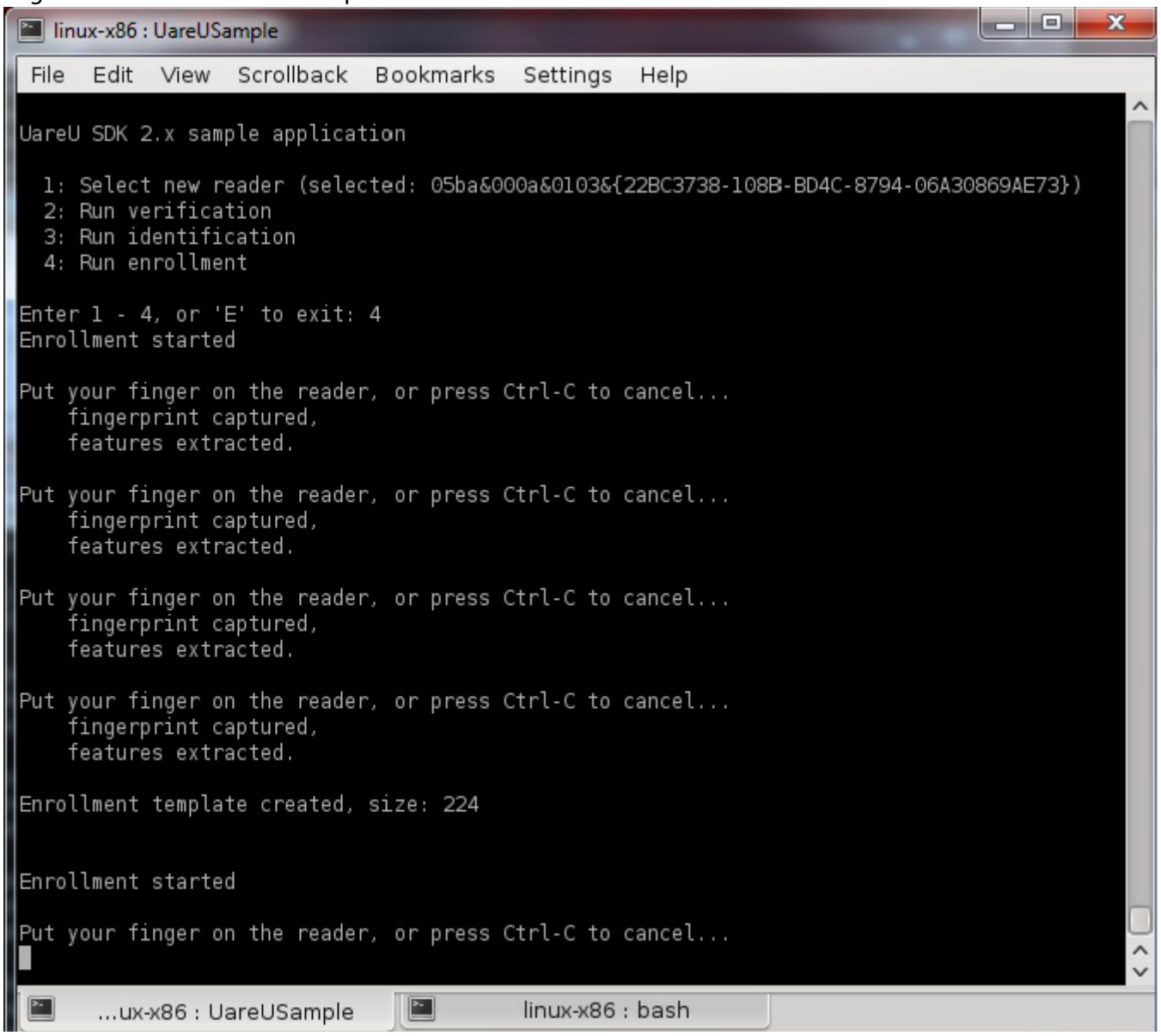
Remember that identification matches a single fingerprint against a set of fingerprints. You will be prompted to touch each of your fingers on the reader in turn, as shown below. When you have entered all of your fingerprints, you will be prompted to touch any finger on the reader. Try touching one of your fingers to ensure that the fingerprint matches. Try touching the reader with a finger from your other hand and ensure that it does NOT match.



Press Ctrl C to return to the main menu.

Let's test enrollment now. Press **4** and Enter to start.

The enrollment test is to enroll a single finger. Follow the prompts on the screen (as shown below) and touch your finger on the reader when prompted. Then when prompted, touch the same finger on the reader again. It typically takes four scans to get a good enrollment. If you touch some other finger on the reader during enrollment, or if a scan is of poor quality, that scan will be dropped and the application will request additional finger scans in order to do a complete enrollment.



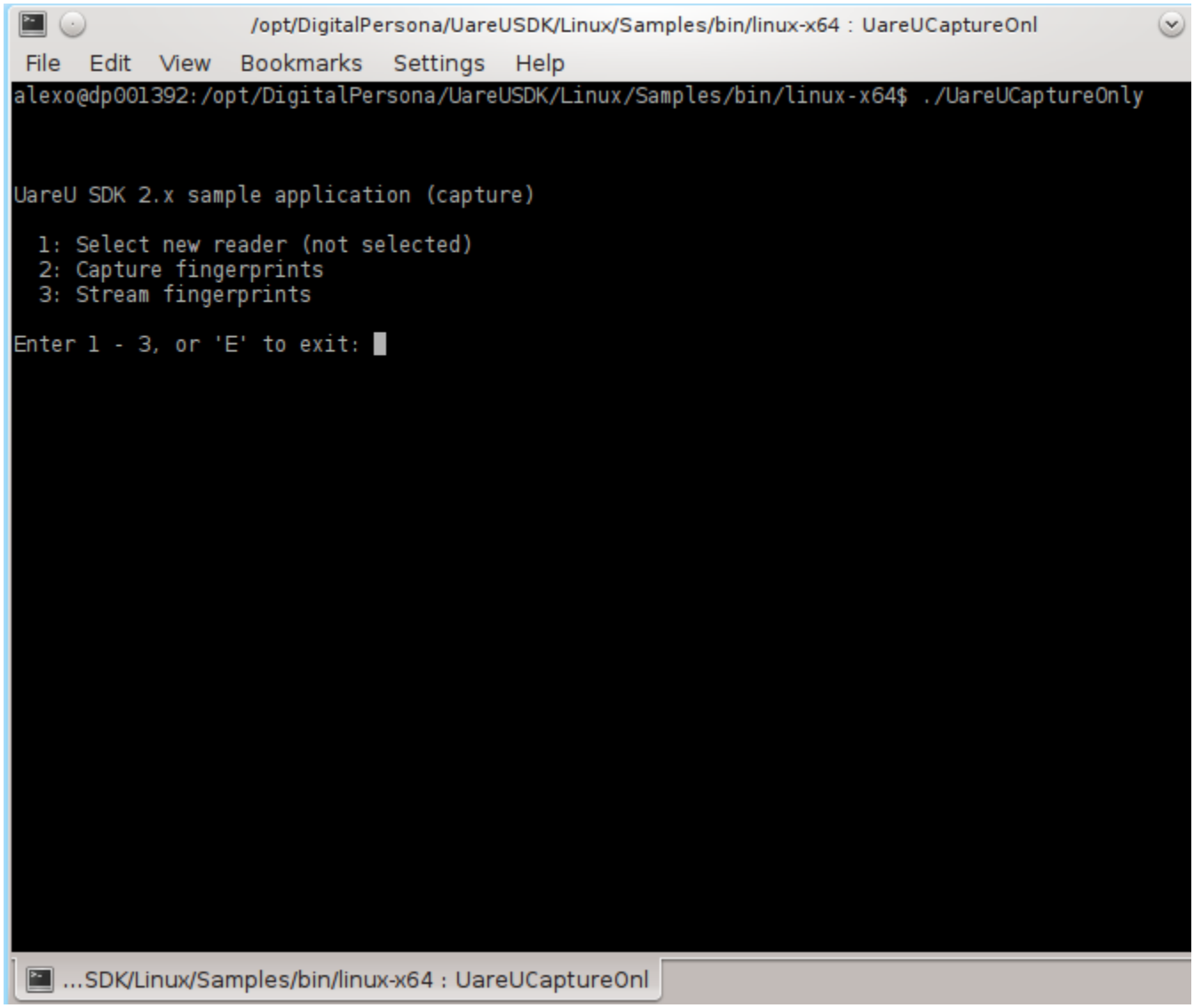
Once you have finished testing the enrollment feature of the sample application, press Ctrl C to return to the main menu.

This completes the demonstration of the `UareUSample` application. Press E to exit from the main menu.

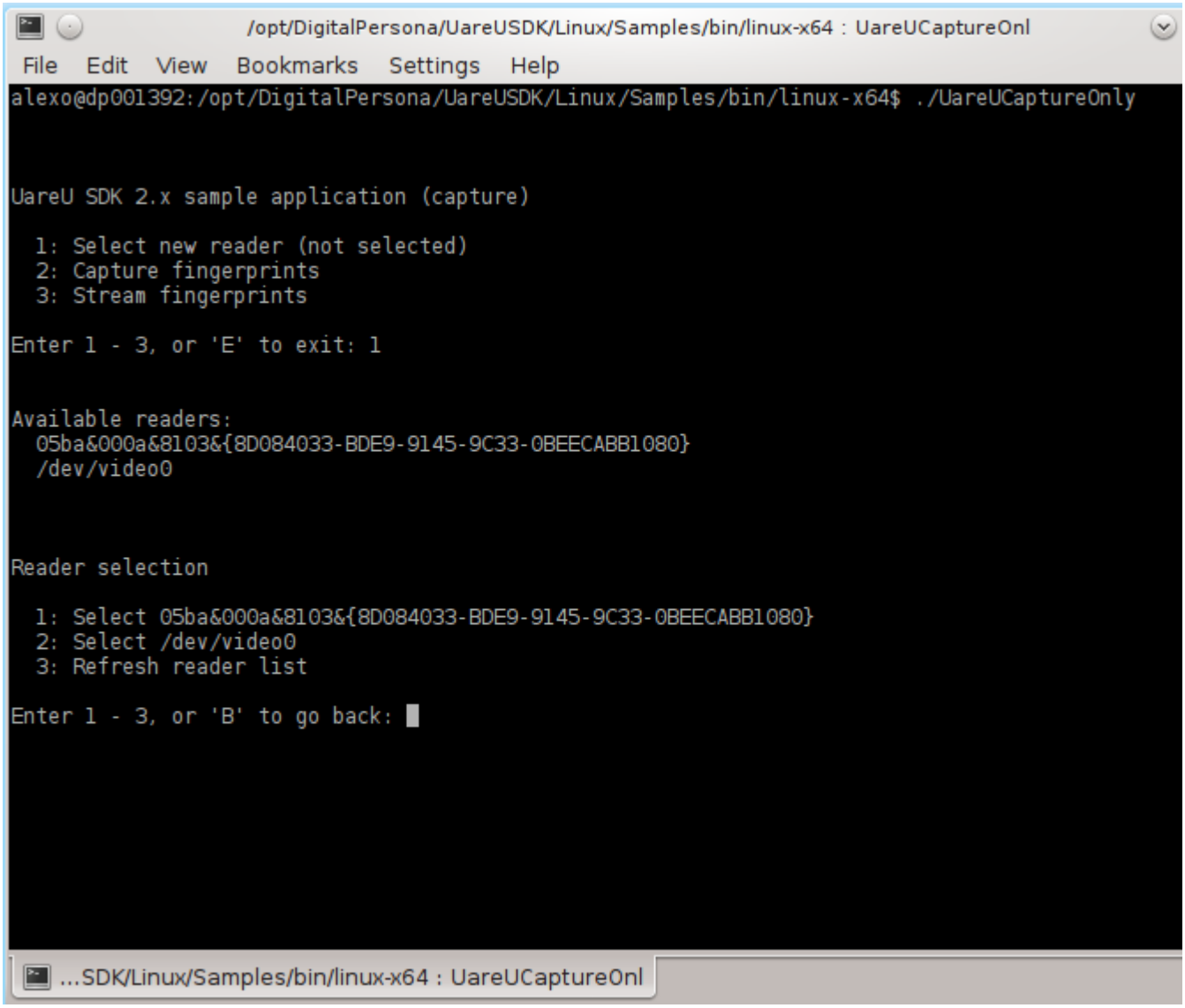
Running the UareUCaptureOnly Application

The `UareUCaptureOnly` application demonstrates the Capture API alone (no dependencies on the FingerJet Engine library). The `UareUCaptureOnly` sample application is a console applications -- it does not require any graphic libraries or desktop environment.

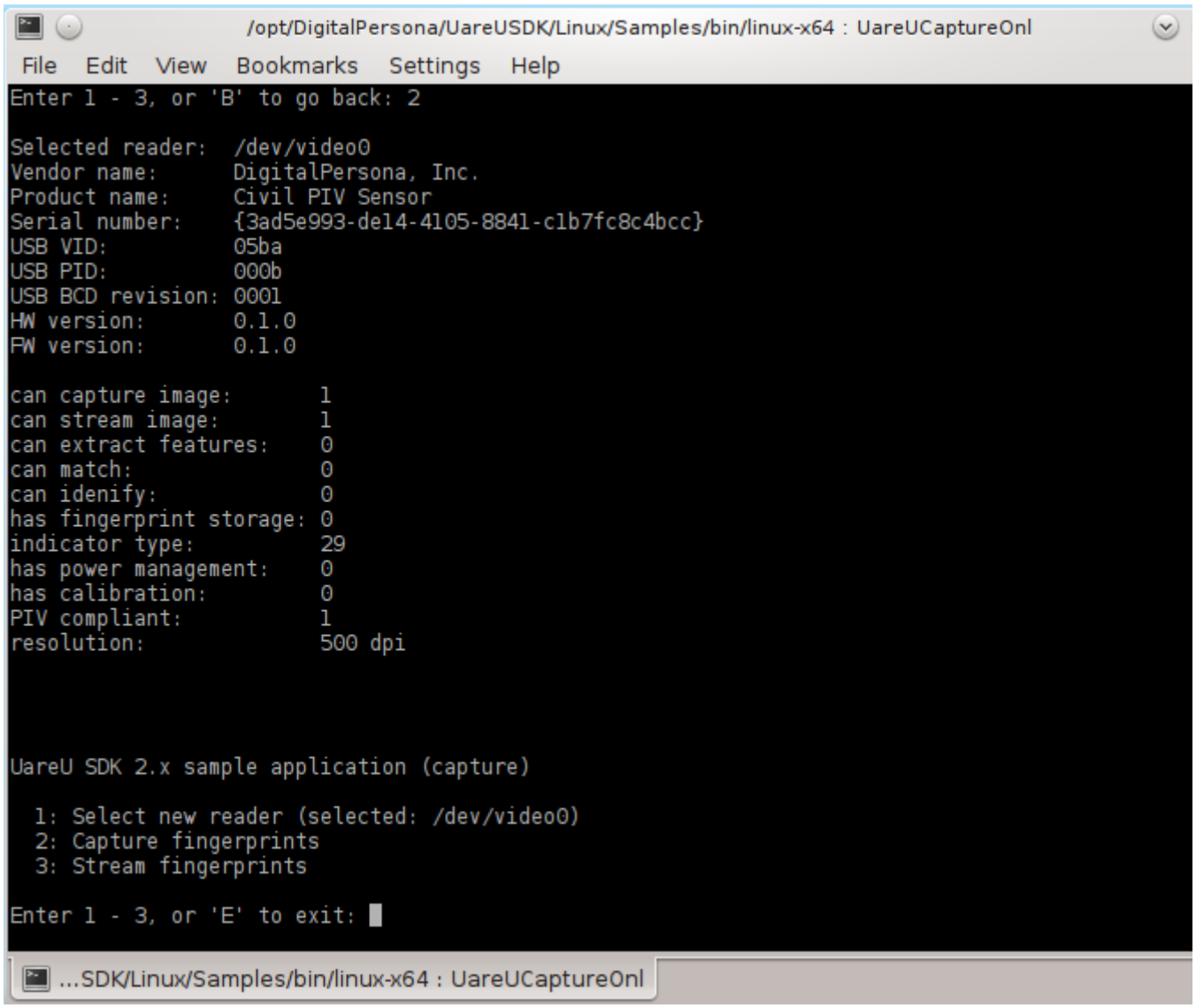
Be sure that LD_LIBRARY_PATH points to the location of the DigitalPersona libraries. When you launch the application you will see the main menu, as shown below:



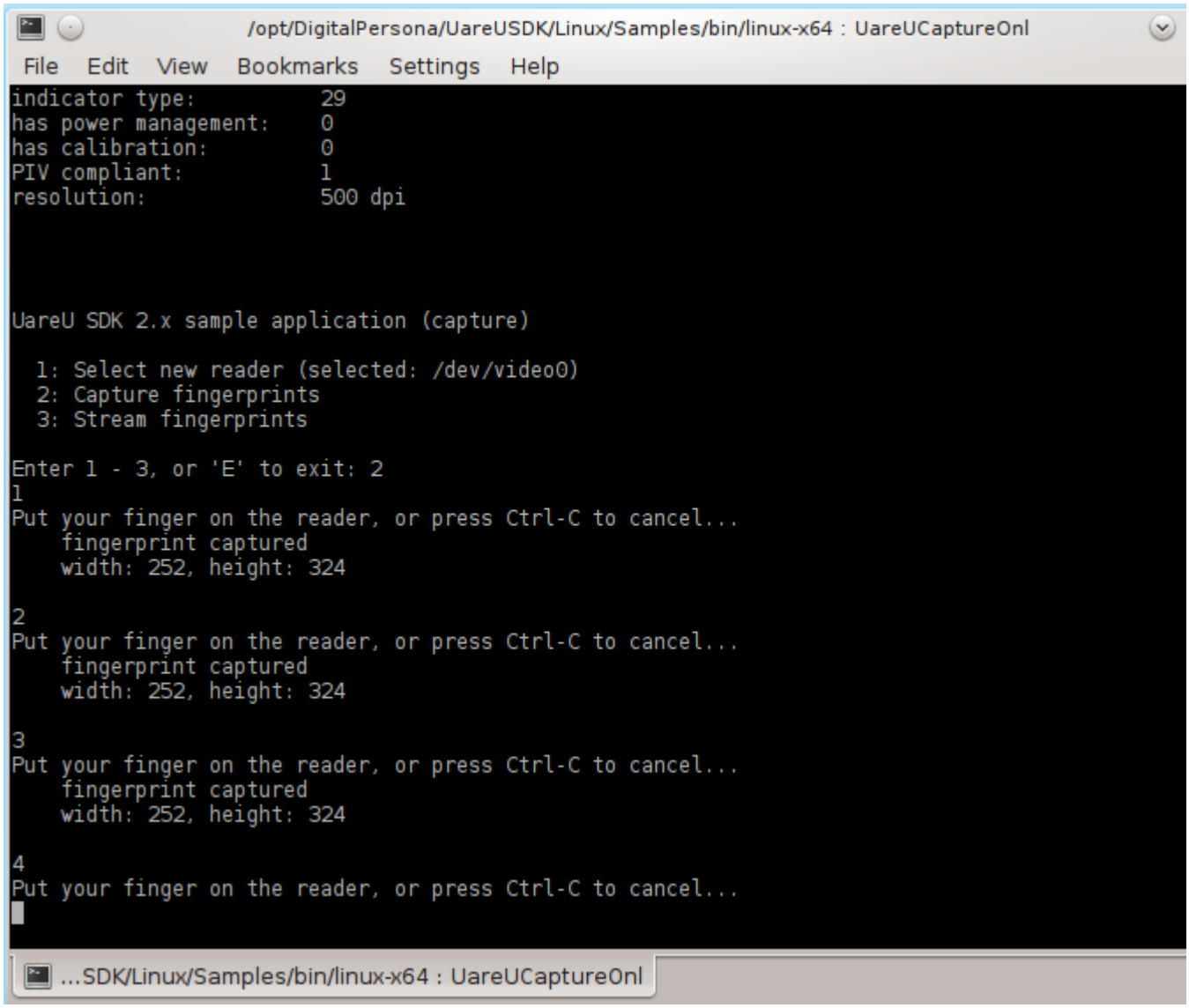
You must begin by selecting a fingerprint reader device to use with the application. After you press **1** to select the reader, the application will search for any readers and display a list of any readers it finds, as shown below. Choose the desired reader and press Enter to continue.



Once a reader is selected, you will see a screen that shows information about the reader and its capabilities, as shown below.

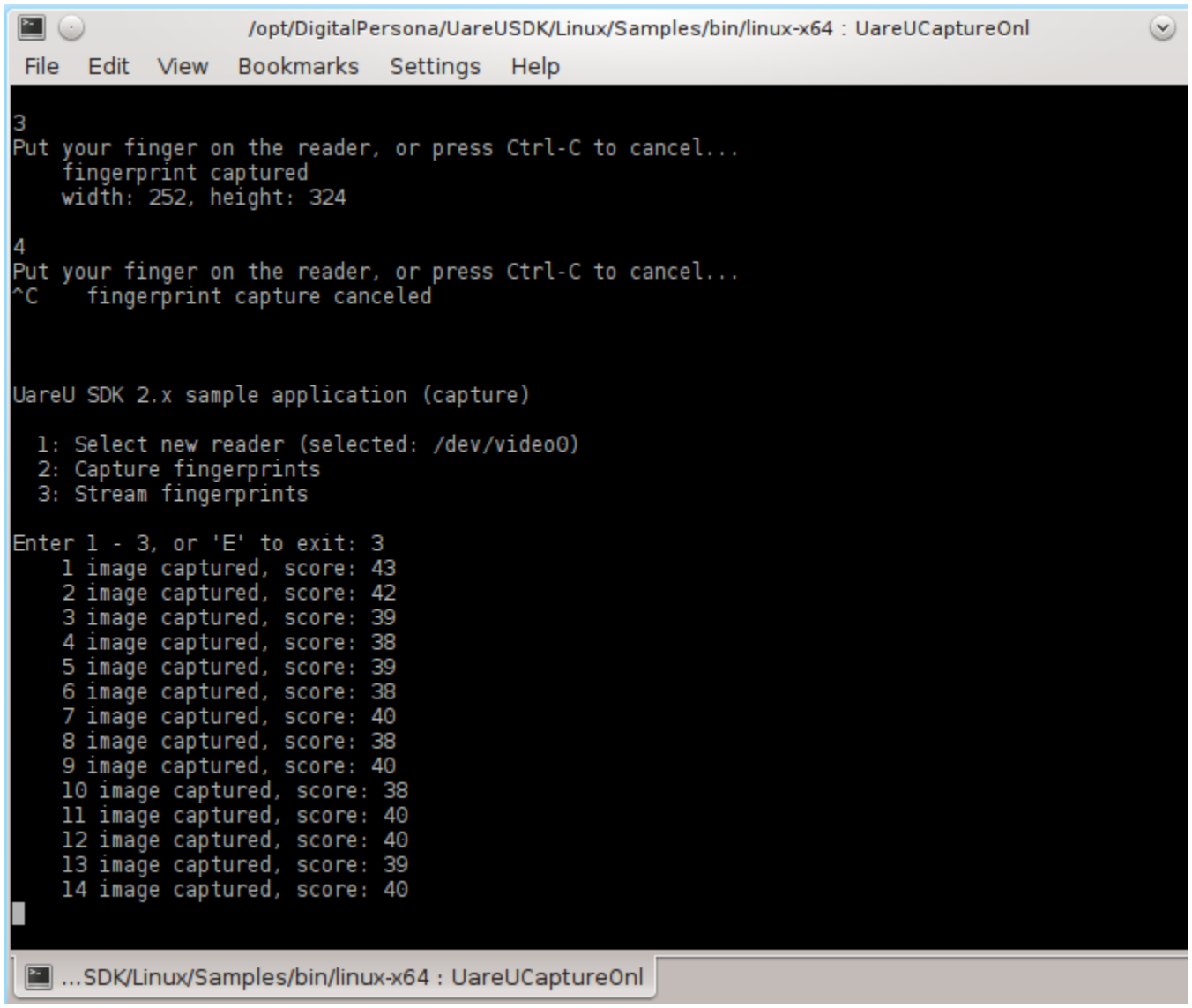


Select **2** and press Enter to test the capture function of the sample application. The capture function simply puts the reader into fingerprint capture mode and waits til a fingerprint is detected, as shown in the picture below.



Once you have finished testing the capture feature of the sample application, press Ctrl C to return to the main menu.

To test the streaming feature (not available on all fingerprint readers), select **3** and press Enter to test the streaming function of the sample application. The streaming function simply puts the reader into fingerprint stream mode and waits til a fingerprint is detected, as shown in the picture below.



As you press your finger on the reader, images will be captured and their quality score displayed. Press E to exit streaming mode.

This completes the demonstration of the `UareUCaptureOnly` application. Press E to exit from the main menu.

Developing Applications with Java 4

Pre-Requisites

This chapter assumes that you have a working knowledge of Java and that you know how to develop for Linux readers.

System Requirements

Development System

The development system can be any Linux system, as long as it has a kernel supported by the reader (as specified in *Developing Applications with C/C++* on *page 12*).

Target Runtime Hardware (Linux Reader)

The Linux-based reader that will run the application must be one of the following hardware platforms:

- Intel x86 architecture with CPU from 600MHz and at least 96MB of available RAM
- Intel x64 (x86-64) architecture with CPU from 600MHz and at least 96MB of available RAM

The file sizes are (in Kb):

	x86	**x64**
Capture runtime (drivers + SDK layer) with fingerprint recognition	100	120

In addition, the reader must also have:

- a USB port

The SDK works on a variety of hardware and is intended to have a small footprint so that it can run even on minimal hardware. Less capable hardware will work, but response time may not be optimal.

Installing on the Target Hardware

To install the run-time environment on the target hardware platform:

1. Install as described *Installation* on *page 6*.
2. Copy `/opt/DigitalPersona/UareUSDK/Linux/lib/java/dpuareu.jar` and `/opt/DigitalPersona/UareUSDK/Linux/lib/<x86 or x64 or arm>/libdpuareu_jni.so*` to the location of your choice.

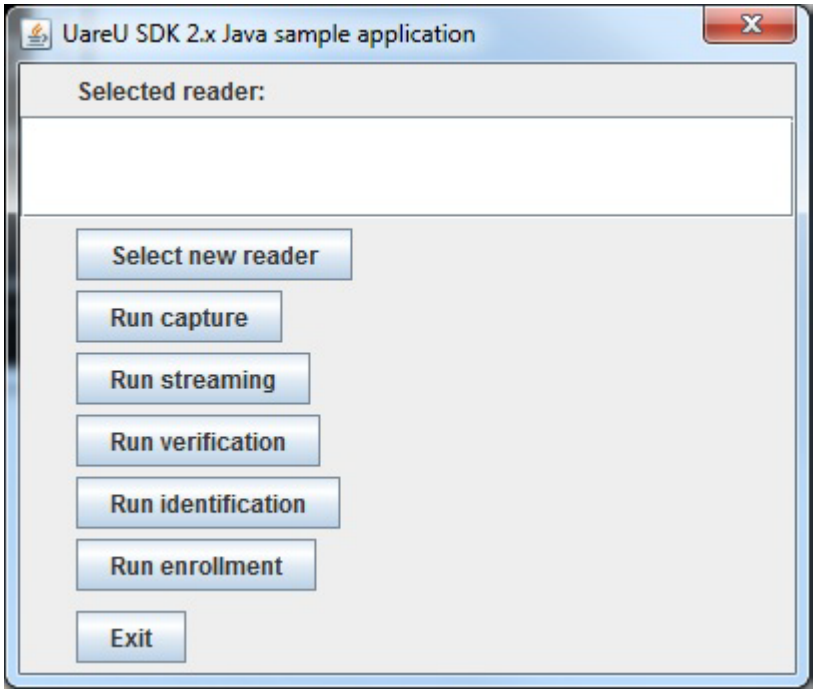
3. Make sure that `dpuareu.jar` is in the classpath and `libdpuareu_jni.so` is accessible by JVM. For example:

```
java -cp "./UareUSampleJava.jar:/opt/DigitalPersona/UareUSDK/Linux/lib/
java/dpuareu.jar" -Djava.library.path="/opt/DigitalPersona/UareUSDK/
Linux/lib/x86" UareUSampleJava
```

The Java Sample Application

U.are.U SDK includes a sample application to demonstrate the features of the SDK when using the Java API. The sample application is located in the `Samples` folder. The compiled file, `UareUSampleJava.exe` can be downloaded to your reader for testing or you can compile it for yourself using the source files provided.

The application demonstrates the features of the SDK. When you launch the application, you see the main screen as shown below.



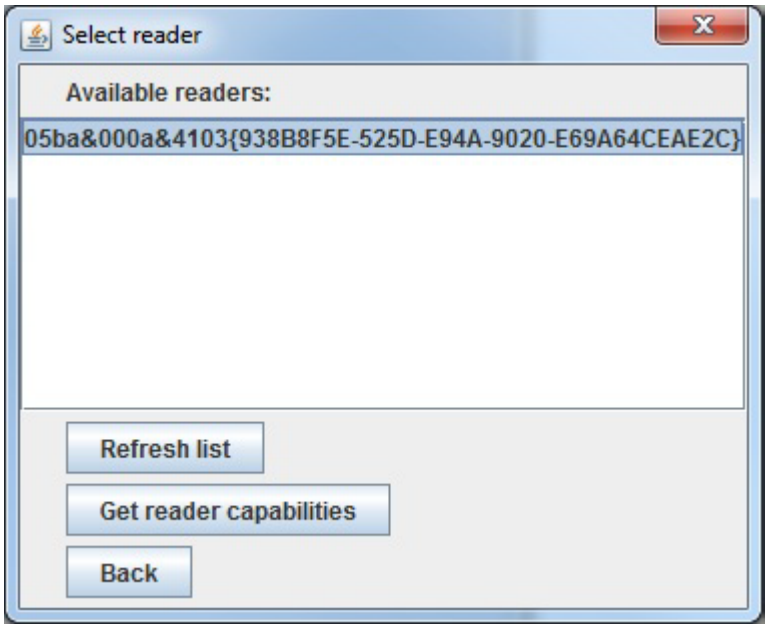
The sample program demonstrates:

- How to enroll a subject finger
- How to identify a fingerprint
- How to verify a fingerprint
- The built-in control for enrollment

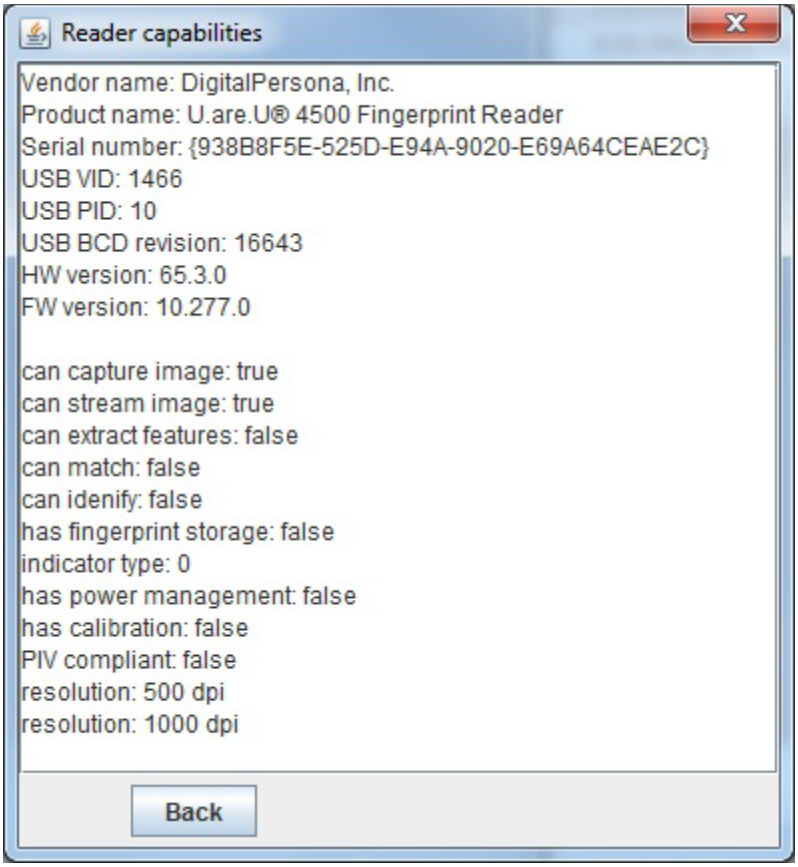
- The built-in control for identification
- How to use the streaming feature to display live fingerprint data on the screen

Selecting a Reader

To choose the reader, click on the **Select new reader** button. You will see a list of available readers and you can choose the desired device, as shown below:



To see the reader capabilities, click on the **Get reader capabilities** button. The capabilities will be displayed, as shown in the image below.



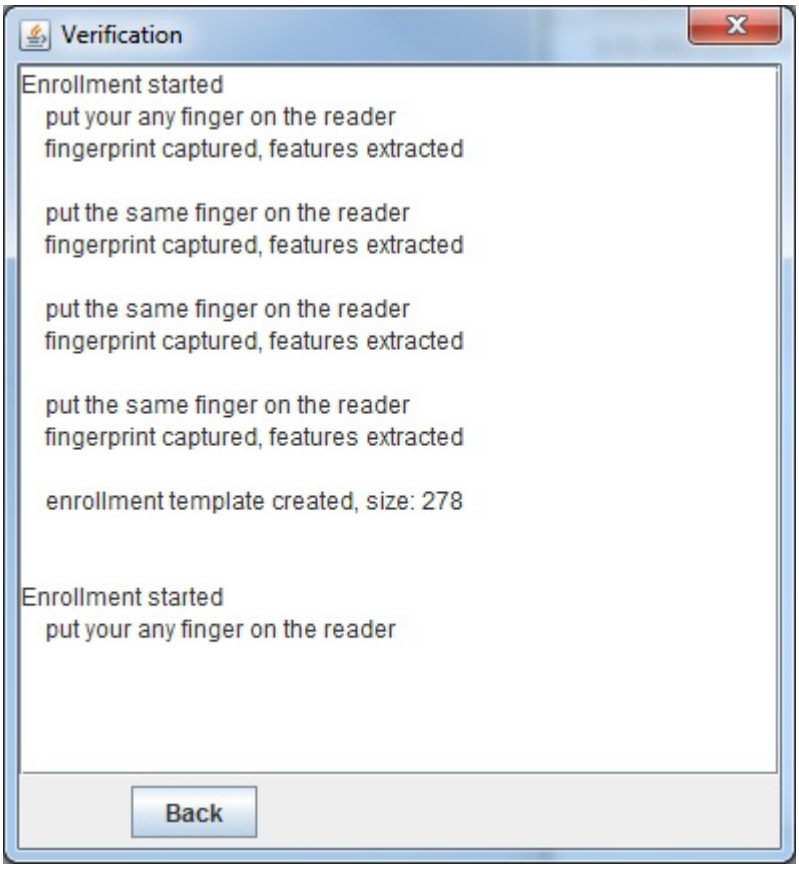
Click on the **Back** button to continue.

Click on the **Back** button from the previous screen to return to the main screen.

Enrolling a Finger

Click on **Run enrollment** to begin enrolling a test subject.

You will see a series of prompts to scan fingers for enrollment, as shown below.



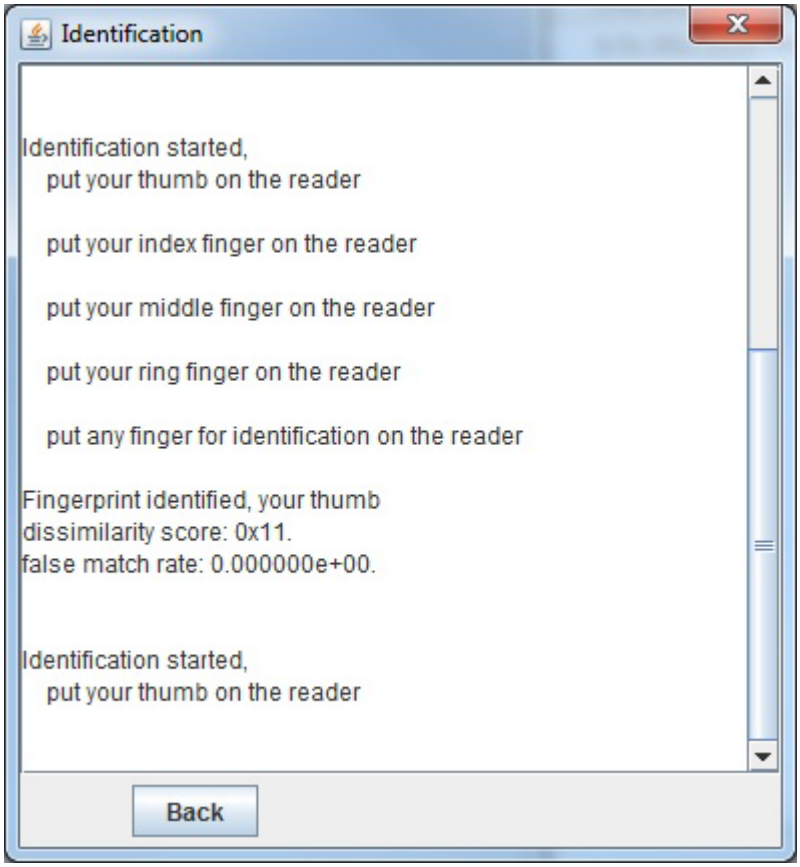
After the first finger is successfully scanned, you will be prompted to scan additional fingers until a sufficient number of high quality scans are complete. The number of fingers requested will vary depending on the image scans - the enrollment functions will continue to request scans until an acceptable enrollment record has been created.

When enrollment is complete, click **Back** to return to the main screen.

Identifying a Fingerprint

To test the identification feature, click on the **Run identification** button. Recall that identification is a 1-to-many comparison where the application searches through all of the enrolled fingers to find a match. For this example, we do not have a stored database, so the sample application first prompts you to put fingers on the reader so that it can create a FMD. After a FMD is created, you will be prompted to put any finger on the reader to identify against the FMD that was just created.

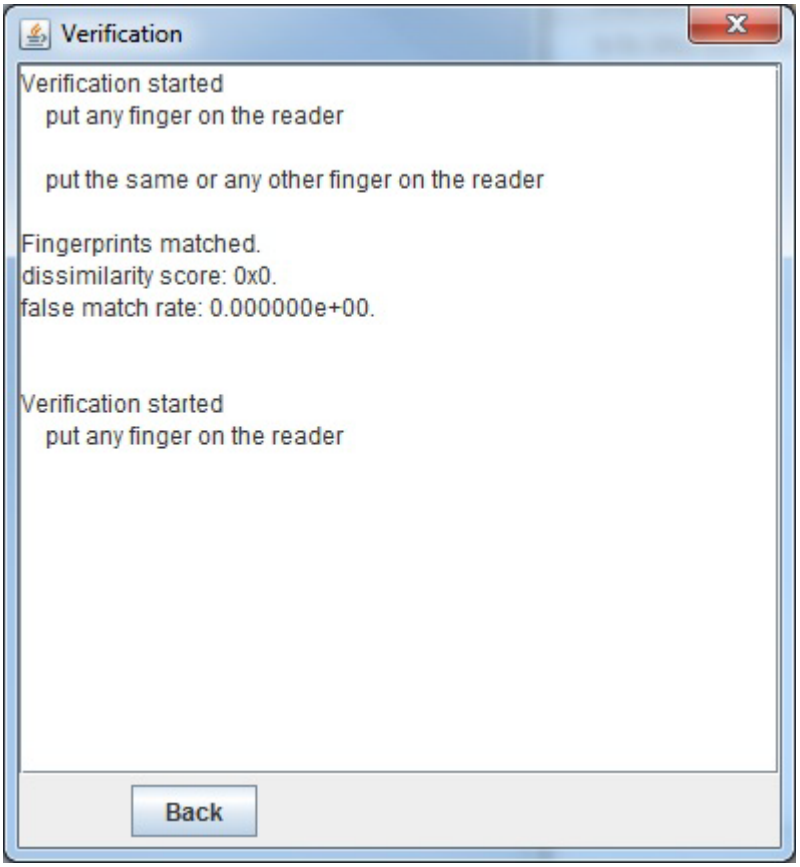
When you click the **Run identification** button, you will be prompted to place your fingers on the reader. If you press an enrolled finger on the reader, you will see that a match was found. In the screen image below, we successfully identified a user.



To exit identification mode, click on the **Back** button.

Verifying a Fingerprint

To test the verification feature, click on the **Run verification** button. Recall that verification is a 1-to-1 comparison where the application matches against a specified fingerprint. When you click the **Run verification** button, you will be prompted to place your finger on the reader. As with the identification example above, in the screen below, we have successfully verified a user.



To exit identification mode, click on the **Back** button.

Using the Capture and Streaming Feature

The sample application also demonstrates the streaming feature (on fingerprint readers that support that feature). To test capturing or streaming, from the main window, click on **Run capture** or **Run streaming** button.

This puts the reader into capture/streaming mode and immediately the results are displayed in the window. For streaming mode, the window then becomes like a live window on the reader as it streams results. Placing a finger on the reader displays the streamed fingerprint, as shown below.

For streaming, removing the finger shows a blank stream.

To exit capture / streaming mode, click on **Back**.

Developing Applications with JavaPOS 5

Pre-Requisites

This chapter assumes that you have a working knowledge of JavaPOS and that you know how to develop for Linux readers.

System Requirements

This section lists the minimum software and hardware requirements needed to work with U.are.U SDK.

Linux Kernel

For both target and development systems, the kernel object for the U.are.U 4500 reader will compile correctly only for Linux kernel versions 2.6.18 to 3.4.0

Development System

The development system can be any Linux system, as long as it has a kernel supported by the reader (as listed above).

Target Runtime Hardware (Linux Reader)

The Linux-based reader that will run the application must be one of the following hardware platforms:

- Intel x86 architecture with CPU from 600MHz and at least 96MB of available RAM
- Intel x64 (x86-64) architecture with CPU from 600MHz and at least 96MB of available RAM

The file sizes are (in KB):

	x86	**x64**
Capture runtime (drivers + SDK layer) with fingerprint recognition	1777	1796

Installation Notes

After installing the Java API on your development system as described on *Installation* on *page 6*, you need to do the following additional steps.

Installing on the Target Hardware

To install the run-time environment on the target hardware platform:

1. Install as described *Installation* on *page 6*.
2. Copy `/opt/DigitalPersona/UareUSDK/Linux/lib/java/*.jar` and `/opt/DigitalPersona/UareUSDK/Linux/lib/<x86 or x64 or arm>/libdpuareu_jni.so*` to the location of your choice.
3. Make sure that `dpuareu.jar` is in the classpath and `libdpuareu_jni.so` is accessible by JVM. For example:

```
java -cp "./UareUSampleJava.jar:/opt/DigitalPersona/UareUSDK/Linux/lib/
java/dpuareu.jar" -Djava.library.path="/opt/DigitalPersona/UareUSDK/
Linux/lib/x86" UareUSampleJava
```

Enabling and Disabling the USB Hotplug Device

IMPORTANT: You must be root to enable or disable the USB hotplug device.

To enable or disable the USB hotplug device

- Run `/UareUSDK/drivers/hotplug.sh`

Registering the DigitalPersona U.are.U Device Service

To enable DigitalPersona U.are.U support in your JavaPOS environment, you may need to register the DigitalPersona U.are.U Device Service.

To register your device

1. Modify the `JAVA_POS_CONFIG_PATH` variable in the `register.sh` file to point to the `config` folder. (This file is located in the `/opt/DigitalPersona/UareUSDK/Linux/lib/java` directory.)
2. Run `register.sh`

To unregister the Device Service

Run `register.sh -u`

Upgrading from Previous Versions of the JavaPOS API

To upgrade your existing applications, be sure to do the following steps:

1. Add a reference to `/opt/DigitalPersona/UareUSDK/Linux/Lib/Java/dpuareu.jar` in your classpath. This change is often done in a build or run script.
2. Replace the old `dpjavapos.jar` with the newest one, located in `/opt/DigitalPersona/UareUSDK/Linux/Lib/Java`.
3. Before running your application, set the path reference to the JavaPOS Wrapper JNI layer with this command:

```
Set LD_LIBRARY_PATH = Linux/Lib/Win32
OR
Set LD_LIBRARY_PATH = Linux/Lib/x64
```

The JavaPOS Sample Application

This section describes the functionality of the sample application, which is located in the `/opt/UareUSDK/Linux/Samples` directory. For more information about the sample application and the sample code, particularly button functionality, refer to the `readme.txt` file located in the same directory.

IMPORTANT: To run the sample application, Java runtime environment (JRE) 1.5 or higher must be installed on your computer.

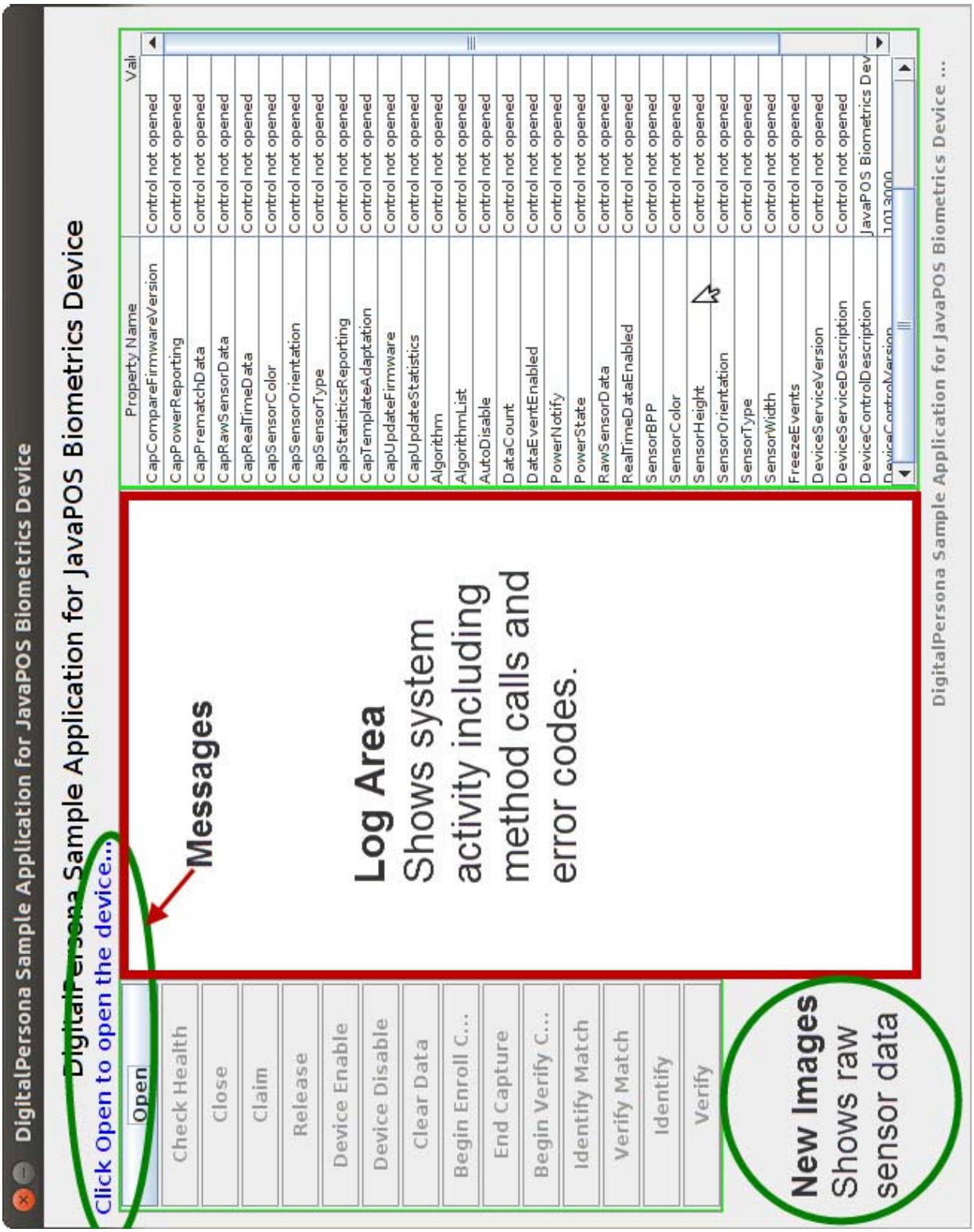
To start the application

1. Change to the `/opt/UareUSDK/Linux/Samples/bin/java` directory.
2. For x86 machines, run
   `./run_x86.sh`

   For x64 machines, run
   `./run_x64.sh`

   For ARM machines, run
   `./run_arm.sh`

   The **DigitalPersona** U.are.U SDK window appears as shown below.


DigitalPersona Sample Application for JavaPOS Biometrics Device
DigitalPersona Sample Application for JavaPOS Biometrics Device
Click Open to open the device...
Open
Check Health
Close
Claim
Release
Device Enable
Device Disable
Clear Data
Begin Enroll C...
End Capture
Begin Verify C...
Identify Match
Verify Match
Identify
Verify
Messages
Log Area
Shows system activity including method calls and error codes.
Property Name
Val
CapCompareFirmwareVersion
CapPowerReporting
CapPrematchData
CapRawSensorData
CapRealTimeData
CapSensorColor
CapSensorOrientation
CapSensorType
CapStatisticsReporting
CapTemplateAdaptation
CapUpdateFirmware
CapUpdateStatistics
Algorithm
AlgorithmList
AutoDisable
DataCount
DataEventEnabled
PowerNotify
PowerState
RawSensorData
RealTimeDataEnabled
SensorBPP
SensorColor
SensorHeight
SensorOrientation
SensorType
SensorWidth
FreezeEvents
DeviceServiceVersion
DeviceServiceDescription
DeviceControlDescription
DeviceControlVersion
Control not opened
JavaPOS Biometrics Dev
1013000
New Images
Shows raw sensor data
DigitalPersona Sample Application for JavaPOS Biometrics Device ...

The U.are.U SDK window is made up of the following four areas:

- Buttons area

  This area is located at the left of the window and contains buttons that initiate calls to various methods for interacting with the fingerprint reader and for performing fingerprint enrollment, verification, and identification operations.

- Messages area

  This area is located above the Buttons area and displays messages that inform the user of system activity, invite the user to perform actions such as touching the fingerprint reader, or advise the user of system errors. The message that appears when you start the application is "Click to open the device...".

- New Image area

  This area is located at the bottom left and displays raw sensor data when a StatusUpdate event is returned signaling raw data is available.

- Log area

  This area is located in the middle of the window and displays a log of system activity, including method calls and error codes.

- Properties area

  This area is located at the right of the window and displays a list of properties, both common and specific (in the **Property name** column), and their current values (in the **Value** column).

To open the connection with the fingerprint reader

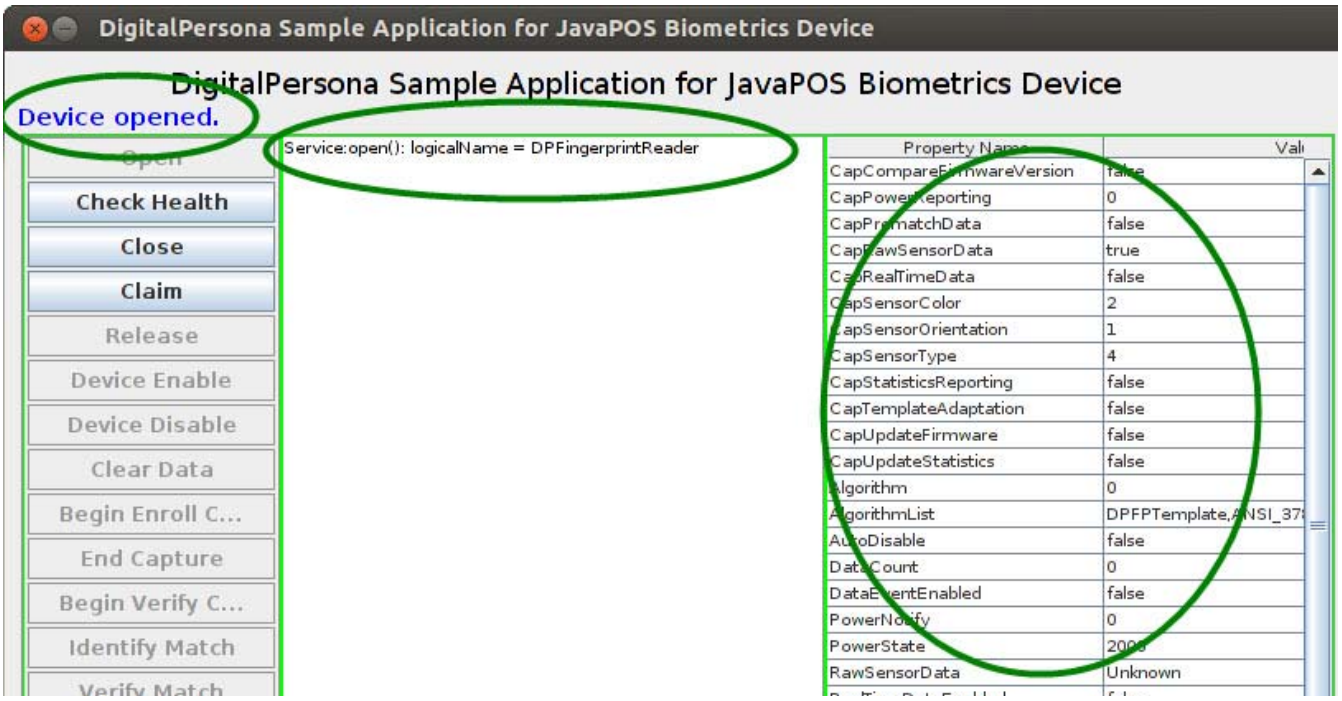
- Click **Open**.

  The **open** method of the Device Control object is called.

  If the call succeeds, the connection with the fingerprint reader is opened and various properties (common and specific) are set to their default values. These properties and values are displayed in the Properties area, and "Device opened..." appears in the Messages area, as shown in the screen shot below.

  NOTE: As each method is called, any properties that change are displayed in the Properties area.

If the method call fails, a failure message appears in the Messages area and error codes are displayed in the Log area.



Once the connection with the fingerprint reader has been opened, it must be claimed.

To claim the fingerprint reader

- Click **Claim**.

  The **claim** method of the Device Control is called, the **claimed** property is set to true, and "Exclusive accessed" appears in the Messages area.

Once the connection with the fingerprint reader has been claimed, it must be enabled.

To enable the fingerprint reader

- Click **Device Enable**.

  The **deviceEnabled** property is set to true and "Physical Device Operational" appears in the Messages area.

Enrolling a fingerprint consists of capturing four fingerprint images, converting them into fingerprint pre-enrollment templates, and then creating an enrollment template from these templates.

To perform fingerprint enrollment

1. Click **Begin Enroll Capture**.

   The **beginEnrollCapture** method of the Device Control is called and "Touch the sensor four times" appears in the Messages area.

2. Touch the fingerprint reader four times. Follow the instructions that appear in the Messages area to guide you.

   If the method call succeeds, an enrollment template is created and "Total enrollment completed: N" appears in the Messages area, where N is the number of total enrollments.

   If the method call fails, a failure message appears in the box in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

To perform fingerprint verification

1. Click **Begin Verify Capture**.

   The **beginVerifyCapture** method of the Device Control is called and "Touch the sensor to capture sample data" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

2. Touch the fingerprint reader.

   If the method call succeeds, a verification template is created and "Sample Data Captured" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

3. Click **Verify Match**.

   The **verifyMatch** method of the Device Control is called.

   If the method call succeeds, a match is performed using the latest enrollment template available and the verification template that was created in step 2, and "Verification success!" or "Verification failed!" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear the Messages area, and error codes are displayed in the Log area.

To perform fingerprint identification

1. Click **Begin Verify Capture**.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

2. Touch the fingerprint reader.

   If the method call succeeds, a verification template is created and "Sample Data Captured" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

3. Click **Identify Match**.

   The **identifyMatch** method of the Device Control is called.

   If the method call succeeds, a match is performed using all of the enrollment templates available and the verification template that was created in step 2. A candidate ranking is generated by listing only the indices of the enrollment templates that match, and "Identification success!" or "Identification Failed!" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

To perform fingerprint verification using a verification template created on-the-fly

1. Click **Verify**.

   The **verify** method of the Device Control is called, and "Please touch the sensor for verification" appears in the Messages area.

2. Touch the fingerprint reader.

   If the method call succeeds, a verification template is created on-the-fly. Then a match is performed using the latest enrollment template available and the verification template, and "Verification success!" or "Verification failed!" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

To perform fingerprint identification using a verification template created on-the-fly

1. Click **Identify**.

   The **identify** method of the Device Control is called and "Please touch the sensor for Identification" appears in the Messages area.

2. Touch the fingerprint reader.

   If the method call succeeds, a verification template is created on-the-fly. Then a match is performed using all of the enrollment templates available and the verification template. A candidate ranking is generated by listing only the indices of the enrollment templates that match, and "Identification success!" or "Identification Failed!" appears in the Messages area.

   If the method call fails, a failure message appears in the Messages area. If an error occurs, appropriate messages appear in the Messages area, and error codes are displayed in the Log area.

To clear the enrollment template array set and the verification template

- Click **Clear Data**.

  The **clearInput** method of the Device Control is called and "Clear data to start enrolling again" appears in the Messages area.

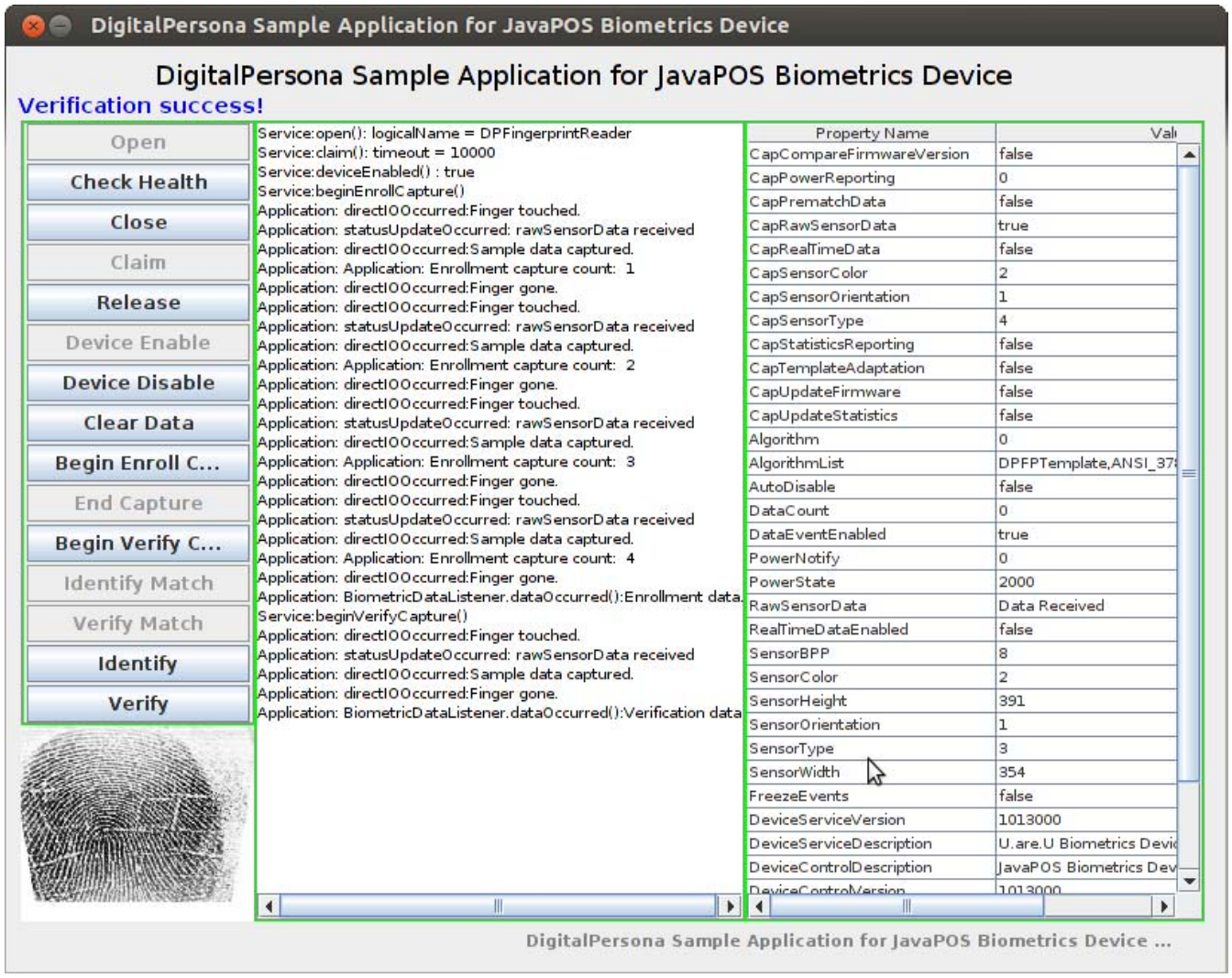
If the method call succeeds, the enrollment template array set and the verification template are cleared. A new verification template and a set of enrollment templates can now be created.

If the method call fails, a failure message appears in the Messages area, and error codes are displayed in the Log area.

Here is a demonstration of a sample sequence, showing the log area and messages.

In the screenshot below, the sample application window shows the following sequence of actions:

- Open
- Claim,
- Device enable,
- Begin enroll capture,
- Captured 4 fingers,
- Begin verify capture,
- Captured 1 finger,
- Verify match,
- Returns a success notification (final message at top left)


DigitalPersona Sample Application for JavaPOS Biometrics Device
DigitalPersona Sample Application for JavaPOS Biometrics Device
Verification success!
Open
Check Health
Close
Claim
Release
Device Enable
Device Disable
Clear Data
Begin Enroll C...
End Capture
Begin Verify C...
Identify Match
Verify Match
Identify
Verify
Service:open(): logicalName = DPFingerprintReader
Service:claim(): timeout = 10000
Service:deviceEnabled() : true
Service:beginEnrollCapture()
Application: directIOOccurred:Finger touched.
Application: statusUpdateOccurred: rawSensorData received
Application: directIOOccurred:Sample data captured.
Application: Application: Enrollment capture count: 1
Application: directIOOccurred:Finger gone.
Application: directIOOccurred:Finger touched.
Application: statusUpdateOccurred: rawSensorData received
Application: directIOOccurred:Sample data captured.
Application: Application: Enrollment capture count: 2
Application: directIOOccurred:Finger gone.
Application: directIOOccurred:Finger touched.
Application: statusUpdateOccurred: rawSensorData received
Application: directIOOccurred:Sample data captured.
Application: Application: Enrollment capture count: 3
Application: directIOOccurred:Finger gone.
Application: directIOOccurred:Finger touched.
Application: statusUpdateOccurred: rawSensorData received
Application: directIOOccurred:Sample data captured.
Application: Application: Enrollment capture count: 4
Application: directIOOccurred:Finger gone.
Application: BiometricDataListener.dataOccurred():Enrollment data.
Service:beginVerifyCapture()
Application: directIOOccurred:Finger touched.
Application: statusUpdateOccurred: rawSensorData received
Application: directIOOccurred:Sample data captured.
Application: directIOOccurred:Finger gone.
Application: BiometricDataListener.dataOccurred():Verification data
Property Name
Val
CapCompareFirmwareVersion false
CapPowerReporting 0
CapPrematchData false
CapRawSensorData true
CapRealTimeData false
CapSensorColor 2
CapSensorOrientation 1
CapSensorType 4
CapStatisticsReporting false
CapTemplateAdaptation false
CapUpdateFirmware false
CapUpdateStatistics false
Algorithm 0
AlgorithmList DPFPTemplate,ANSI_37
AutoDisable false
DataCount 0
DataEventEnabled true
PowerNotify 0
PowerState 2000
RawSensorData Data Received
RealTimeDataEnabled false
SensorBPP 8
SensorColor 2
SensorHeight 391
SensorOrientation 1
SensorType 3
SensorWidth 354
FreezeEvents false
DeviceServiceVersion 1013000
DeviceServiceDescription U.are.U Biometrics Devic
DeviceControlDescription JavaPOS Biometrics Dev
DeviceControlVersion 1013000
DigitalPersona Sample Application for JavaPOS Biometrics Device ...

To close the connection with the fingerprint reader

- Click **Close**.

  The **Close** method of the Device Control is called.

  If the method call succeeds, the connection with the fingerprint reader is closed, all of the controls other than the **Open** button are disabled, and the properties are reset, or cleared.

  If the method call fails, a failure message appears in the Messages area, and error codes are displayed in the Log area.

To close the application

Click the **Close** button at the top right of the window.

Redistribution 6

You may redistribute the files in the `redist` folder in the U.are.U SDK software package (typically located at `/opt/DigitalPersona/UareUSDK/redist`) to your end users pursuant to the terms of the end user license agreement (EULA), attendant to the software and located in the `/opt/DigitalPersona/UareUSDK/Linux/Docs` folder in the SDK software package.

When you develop a product based on the U.are.U SDK, you need to provide the redistributables to your end users. These files are designed and licensed for use with your application.

Per the terms of the EULA, DigitalPersona grants you a non-transferable, non-exclusive, worldwide license to redistribute, either directly or via the respective merge modules, the following files contained in the `redist` folder in the U.are.U SDK software package to your end users and to incorporate these files into derivative works for sale and distribution.

Merge Modules

The table below shows the merge modules that are required for each platform.

		C/C++		Java		JavaPOS	
Merge Module File	**Description**	**x86**	**x64**	**x86**	**x64**	**x86**	**x64**
DpDrivers	Device Components	x	x	x	x	x	x
DpPolicies_OTW		x	x	x	x	x	x
DpCore	Required components	x	x	x	x	x	x
DpCore_x64			x		x		x
DpProCore	DigitalPersona Workstation Pro components	x		x		x	
DpProCore_x64			x		x		x

Fingerprint Reader Documentation

You may redistribute the documentation included in the `/redist` directory to your end users pursuant to the terms of this section and of the EULA, attendant to the software and located in the `/Docs` directory.

Hardware Warnings and Regulatory Information

If you distribute DigitalPersona U.are.U fingerprint readers to your end users, you are responsible for advising them of the warnings and regulatory information included in the **Warnings and Regulatory Information.pdf**

file in the `redist` directory. You may copy and redistribute the language, including the copyright and trademark notices, set forth in the **Warnings and Regulatory Information.pdf** file.

Fingerprint Reader Use and Maintenance Guide

The DigitalPersona U.are.U Fingerprint Reader Use and Maintenance Guide, **DigitalPersona Reader Maintenance.pdf**, is located in the `/redist` directory. You may copy and redistribute the **DigitalPersona Reader Maintenance.pdf** file, including the copyright and trademark notices, to those who purchase a U.are.U module or fingerprint reader from you.