

DigitalPersona, Inc.

DigitalPersona Pro SDK

.NET Edition

Version 4.3

Developer Guide



digitalPersona.

DigitalPersona, Inc.

© 1996–2008 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.

DigitalPersona, U.are.U, and One Touch are trademarks of DigitalPersona, Inc., registered in the United States and other countries. Adobe and Adobe Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Microsoft, Visual C++, Visual Studio, Windows, Windows Server, and Windows Vista are registered trademarks of Microsoft Corporation in the United States and other countries.

This guide and the software it describes are furnished under license as set forth in the “License Agreement” that is shown during the installation process.

Except as permitted by such license or by the terms of this guide, 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 guide 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

Upon your purchase of a Developer Support package (available from <http://buy.digitalpersona.com>), you are entitled to a specified number of hours of telephone and email support.

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

Table of Contents

1	Introduction	1
	Target Audience	2
	Chapter Overview	2
	Document Conventions	3
	Notational Conventions	3
	Typographical Conventions	3
	Naming Conventions	3
	Additional Resources	4
	Related Documentation	4
	Online Resources	4
	System Requirements	4
	Supported DigitalPersona Products	5
	Fingerprint Template Compatibility	5
2	Quick Start	6
	Quick Concepts	6
	Install the Software	6
	Using the UI Support Sample Application	7
	Using the Engine Sample Application	9
3	Installation	11
	Installing the SDK	11
	Installing the Runtime Environment (RTE)	13
	Installing and Uninstalling the RTE Silently	16
4	Overview	17
	Pro SDK Terminology and Concepts	17
	Old and New Terms	17
	Operations	17
	Priorities	18
	User Interface Support	18
	Fingerprint Verification/Authentication	18
	Fingerprint Identification	18
	Operation Workflows	20
	Fingerprint Acquisition Operation Workflows	20
	Performing Fingerprint Authentication without UI Support	20
	Performing Fingerprint Authentication with UI Support	22
	Performing Fingerprint Identification with Authentication without UI Support	23

	Performing Fingerprint Identification with Authentication with UI Support	23
5	API Reference	24
	Components	24
	Exceptions	24
	Capture component	25
	DPPRO.Capture	25
	DPPRO.Capture.CaptureFeedback	27
	DPPRO.Capture.CapturePriority	28
	DPPRO.Capture.CaptureEventHandler	28
	DPPRO.Capture.ReadersCollection	29
	DPPRO.ReaderDescription	31
	DPPRO.ReaderDescription.ReaderImpressionType	32
	DPPRO.ReaderDescription.ReaderSerialNumberType	32
	DPPRO.ReaderDescription.ReaderTechnology	32
	DPPRO.ReaderDescription.ReaderVersion	33
	Error Components	34
	DPPRO.Error.ErrorCodes	34
	SDKException	35
	GUI Controls	36
	DPPRO.InterfaceMode	36
	DPPRO.IdentifyUser	36
	DPPRO.IdentifyUser.Status	37
	DPPRO.VerifyUser	38
	DPPRO.VerifyUser.Status	38
	Identification & Authentication Component	39
	DPPRO.Credential	39
	DPPRO.DataLocation	39
	DPPRO.FingerPrint	40
	DPPRO.Password	40
	DPPRO.Secret	40
	DPPRO.User	44
	DPPRO.User.UserNameType	46
	DPPRO.User.UserInfo	47
6	Graphical User Interfaces	48
	DPPRO.VerifyUser Graphical User Interface	48
	DPPRO.IdentifyUser Graphical User Interface	50

7	Developing Citrix-aware applications	54
8	Redistribution	55
	RTE\Install Folder	55
	Redist Folder	55
	Fingerprint Reader Documentation	59
	Hardware Warnings and Regulatory Information	59
	Fingerprint Reader Use and Maintenance Guide	59
A	Glossary	60
	Index	64

The DigitalPersona Pro SDK: .NET Edition is a software development tool that enables developers to integrate fingerprint biometrics into a wide set of Microsoft® Windows®-based applications, services, and products. The tool enables developers to perform basic fingerprint biometric operations: capturing a fingerprint from a DigitalPersona fingerprint reader, extracting the distinctive features from the captured fingerprint sample, and storing the resulting data in a template for later comparison of a submitted fingerprint with an existing fingerprint template.

In addition, the DigitalPersona Pro SDK: .NET Edition enables developers to use a variety of programming languages in a number of development environments to create their applications. The product includes detailed documentation and sample code that can be used to guide developers to quickly and efficiently produce fingerprint biometric additions to their products.

The DigitalPersona Pro SDK: .NET Edition builds on a decade-long legacy of fingerprint biometric technology, being the most popular set of development tools with the largest set of enrolled users of any biometric product in the world. Because of its popularity, the DigitalPersona® Fingerprint Recognition Engine software—with its high level of accuracy—and award-winning U.are.U® Fingerprint Reader hardware have been used with the widest-age, hardest-to-fingerprint demographic of users in the world.

The DigitalPersona Pro SDK: .NET Edition has been designed to authenticate users on the Microsoft® Windows Vista® and Microsoft® Windows® XP operating systems running on any of the x86-based platforms. The product is used with DigitalPersona fingerprint readers in a variety of useful configurations: standalone USB peripherals, modules that are built into customer platforms, and keyboards. The DigitalPersona One Touch I.D. SDK product can also be implemented along with the DigitalPersona Pro SDK: .NET Edition to add fast fingerprint identification capability to a developer's design.

Fingerprint Authentication on a Remote Computer

This SDK includes transparent support for fingerprint authentication through Windows Terminal Services (including Remote Desktop Connection) and through a Citrix connection to Metaframe Presentation Server using a client from the Citrix Presentation Server Client package.

Through Remote Desktop or a Citrix session, you can use a local fingerprint reader to log on to, and use other installed features of, a remote machine running your fingerprint-enabled application.

The following types of Citrix clients are supported:

- • Program Neighborhood
- • Program Neighborhood Agent
- • Web Client

Note that to take advantage of this feature, your fingerprint-enabled application must run on the Terminal Services or Citrix server, not on the client. If you are developing a Citrix-aware application, see additional information in the *Developing Citrix-aware applications* chapter on page 54.

Target Audience

This guide is for developers who have a working knowledge of the Microsoft® .NET programming language, and the Microsoft® Visual Studio® 2005 (or later) development environment.

Chapter Overview

Chapter 1, Introduction (this chapter), describes the audience for which this guide is written; defines the typographical, notational, and naming conventions used throughout this guide; cites a number of resources that may assist you in using the DigitalPersona Pro SDK: .NET Edition; identifies the minimum system requirements needed to run the DigitalPersona Pro SDK: .NET Edition; and lists the DigitalPersona products and fingerprint templates supported by the DigitalPersona Pro SDK: .NET Edition.

Chapter 2, Quick Start, provides a quick introduction to the DigitalPersona Pro SDK: .NET Edition using one or more of the sample applications provided as part of the SDK.

Chapter 3, Installation, contains instructions for installing the various components of the product and identifies the files and folders that are installed on your hard disk.

Chapter 4, Overview, introduces DigitalPersona Pro SDK: .NET Edition terminology and concepts. This chapter also includes typical workflow diagrams and explanations of the One Touch for Windows: .NET Edition API components used to perform the tasks in the workflows.

Chapter 5, API Reference, defines the components that are used for developing applications based on the One Touch for Windows: .NET Edition API.

Chapter 6, Graphical User Interfaces, describes the functionality of the graphical user interfaces included with the Verify and Identify control objects.

Chapter 8, Redistribution, identifies the files that you may distribute according to the End User License Agreement (EULA) and lists the functionalities that you need to provide to your end users when you develop products based on the One Touch for Windows: .NET Edition API.

A glossary and an index are also included for your reference.

Document Conventions

This section defines the notational, typographical, and naming conventions used in this guide.

Notational Conventions

The following notational conventions are used throughout this guide:

NOTE: Notes provide supplemental reminders, tips, or suggestions.

IMPORTANT: Important notations contain significant information about system behavior, including problems or side effects that can occur in specific situations.

Typographical Conventions

The following typographical conventions are used in this guide:

Typeface	Purpose	Example
Bold	Used for keystrokes and window and dialog box elements and to indicate data types	Click Next . The UI Sample window displays. String that contains a fingerprint reader serial number
Courier bold	Used to indicate computer programming code	Check the TemplateStatus property after each call to the AddFeatures method. Initialize a new instance of the DPPRO.Capture.Capture class.
<i>Italics</i>	Used for emphasis or to introduce new terms If you are viewing this document online, clicking underlined text in italics may also activate a hypertext link to other areas in this guide or to URLs.	<i>A fingerprint</i> is an impression of the ridges on the skin of a finger. (new term) See <i>Installing the SDK on page 8</i> . (link to heading and page)

Naming Conventions

All API members are enclosed in the DPPRO namespace.

Additional Resources

You can refer to the resources in this section to assist you in using the DigitalPersona Pro SDK: .NET Edition.

Related Documentation

Subject	Document
Fingerprint recognition, including the history and basics of fingerprint identification and the advantages of DigitalPersona's Fingerprint Recognition Engine	The DigitalPersona White Paper: Guide to Fingerprint Recognition. The file, Fingerprint Guide.pdf, is located in the Docs folder in the SDK software package, and is not automatically installed on your computer as part of the setup process.
Late-breaking news about the product	The Readme.txt files provided in the root directory in the SDK software package as well as in some subdirectories

Online Resources

Web Site name	URL
DigitalPersona Developer Connection Forum for peer-to-peer interaction between DigitalPersona Developers	http://www.digitalpersona.com/webforums/
Latest updates for DigitalPersona software products	http://www.digitalpersona.com/support/downloads/software.php

System Requirements

This section lists the minimum software and hardware requirements needed to run the DigitalPersona Pro SDK: .NET Edition.

- x86-based processor or better
- Microsoft® Windows® 2000 SP3, Windows® XP Professional, Windows® Server 2003 or Windows Vista®
- Microsoft .NET Framework (version 2 and later), which is required for .NET projects to run and is obtainable from Microsoft
- USB connector on the computer where the fingerprint reader is to be connected
- DigitalPersona U.are.U 4000B or U.are.U 2500 fingerprint reader
- Administrator privileges

This section lists the minimum software and hardware requirements needed to run the Runtime Environment (RTE).

- x86-based processor or better
- Microsoft® Windows® 2000 SP3, Windows® XP Professional (32-bit only), Windows® Server 2003 or Windows Vista®
- Microsoft High Encryption Pack
- Microsoft .NET Framework (version 2 and later), which is required for .NET projects to run and is obtainable from Microsoft
- USB connector on the computer where the fingerprint reader is to be connected
- DigitalPersona U.are.U 4000B or U.are.U 2500 fingerprint reader
- Administrator privileges

In order to use the RTE, none of the following products should be installed.

- DigitalPersona Pro Server
- DigitalPersona Pro Workstation
- DigitalPersona Pro Kiosk
- DigitalPersona Personal
- DigitalPersona Password Manager
- DigitalPersona Online Client/Kiosk (any version lower than 4.2.0)

Supported DigitalPersona Products

The DigitalPersona Pro SDK: .NET Edition supports the following DigitalPersona products:

- DigitalPersona U.are.U 4000B fingerprint readers and modules
- DigitalPersona U.are.U 2500 fingerprint readers and modules
- DigitalPersona U.are.U Fingerprint Keyboard

Fingerprint Template Compatibility

Fingerprint templates produced by the DigitalPersona Pro SDK are also compatible with the following DigitalPersona SDKs:

- Gold SDK
- Gold CE SDK
- One Touch for Windows SDK, all editions
- One Touch for Linux SDK, all distributions

NOTE: Platinum SDK registration templates must be converted to a compatible format to work with these SDKs.

This chapter provides a quick introduction to the DigitalPersona Pro SDK: .NET Edition using two sample applications provided as part of the DigitalPersona Pro SDK: .NET Edition.

The sample applications are provided in both the .NET and C# languages as Microsoft Visual Studio 2005 projects that demonstrates the main functions of the SDK.

Quick Concepts

The following definitions will assist you in understanding the purpose and functionality of the sample applications that are described in this section.

Verification (or authentication)—The process of comparing a captured fingerprint to a fingerprint template to determine whether the two match.

Identification—The process of identifying a specific user from their fingerprint.

Install the Software

Before you can use the sample application, you must install the DigitalPersona Pro SDK: .NET Edition and one of the following:

- DigitalPersona Pro Workstation 4.3 or above
- DigitalPersona Pro Kiosk 4.3 or above
- DigitalPersona Pro RTE 4.3 or above.

Additionally you must have enrolled at least one fingerprint in Pro Workstation or Pro Kiosk.

To install the DigitalPersona Pro SDK: .NET Edition

1. In the SDK folder in the software package, open the Setup.exe file, and then click **Next**.
2. Follow the installation instructions as they appear.

To install the DigitalPersona Pro SDK: .NET Edition RTE

1. In the RTE folder in the software package, open the Setup.exe file, and then click **Next**.
2. Follow the installation instructions as they appear.
3. Restart your computer.

To use an external fingerprint reader DigitalPersona Pro SDK: .NET Edition

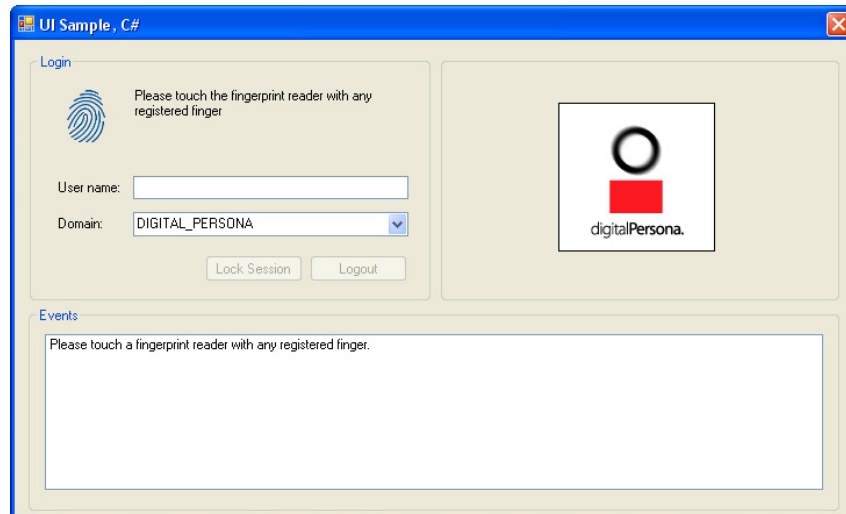
- Insert the fingerprint reader into the USB connector on the system where you installed the SDK.

Using the UI Support Sample Application

To start the Engine sample application

- Open the DPPROEngNetSample.exe file located in the <destination folder>\Pro SDK\.NET\Samples\Visual Studio 2005\CSharp\UI Support\Release folder. There is also a VB.NET version in the <destination folder>\Pro SDK\.NET\Samples\Visual Studio 2005\VBNET\UI Support\Release folder.

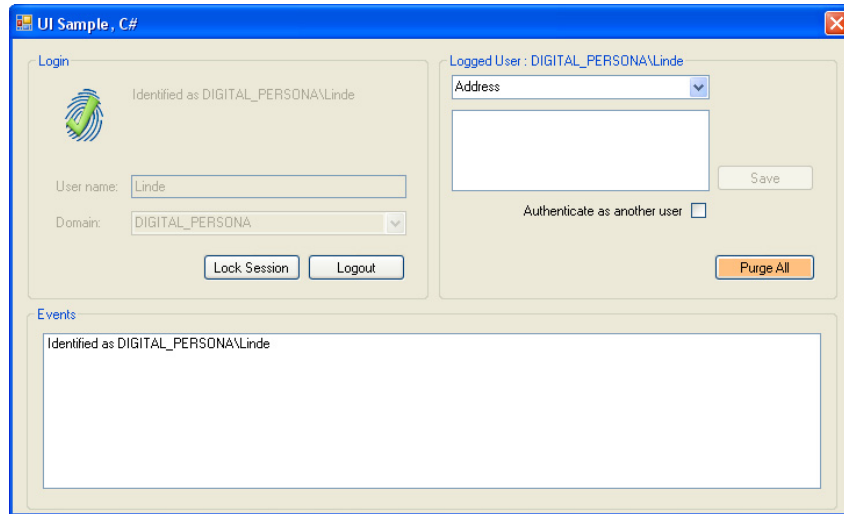
The **UI Support Sample** dialog box appears.



This dialog illustrates:

- events and related messages that are fired during identification, verification and Secret access processes.
- locking the application's edit mode and unlocking it with a fingerprint
- logging out and logging back in with a fingerprint and applicable username/domain, either through identification (fingerprint only) or verification (fingerprint plus username/domain).
- entering content into a Secret and saving the Secret
- purging the content of a Secret.

1. To enable identification, first a full authentication (username and domain) should be provided by the user, along with a matching fingerprint. If this specific user has already logged in and been fully authenticated, then identification is already enabled.



2. Click **Lock Session** to lock the session. You can then unlock the session with your fingerprint.
3. Click **Logout** to log out from the session. You can then log in with your fingerprint.
4. To enter content into a Secret, select a label from the dropdown menu and enter the content into the text box below and then click **Save**.
5. To delete the content of all Secrets, click **Purge all**.

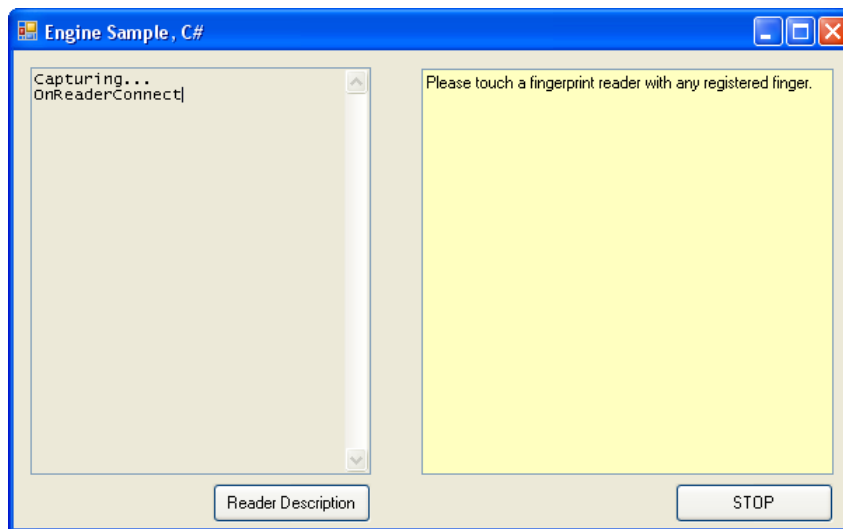
Using the Engine Sample Application

This sample illustrates obtaining Secret content saved in the UI Support sample application through fingerprint identification. For optimum results, use the UI Support application to save Secret content prior to running this sample.

To start the Engine sample application

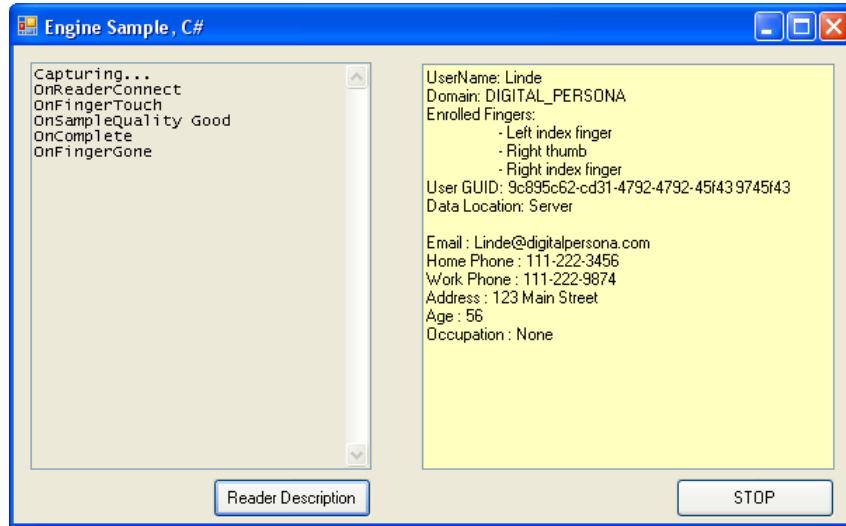
- Open the DPPROEngNetSample.exe file located in the <destination folder>\Pro SDK\.NET\Samples\Visual Studio 2005\CSharp\Engine\Release folder. There is also a VB.NET version in the <destination folder>\Pro SDK\.NET\Samples\Visual Studio 2005\VBNET\Engine\Release folder.

The **Engine Sample** dialog box appears.

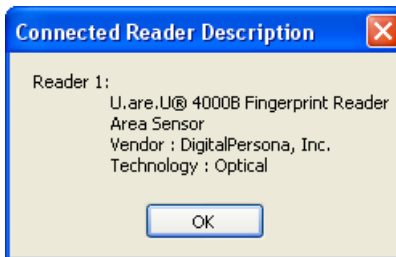


This dialog illustrates the events that are fired during the capture process, the display of user information after successful identification, the release of the user's Secret and the information obtained from the fingerprint reader.

1. Place or swipe your finger on the fingerprint reader to view additional events fired during the capture process.



2. Upon successful identification, user details are shown and the user's Secret is released. Note that in order to enable identification, a user should be authenticated first. This can be done through the UI Sample code.
3. Click **Reader Description** to view the information obtained about all the fingerprint readers that are available.



This chapter contains instructions for installing the various components of the DigitalPersona Pro SDK: .NET Edition and identifies the files and folders that are installed on your hard disk.

The following two installations are located in the SDK software package:

- SDK, which you use in developing your application. This installation is located in the SDK folder.
- RTE (runtime environment), which you must provide to your end users to support the DigitalPersona Pro SDK: .NET Edition API components. This installation is located in the RTE folder. (The RTE installation is also included in the SDK installation.)

Installing the SDK

To install the DigitalPersona Pro SDK: .NET Edition for 32-bit operating systems

1. In the SDK folder in the SDK software package, open the Setup.exe file, and then click **Next**.
2. Follow the installation instructions as they appear.
3. Restart your computer.

To install the DigitalPersona Pro SDK: .NET Edition for 64-bit operating systems

1. In the SDK\x64 folder in the SDK software package, open the Setup.exe file, and then click **Next**.
2. Follow the installation instructions as they appear.
3. Restart your computer.

Table 1 describes the files and folders that are installed in the *<destination folder>* folder on your hard disk for the 32-bit and 64-bit installations. The RTE files and folders, which are described in *Table 2* on *page 13* for the 32-bit installation and in *Table 3* on *page 15* for the 64-bit installation, are not installed as part of the SDK install, but must be installed separately.

Table 1. DigitalPersona Pro SDK: .NET Edition installed files and folders (32-bit and 64-bit)

Folder	File	Description
<INSTALLDIR>\DigitalPersona\Pro SDK\C-C++\Docs	DigitalPersona Pro SDK C++ Developer Guide.pdf	DigitalPersona Pro SDK: C++ Edition Developer Guide
<INSTALLDIR>\DigitalPersona\Pro SDK\C-C++\Include		C++ header files and other resources
<INSTALLDIR>\DigitalPersona\Pro SDK\C-C++\Lib		C++ libraries
<INSTALLDIR>\DigitalPersona\Pro SDK\C-C++\Samples		This folder contains C++ sample applications that demonstrate various aspects of the C/C++ API.
<INSTALLDIR>\DigitalPersona\Pro SDK\.NET\Docs	DigitalPersona Pro SDK .NET Developer Guide.pdf	DigitalPersona Pro SDK: .NET Edition Developer Guide
<INSTALLDIR>\DigitalPersona\Pro SDK\.NET\Bin	DPPROShrNET.dll DPPRODevNET.dll DPPROEngNET.dll DPPROGuiNET.dll	DLLs used by the DigitalPersona Pro SDK: .NET Edition API
<INSTALLDIR>\DigitalPersona\Pro SDK\.NET\Samples		This folder contains C++ sample applications that demonstrate various aspects of the .NET API.

Installing the Runtime Environment (RTE)

When you develop a product based on the DigitalPersona Pro SDK: .NET Edition, you need to provide certain redistributables to your end users. These files are designed and licensed for use with your application. You may include the installation files located in the RTE\Install folder in your application or you may incorporate the redistributables directly into your installer. You may also use the merge modules located in the Redist folder in the SDK software package to create your own MSI installer. (See *Redistribution* on page 55 for licensing terms.)

If you created an application based on the DigitalPersona Pro SDK: .NET Edition API that does not include an installer, your end users must install the DigitalPersona Pro SDK: .NET Edition Runtime Environment to run your application. The latest version of the RTE is available from the DigitalPersona Web site at <http://www.digitalpersona.com/support/downloads/software.php>.

To install the DigitalPersona Pro SDK: .NET Edition Runtime Environment for 32-bit operating systems

1. In the RTE folder in the SDK software package, open the Setup.exe file.
2. Follow the installation instructions as they appear.

Table 2. DigitalPersona Pro SDK: .NET Edition RTE installed files and folders, 32-bit installation

Folder	File	Description
<destination folder>\Bin	bsapi.dll DpBranding.dll DPCOper2.dll DPCrStor.dll DPDB.dll DPDevice2.dll DPDeviceAfss.cat DPDeviceAfss.dll DPDeviceAuthentec.dll DPDeviceUpekBs.dll DPDeviceValidity.dll DPDevTS.dll dpfailurescan.wav DpFnd2.dll DpHostW.exe DPILPro.dll DPmsg.dll DPMux.dll DPPS2.dll dpsuccessscan.wav, DpSvInfo2.dll, DPTSCInt.dll, KfsLib.dll	DLLs and executable file used by the all of the DigitalPersona Pro SDK APIs
<Citrix Folder>	DPICACnt.dll	

Table 2. DigitalPersona Pro SDK: .NET Edition RTE installed files and folders, 32-bit installation (*continued*)

Folder	File	Description
GlobalAssemblyCache	DPPROShrNET.dll DPPRODevNET.dll DPPROEngNET.dll DPPROGuiNET.dll	DLLs used by the DigitalPersona Pro SDK: .NET Edition API
<system folder>	DpClback.dll DPCOper2.dll DPCrStor.dll DPDevice2.dll DPFPApi.dll DpHostW.exe DPmsg.dll DPMux.dll dpHFtrEx.dll dpHMatch.dll DpSDApi.dll	DLLs used by all of the DigitalPersona Pro SDK editions

To install the One Touch for Windows: .NET Edition Runtime Environment for 64-bit operating systems

1. In the RTE\x64 folder in the SDK software package, open the Setup.exe file.
2. Follow the installation instructions as they appear.

Table 3 identifies the files that are installed on your hard disk for 64-bit versions of the supported operating systems.

Table 3. DigitalPersona Pro SDK: .NET Edition RTE installed files and folders, 64-bit installation

Folder	File	Description
<destination folder>\Bin	DPCOper2.dll DPDevice2.dll DPDevTS.dll DpHostW.exe DPMux.dll DpSvInfo2.dll DPTSCInt.dll DPCrStor.dll	DLLs and executable file used by the DigitalPersona Pro SDK APIs
<destination folder>\Bin\x64	DPmsg.dll	DLL used by the all of the DigitalPersona Pro SDK APIs
GlobalAssemblyCache	DPPROShrNET.dll DPPRODevNET.dll DPPROEngNET.dll DPPROGuiNET.dll	DLLs used by the all of the DigitalPersona Pro SDK APIs
<SystemFolder> (x86)	DpClback.dll DPCOper2.dll DPCrStor.dll DPDevice2.dll DPFPApi.dll DpHostW.exe DPmsg.dll DPMux.dll dpHFtrEx.dll dpHMatch.dll DpSDAapi.dll	32-bit DLLs used by both of the DigitalPersona Pro SDK APIs
<System64Folder>	DpClback.dll DPFPApi.dll dpHFtrEx.dll dpHMatch.dll DPSDAapi.dll	64-bit DLLs used by both of the DigitalPersona Pro APIs

Table 3. DigitalPersona Pro SDK: .NET Edition RTE installed files and folders, 64-bit installation (*continued*)

Folder	File	Description
<Program Files (x86)>\DigitalPersona\Bin	Bsapi.dll DpBranding.dll DPDB.dll DPDeviceAfss.cat DPDeviceAfss.dll DPDeviceAuthentec.dll DPDeviceUpekBs.dll DPDeviceValidity.dll DpFnd2.dll DPILPro.dll KfsLib.dll DpSvInfo2.dll DpFailureScan.wav DpSuccessScan.wav DPDevTS.dll DPSvInfo2.dll	
<INSTALLDIR>\DigitalPersona\Bin	DpBranding.dll DpFnd2.dll DPTSCIn.dll	
<Citrix Folder>	DPICACnt.dll	

Installing and Uninstalling the RTE Silently

The DigitalPersona Pro SDK: .NET Edition software package contains a batch file, `InstallOnly.bat`, that you can use to silently install the RTE. In addition, you can modify the file to selectively install the various features of the RTE. Refer to the file for instructions.

The SDK software package also contains a file, `UninstallOnly.bat`, that you can use to silently uninstall the RTE.

This chapter introduces DigitalPersona Pro SDK: .NET Edition terminology and concepts; shows how data flows among the DigitalPersona Pro Workstation and Server, the fingerprint-enabled application, and Microsoft Active Directory; and includes typical workflow diagrams and explanations of the One Touch for Windows: .NET Edition API functions used to perform the tasks in the workflows.

Pro SDK Terminology and Concepts

This section defines the DigitalPersona Pro SDK: .NET Edition terminology and concepts that are used throughout this guide.

Old and New Terms

This section is for those who are already familiar with the terminology and concepts presented in the *DigitalPersona Pro for Active Directory Administrator Guide* and the *DigitalPersona Pro SDK: C++ Edition*. Some of the terms used in this guide, although similar in meaning to the terms used in the administrator guide, have been changed. A comparative glossary is included beginning on *page 60*. In addition, old terms are cross-referenced to new terms in the glossary.

Operations

Each time a user touches a fingerprint reader with a finger and the fingerprint reader successfully scans the fingerprint, the reader sends a fingerprint image to the Local Biometric Authentication Service (BAS). The Local BAS then dispatches this message to the fingerprint-enabled application (FEA). Only one FEA can obtain the results of a single fingerprint scan. (The *fingerprint-enabled application* is the application that you develop using Pro SDK API functions.)

When the FEA is required to perform an action in response to a successful fingerprint scan, it should begin by creating one of the following *operations*:

1. Fingerprint capture operation

A *fingerprint capture operation* is created for the purpose of capturing a *supplied fingerprint credential* to be used for

- Performing *fingerprint verification*, which is the process of comparing a supplied fingerprint credential with a stored fingerprint credential based on the user name and a fingerprint provided by the user. If the two fingerprint credentials match, the operation returns the DigitalPersona Pro Secret or a message confirming that the fingerprint is from the user.
- Performing *fingerprint identification*, which is the process of comparing a supplied fingerprint credential with one or more enrollment fingerprint credentials based on a fingerprint provided by the user. If a matching stored fingerprint credential is found, the operation returns the user name.

- Managing user data, which includes retrieving information about a user.
- Managing secrets, which includes writing or deleting secret content from a user record.

Priorities

During the creation of a fingerprint capture operation, the FEA specifies a priority level, which can be low, normal, or high. It is possible to create and subscribe to any number of fingerprint acquisition operations with normal priority, but only one operation for each of low and high priority can be created and subscribed to at a time.

When the Local BAS is ready to dispatch the results of the fingerprint scan, it processes operations using the following hierarchy:

- If there is a high-priority operation subscribed to, the result is dispatched to the process that owns the operation.
- If there is no high-priority operation subscribed to, the Local BAS determines which process owns the topmost window. If there is a normal-priority operation owned by this process, it receives the result.
- If the first two steps do not allow the Local BAS to dispatch the result, the process owning the low-priority operation (if subscribed to) receives the result.
- If the result is still not dispatched, it is discarded.

User Interface Support

The Pro SDK offers two .NET controls to aid in creating a biometrics-enabled GUI. These components address fingerprint authentication and identification by simplifying their respective functionalities and employing a customizable user-friendly interface.

Fingerprint Verification/Authentication

Fingerprint verification/authentication is handled by the `DPPRO.VerifyUser` control. For a detailed description with images, see page 48. For methods, events and properties, see page 38.

Fingerprint Identification

Fingerprint identification is handled by the `DPPRO.IdentifyUser` control. For a detailed description with images, see page 50. For methods, events and properties, see page 36.

Pro SDK Data Flow

Figure 1 illustrates how data flows among the DigitalPersona Pro Workstation and Server, the FEA, and Active Directory. The figure is followed by the steps in the typical Pro SDK authentication operation, which is remote fingerprint authentication. During fingerprint identification or when fingerprint authentication takes place on a local machine, data is passed within the DigitalPersona Pro Workstation framework only.

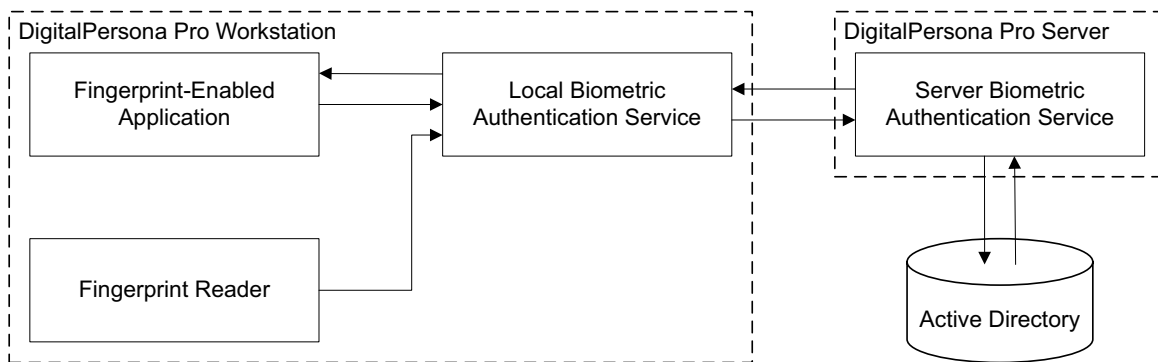


Figure 1. Typical Pro SDK fingerprint authentication operation

IMPORTANT: To perform fingerprint authentication without first performing fingerprint identification, a user name is required. You must build the functionality for acquiring the user name into your application, as it is not part of the Pro API.

1. The FEA initiates an authentication operation and invites the user to provide a fingerprint by touching the fingerprint reader.
2. The user touches the fingerprint reader, and the fingerprint reader sends a fingerprint image to the Local BAS.
3. The Local BAS receives the fingerprint image, creates a supplied fingerprint credential from the fingerprint image, and sends the handle to the supplied fingerprint credential to the FEA.
4. The FEA receives the handle and sends a request to the Local BAS for the DigitalPersona Pro Secret. This request includes the user name and the handle to the supplied fingerprint credential.
5. The Local BAS receives the request from the FEA, generates a new request for the DigitalPersona Pro Secret, and sends it to the Server BAS. This request includes the user name, the supplied fingerprint credential, and the name of the requested DigitalPersona Pro Secret.
6. The Server BAS receives the request from the Local BAS and sends a request for a stored fingerprint credential to Active Directory.
7. Active Directory returns the stored fingerprint credential.
8. The Server BAS compares the supplied fingerprint credential from the Local BAS with the stored fingerprint credential from Active Directory. If the two match, the Server BAS sends a request for the DigitalPersona Pro Secret to Active Directory.
9. Active Directory returns the requested DigitalPersona Pro Secret to the Server BAS.
10. The Server BAS releases the DigitalPersona Pro Secret to the Local BAS.
11. The Local BAS returns the DigitalPersona Pro Secret to the FEA.

Operation Workflows

Typical operation workflows are presented in this section with illustrations and explanations of the API components used to perform the tasks in each of the following workflows:

- Fingerprint authentication

This subsection contains two workflows: one without UI support and one with UI support.

- Fingerprint identification

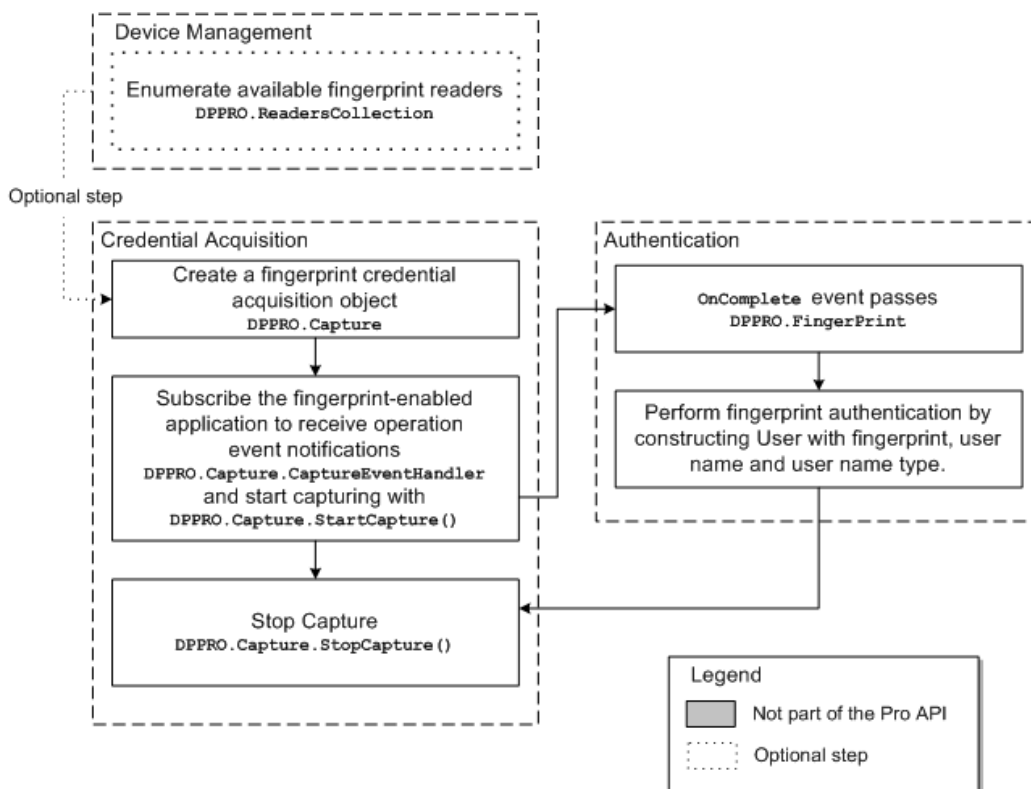
This subsection contains two workflows: one without UI support and one with UI support.

Fingerprint Acquisition Operation Workflows

This section contains workflows for the fingerprint acquisition operation: two for performing fingerprint authentication—with and without UI support—and two for performing fingerprint identification with fingerprint authentication—with and without UI support.

Performing Fingerprint Authentication without UI Support

The typical fingerprint authentication workflow without UI support is illustrated below and is followed by explanations of the Pro API functions used to perform the tasks in the workflow.



Device Management

Enumerate the available fingerprint readers (devices) connected to a computer by constructing `DPPRO.ReadersCollection`. Iterate through the collection and construct `ReaderDescription` objects through one of the collection's indexers.

Credential Acquisition

IMPORTANT: To perform fingerprint authentication without first performing fingerprint identification, a user name is required. You must build the functionality for acquiring the user name into your application, as it is not part of the Pro API.

1. Create a fingerprint acquisition object by constructing `DPPRO.Capture` (*page 25*).
2. Subscribe `DPPRO.Capture` component to receive fingerprint capture operation event notifications by loading a `DPPRO.Capture.CaptureEventHandler` into the `EventHandler` property of `DPPRO.Capture`. Then invoke `DPPRO.Capture.StartCapture()`.
3. Acquire a fingerprint credential from the fingerprint reader.

NOTE: This functionality is provided by the fingerprint reader and is not part of the Pro API.

A supplied fingerprint credential is created. `DPPRO.FingerPrint` can then be passed on construction to a `DPPRO.User` object (*page 44*).

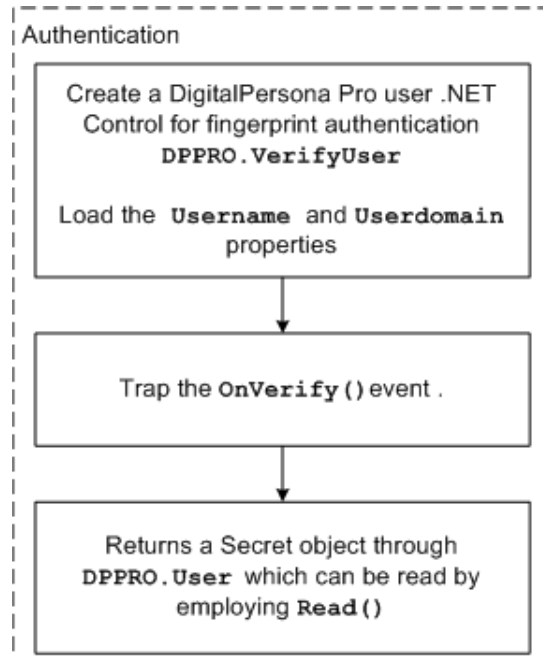
4. Stop receiving fingerprint acquisition operation event notifications by calling `DPPRO.Capture.StopCapture`.

Authentication

1. Perform fingerprint authentication and return a `DigitalPersona Pro Secret` through `DPPRO.User.Secret()` (*page 44*). Then read the secret's content by employing `DPPRO.Secret.Read()`.

Performing Fingerprint Authentication with UI Support

The typical fingerprint authentication workflow with UI support is illustrated below, and is followed by explanations of the Pro API functions used to perform the tasks in the workflow.



Authentication

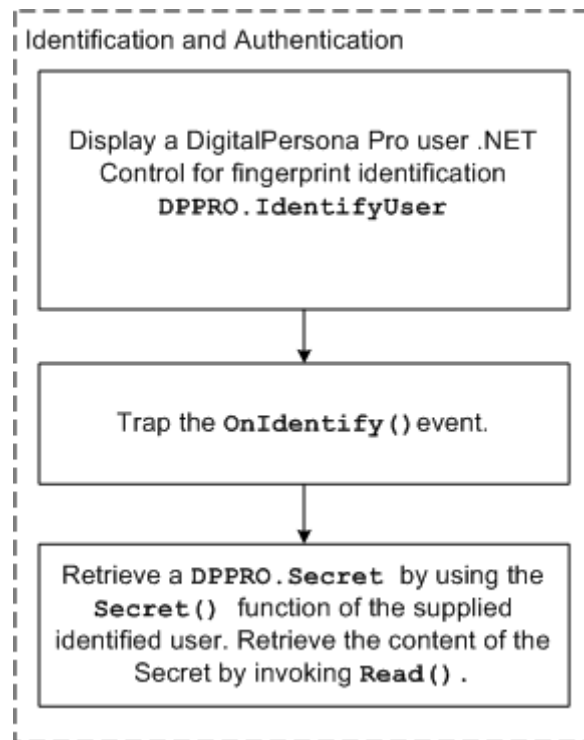
1. Display a DigitalPersona Pro **VerifyUser** Control for fingerprint authentication (*page 38*) and load the `Username` and `Userdomain` properties. Explicitly assigning null to `Username` will automatically set the `Username` to the currently logged in user.
2. Trap the `OnVerify ()` event. Upon verification, the authenticated `DPPRO.User` object will be passed.
3. Return the DigitalPersona Pro Secret by retrieving a `DPPRO.Secret` object from `DPPRO.User` (*page 44*). Read the content of the Secret through `Read ()` (*page 44*).

Performing Fingerprint Identification with Authentication without UI Support

The typical fingerprint identification with fingerprint authentication workflow without UI support is identical to the previous workflow described with UI support with the exception that `DPPRO.User` is constructed with a fingerprint only. Name and NameType are not required. (page 46)

Performing Fingerprint Identification with Authentication with UI Support

The typical fingerprint identification with authentication workflow with UI support is illustrated below and is followed by explanations of the Pro API functions used to perform the tasks in the workflow.



Identification and Authentication

1. Display the DigitalPersona Pro .NET Control for fingerprint identification, `DPPRO.IdentifyUser` (page 36).
2. Trap the `OnIdentify()` event. On successful identification, a `DPPRO.User` will be passed (page 44).
3. Retrieve a `DPPRO.Secret` through the identified user's `Secret()` function, and read the Secret's content by invoking `Read()`.

Components

The DigitalPersona Pro SDK: .NET Edition API includes the components defined in the remainder of this chapter. Use the following list to quickly locate a component by name, by page number, or by description.

Component	Page	Description
Capture component	25	This component provides the ability to <ul style="list-style-type: none"> ■ Capture fingerprint credentials from fingerprint readers ■ Pass events from fingerprint readers ■ Provide information about the fingerprint readers connected to a system
Error Components	34	Provides error management through SDK-specific exceptions and error codes.
GUI Controls	36	This component provides graphical user interfaces for performing fingerprint identification and authentication and event handler status feedback.
Identification & Authentication	39	These components provides for fingerprint authentication and identification, as well as secret management.

Exceptions

The DigitalPersona Pro SDK: .NET Edition extensively employs the .NET exception mechanism to signify a failed operation. Since the SDK sometimes returns a return code/value on function invocation, it is strongly recommended that you use effective exception handling when calling the API.

All SDK-specific exceptions are packaged within the **DPPRO.Error.SDKException** class and should be trapped accordingly. The *Exceptions* column in this API reference list delivery possibilities for each method of each SDK object.

The **DPPRO.Error.SDKException** class extends `ApplicationException` by adding the **ErrorCode** property defined on `Describe`, which returns an enumerated value of **DPPRO.Error.ErrorCodes** (page 34).

The **DPPRO.Error.SDKException** class also provides more information through the following properties:

- **Message**—a string detailing the nature of the exception
- **InnerException**—a `System.Exception` further detailing the nature of the exception

Capture component

Handles capture operations and capture event management. Also allows for reader enumeration.

DPPRO.Capture

Type: Class

Description: Captures a fingerprint credential from a fingerprint reader.

Constructors	Parameters	Exceptions
Capture() Public Initializes Capture for capturing off a specified reader.	String [in] ReaderSerialNum Reader serial number CapturePriority [in] CapturePriority Capture priority	Error.ErrorCodes.NotInitialized SDK Failed to initialize.
Capture() Public Initializes Capture for capturing off a specified reader in normal priority.	String [in] ReaderSerialNum Reader serial number	Error.ErrorCodes.NotInitialized SDK Failed to initialize.
Capture() Public Initializes Capture for capturing off all available readers.	CapturePriority [in] CapturePriority Capture priority	Error.ErrorCodes.NotInitialized SDK Failed to initialize.
Capture() Public Initializes Capture for capturing off all available readers in normal priority.		Error.ErrorCodes.NotInitialized SDK Failed to initialize.

Method	Description	Exceptions
StartCapture() void Public	Initiates a new Capture session; client must invoke StopCapture() for a pending capture before calling StartCapture() again. A Capture event handler must be properly loaded in order to receive events.	Error.ErrorCodes.UnknownDevice Failed to access supplied reader (supplied reader id is incompatible, missing, or mis-typed). Error.ErrorCodes.Internal Failed to create or start an acquisition (wrongly installed SDK libraries or lack of privileges).
StopCapture() void Public	Stops an active capture session.	
Property	Description	
EventHandler() DPPRO.Capture.CaptureEventHandler Public	Loads a capture event handler. Set to null to clear all registered handlers.	
ReaderSerialNumber() string Public	Returns the reader serial number.	
Priority() DPPRO.Capture.CapturePriority Public	Returns the capture priority.	

DPPRO.Capture.CaptureFeedback

Type: Enumeration

Description: Provides feedback about the quality of a capture operation.

Attribute	Description
Good	The fingerprint sample is of decent quality.
None	The fingerprint sample is missing, or was not received.
TooLight	The fingerprint sample is too light.
TooDark	The fingerprint sample is too dark.
TooNoisy	The fingerprint sample is too noisy.
LowContrast	The fingerprint sample contrast is too low.
NotEnoughFeatures	The fingerprint sample does not contain enough information.
NoCentralRegion	The fingerprint sample is not centered.
NoFinger	The scanned object is not a finger.
TooHigh	The finger was too high on the swipe sensor.
TooLow	The finger was too low on the swipe sensor.
TooLeft	The finger was too close to the left border of the swipe sensor.
TooRight	The finger was too close to the right border of the swipe sensor.
TooStrange	The scan looks strange.
TooFast	The finger was swiped too quickly.
TooSkewed	The fingerprint sample is too skewed.
TooShort	The fingerprint sample is too short.
TooSlow	The finger was swiped too slowly.
TooSmall	The size of the fingerprint sample is too small.

DPPRO.Capture.CapturePriority

Type: Enumeration

Description: Designates Capture Priority for the Sample Capture layer

Attribute	Description
Low	Low priority. The subscriber uses this priority to acquire reader events only if there are no subscribers with high or normal priority. Only one subscriber with this priority is allowed.
Normal	Normal priority. The subscriber uses this priority to acquire device events only if the operation runs in a foreground process. Multiple subscribers with this priority are allowed.
High	High priority. The subscriber uses this priority to acquire device events exclusively. Only one subscriber with this priority is allowed.

DPPRO.Capture.CaptureEventHandler

Type: Interface

Description: Capture eventhandler interface

Event	Parameters
OnComplete() void Public	Object [in] Capture Capture component
Fingerprint captured.	String [in] ReaderSerialNumber Reader serial number
	FingerPrint [in] FingerPrintCredential Fingerprint credential retrieved
OnFingerGone() void Public	Object [in] Capture Capture component
Finger removed from reader	String [in] ReaderSerialNumber Reader serial number
OnFingerTouch() void Public	Object [in] Capture Capture component
Finger touched reader	String [in] ReaderSerialNumber Reader serial number

Event	Parameters
OnReaderConnect() void Public Reader connected	Object [in] Capture Capture component String [in] ReaderSerialNumber Reader serial number
OnReaderDisconnect() void Public Reader disconnected	Object [in] Capture Capture component String [in] ReaderSerialNumber Reader serial number
OnSampleQuality() void Public	Object [in] Capture Capture component String [in] ReaderSerialNumber Reader serial number CaptureFeedback [in] SampleQuality

DPPRO.Capture.ReadersCollection

Type: Class - Extends class SortedList<Guid, ReaderDescription>

Description: A list consisting of all available finger print ReaderCollection attached to the system, sorted by GUID/ID.

Constructor	Description	Exception
ReaderCollection() Public	Enumerates all connected fingerprint Readers	Error.ErrorCodes.NotInitialized SDK failed to initialize Error.ErrorCodes.Internal Failed device enumeration

Method	Description	Exceptions
Refresh() void Public	Refreshes ReaderCollection.	Error.ErrorCodes.Internal Failed device enumeration

Indexer	Parameter	Exceptions
this() ReaderDescription Public Returns a specific ReaderDescription by Guid.	Guid [in] ReaderSerialNumber Reader guid	Error.ErrorCodes.UnknownDevice bad GUID (device not found)
this() ReaderDescription Public Returns a specific ReaderDescription by index.	int [in] Index Reader index	Error.ErrorCodes.InvalidParameter index out of range
this() ReaderDescription Public Returns ReaderDescription by serial number.	string [in] ReaderSerialNumber Reader serial number	Error.ErrorCodes.InvalidParameter Invalid serial number

DPPRO.ReaderDescription

Type: Class

Description: Designates a description for a Reader attached to the system and allows for access to ReaderDescription metadata. ReaderDescription is initialized with its Guid/ID number.

Constructor	Parameters	Exceptions
ReaderDescription() Public	Guid [in] DeviceGUID Reader guid	Error.ErrorCodes.NotInitialized SDK failed to initialize
Constructs ReaderDescription with a specified GUID.		Error.ErrorCodes.UnknownDevice Bad supplied GUID
ReaderDescription() Public	String [in] DeviceID Reader serial number	Error.ErrorCodes.NotInitialized SDK failed to initialize
Constructs ReaderDescription with a specified serial number.		Error.ErrorCodes.UnknownDevice Bad supplied GUID
		Error.ErrorCodes.InvalidParameter Invalid serial number

Property	Description
FirmwareVersion() DPPRO.ReaderDescription.ReaderVersion Public	Firmware Version of specified Reader.
HardwareVersion() DPPRO.ReaderDescription.ReaderVersion Public	Hardware Version of specified Reader.
Language() uint Public	Language of specified Reader.
ImpressionType() DPPRO.ReaderDescription.ReaderImpressionType Public	Impression type of specified Reader.
ProductName() String Public	Product name of specified Reader.
SerialNumber() String Public	Serial number of specified Reader.
SerialNumberType() DPPRO.ReaderDescription.ReaderSerialNumberType Public	Serial number type of specified Reader.
Technology() DPPRO.ReaderDescription.ReaderTechnology Public	Technology of specified Reader.

Property	Description
Vendor() String Public	Vendor of specified Reader.

DPPRO.ReaderDescription.ReaderImpressionType

Type: Enumeration
Description: Specifies a ReaderDescription's impression type

Attribute	Description
Unknown	ReaderDescription impression type is unknown.
Swipe	Swipe reader
Area	Area reader

DPPRO.ReaderDescription.ReaderSerialNumberType

Type: Enumeration
Description: Specifies whether a ReaderDescription serial number is provided by hardware (persistent) or software generated (volatile).

Attribute	Description
Persistent	Serial number is provided by hardware (persistent)
Volatile	Serial number is provided by software (volatile)

DPPRO.ReaderDescription.ReaderTechnology

Type: Enumeration
Description: Designates the Reader's underlying technology.

Attribute	Description
Unknown	ReaderDescription technology is unknown
Optical	Optical reader
Capacitive	Capacitive reader
Thermal	Thermal reader
Pressure	Pressure reader

DPPRO.ReaderDescription.ReaderVersion

Type: Class

Description: Designates a version related to a Reader.

Method	Description
ToString() string	Converts ReaderVersion into its string representation

Property	Description
Build() uint	Build number
Major() uint	Major version
Minor() uint	Minor version

Error Components

Provides error management through SDK specific exceptions and error codes.

DPPRO.Error.ErrorCodes

Type: Enumeration
Description: Error codes for SDK Exception

Error	
Success	Operation completed successfully.
NotInitialized	Some Engine components are missing or inaccessible.
InvalidParameter	One or more parameters are not valid.
NotImplemented	Feature is not implemented.
IO	A generic I/O file error occurred.
NotFound	Object not found.
NoCredentials	No credentials left to authenticate.
NoMemory	There is not enough memory to perform the action.
Internal	An unknown internal error occurred.
BadSetting	Initialization settings are corrupted.
UnknownDevice	The requested device is not known.
InvalidBuffer	A buffer is not valid.
FeatureSetTooShort	The specified fingerprint feature set or fingerprint template buffer size is too small.
InvalidContext	The given context is not valid.
InvalidFeatureSetType	The feature set purpose is not valid.
InvalidFeatureSet	Decrypted fingerprint features are not valid. Decryption may have failed.
Unknown	An unknown exception occurred.

SDKException

Type: Class Extends ApplicationException

Description: DPPRO Exception Container

Method	Description	Parameters
SDKException() Public	Initialize with Error code and Error Message.	Exception [in] Exception ErrorCodes [in] ErrorCode Error Code string [in] ErrorMessage Error Message
ErrorCode() ErrorCodes Public	Returns embedded Error Code.	

GUI Controls

Type: Package

Description: This component provides graphical user interfaces for performing fingerprint identification and authentication and event handler status feedback.

DPPRO.InterfaceMode

Type: Enumeration

Description: Control Interface Mode.

Attribute	Description
Hidden	Hidden mode
Minimal	Minimal mode, showing capture indicator
Standard	Standard mode, showing indicator, messages, and user information
Professional	Professional mode

DPPRO.IdentifyUser

Type: Class

Description: A .NET control providing fingerprint Identification and Verification. See details on page page 50.

Property	Description	Default
Mode() Public	Returns an InterfaceMode type. (See above attributes.)	Professional
Active() bool Public	Returns the control capture status.	False
UserName() String Public	Returns the User name to verify. as a string. Note that the default value for this property is null. This will set the currently logged in user name for identification purposes. You may also explicitly set null to UserName to identify the currently logged in user.	null for current user
UserNameType() UserNameType Public	Type of user name to identify (matching the user name set in UserName).	SamCompatible

Event	Description
OnVerify	Fired on successful user identification. A DPPRO.User object is passed back on this event.
OnStatus	Fired on status change. A DPPRO.IdentifyUser.Status (indicating the new status) and a respective message-string is passed back with the event.

DPPRO.IdentifyUser.Status

Type: Enumeration
Description: Control Status.

Attribute	Description
Disconnected	Fingerprint reader is missing or disconnected
Waiting	Waiting for a successful fingerprint capture
Identified	Fingerprint has been identified
FailedToIdentify	Failed to identify fingerprint

DPPRO.VerifyUser

Type: Class

Description: A .NET control providing fingerprint verification. For further details, see page 48.

Property	Description	Default
Mode() Public	Returns an InterfaceMode type.	Professional
Active() bool Public	Returns the control capture status	False
UserName() String Public	Returns the User name to verify, as a string. Note that the default value for this property is null. This will set the currently logged in user name for verification purposes. You may also explicitly set null to UserName to verify the currently logged in user.	null for current user
UserNameType() UserNameType Public	Type of user name to verify (matching the user name set in UserName).	SamCompatible

Event	Description
OnVerify	Fired on successful user verification. A DPPRO.User object is passed back on this event.
OnStatus	Fired on status change. A DPPRO.VerifyUser.Status (indicating the new status) and a respective message-string is passed back with the event.

DPPRO.VerifyUser.Status

Type: Enumeration

Description: Control Status.

Attribute	Description
Disconnected	Fingerprint reader is missing or disconnected
Waiting	Waiting for a successful fingerprint capture
Verified	Fingerprint has been verified
FailedToVerify	Failed to verify fingerprint

Identification & Authentication Component

Description: This component provides for fingerprint authentication and identification, as well as secret management.

DPPRO.Credential

Type: Class

Description: DPPRO credential container.

Constructor	Description	Parameters
Credential() Public	Initialize as a null (empty) credential.	
Credential() Public	Initialize the credential with a specified value.	object [in] Value

Property	Description
Set() bool Public	Checks whether or not a credential has been set.
Value() Object Public	Access or set credential value.

DPPRO.DataLocation

Type: Enumeration

Description: An enumeration detailing secret data location.

Attribute	Description
Unknown	Unknown or unset Location.
Local	Data is Local.
Server	Data is on Server.

DPPRO.FingerPrint

Type: Class Extends Credential
 Description: A DPPRO fingerprint credential.

Constructor	Description	Parameters
Fingerprint() void Public	Initialize FingerPrint with a fingerprint handle.	UInt32 [in] FingerprintHandle

DPPRO.Password

Type: Class Extends Credential
 Description: A DPPRO password credential.

Constructor	Description	Parameters
Password() void Public	Initialize Password with a string.	string [in] PasswordString

DPPRO.Secret

Type: Class
 Description: A DPPRO Secret container. All Secret operations are handled by this object. Retrieve a DPPRO.Secret from a DPPRO.User.Secret() method.

Constructor	Description	Parameters	Exceptions
Secret() Public	Constructs Secret with its owner and designated name.	User [in] User String [in] Name	Error.ErrorCodes.NotInitialized SDK failed to initialize Error.ErrorCodes.InvalidParameter Invalid parameters supplied on Secret construction

Method	Description	Parameters	Exceptions
Delete() void Public	Deletes secret content from a user record.		<p>Error.ErrorCodes.Internal Failed to delete secret. (SDK misconfiguration or lack of sufficient privileges.)</p> <p>Error.ErrorCodes.NotFound Failed to delete secret. (No secret to delete.)</p> <p>Error.ErrorCodes.NoCredentials User authentication failure. (Insufficient credentials to delete secret.)</p>
Exists() boolean Public	Checks whether the Secret exists. Returns true if secret exists, otherwise returns false.		<p>Error.ErrorCodes.Internal Failed to check on secret. (SDK misconfiguration or lack of sufficient privileges.)</p>
Read() boolean Public	Attempts to read the Secret through authentication. Returns True on success, False on failure.	<p>byte[] [out] Data</p> <p>To be filled with Secret content.</p>	<p>Error.ErrorCodes.InvalidParameter User is missing required fingerprint credential.</p> <p>Error.ErrorCodes.Internal Failed to read secret content. (Misconfigured SDK or lack of privileges.)</p> <p>Error.ErrorCodes.NotFound Failed to read secret content. (No such secret available for this user.)</p> <p>Error.ErrorCodes.NoCredentials Missing required credential. (Insufficient/mismatched credentials to retrieve secret content.)</p> <p>Error.ErrorCodes.Unknown User authentication failure.</p>

Method	Description	Parameters	Exceptions
Write() void Public	Attempts to write the Secret with data content in byte array.	byte[] [in] Data	<p>Error.ErrorCodes.InvalidParameter Failed to write Secret. (Data supplied is null or corrupt.)</p> <p>Error.ErrorCodes.Internal Failed to write Secret. (Internal failure, due to misconfiguration and/or access privileges.)</p> <p>Error.ErrorCodes.NoCredentials User authentication failure. (User not allowed to write secret with supplied credentials.)</p>
Write() void Public	Attempts to write the Secret with data content in a string.	string [in] Data	<p>Error.ErrorCodes.InvalidParameter, Failed to write Secret. (String supplied is null or corrupt.)</p> <p>Error.ErrorCodes.Internal Failed to write Secret. (Internal failure, due to misconfiguration and/or access privileges.)</p> <p>Error.ErrorCodes.NoCredentials User authentication failure. (User not allowed to write secret with supplied credentials.)</p>
Write() void Public	Attempts to write the Secret with data content in a stream.	stream [in] Stream	<p>Error.ErrorCodes.InvalidParameter Failed to write Secret. (Stream supplied is null or corrupt.)</p> <p>Error.ErrorCodes.Internal Failed to write Secret. (Internal failure, due to misconfiguration and/or access privileges.)</p> <p>Error.ErrorCodes.NoCredentials User authentication failure. (User not allowed to write secret with supplied credentials.)</p>

Property	Description
Location() DataLocation Public	Returns Secret location.
Owner() User Public	Returns Secret owner.
Name() String Public	Returns Secret name.

DPPRO.User

Type: [Class](#)

Description: By definition, a DPPRO User is required to have some sort of an identification. The four overloaded constructors handle this requirement. Identification is attempted when a FingerPrint is provided with no identifying name; verification is attempted when a name is provided along with a FingerPrint, and the remaining constructors delegate identity management to the calling application.

Constructor	Description	Parameters	
User() Public	Performs fingerprint identification on construction with specified credential.	FingerPrint [in] FingerPrintCredential	Error.ErrorCodes.NotInitialized SDK Failed to initialize. Error.ErrorCodes.Internal User identification failure. (SDK misconfiguration, or lack of sufficient privileges.) Error.ErrorCodes.InvalidParameter User identification failure. (Cannot identify supplied fingerprint.)
User() Public	Performs fingerprint verification on construction with specified credential, user name and user name type.	FingerPrint [in] FingerPrintCredential String [in] Name UserNameType [in] Type	Error.ErrorCodes.NotInitialized SDK Failed to initialize. Error.ErrorCodes.Internal User verification failure. (SDK misconfiguration, or lack of sufficient privileges.) Error.ErrorCodes.InvalidParameter User verification failure. (Cannot verify supplied fingerprint against user identity.)
User() Public	Constructs user with specified SamCompatible name.	String [in] Name	Error.ErrorCodes.InvalidParameter Invalid user name supplied on construction.
User() Public	Constructs user with specified name and type.	String [in] Name UserNameType [in] Type	Error.ErrorCodes.InvalidParameter Invalid user name or type supplied on construction.

Method	Description	Parameters
AddToIdentifyList() boolean Public	Adds the user to the kiosk identification list. Returns True on success, False on failure.	
Secret() Secret Public	Returns a Secret object with specified name.	String [in] Name Secret name.
Property	Description	Exceptions
Credentials() List<Credential> Public	Returns User's list of credentials (excluding fingerprint).	
FingerPrint() FingerPrint Public	Returns the User's fingerprint credential, if available. When setting a new fingerprint credential, invokes User verification (throwing an exception on failure).	Error.ErrorCodes.Internal User verification failure (SDK misconfiguration or insufficient access privileges) Error.ErrorCodes.InvalidParameter User verification failure (Loaded fingerprint credential failed to verify)
get Info() UserInfo Public	Returns User information.	Error.ErrorCodes.InvalidParameter bad or corrupt user information.
get Name() String Public	Returns User name.	
get Type() UserNameType Public	Returns User name type.	

DPPRO.User.UserNameType

Type: Enumeration

Description: User name type, defaults to SamCompatible in most cases.

Attribute	Description
Unknown	A name not associated with any Windows account.
NetBiosDomain	NetBIOS domain name, for example, "THE_COMPANY".
DnsDomain	A DNS domain name, for example, "thecompany.com".
Unknown	A name not associated with any Windows account. This value is to be used for local databases only.
SamCompatible	A Microsoft Windows NT 4.0 account name, for example, "the_company\jdoe".
AccountSimple	The account name format used in Microsoft(r) Windows NT(r) 4.0, for example, "jdoe".
UniqueID	A GUID string returned by the IIDFromString function, for example, "{4fa050f0-f561-11cf-bdd9-00aa003a77b6}".
UserPrincipal	A user principal name, for example, "jdoe@thecompany.com".
Display	A friendly display name, for example, "John Doe".
SID	A user SID string, for example, "S-1-5-21-1004".

DPPRO.User.UserInfo

Type: Class

Description: User information.

Property	Description
ID() Guid Public	User's GUID.
Flags() long Public	User's account control bits.
EnrolledFingerMask() long Public	A bit mask combination representing the user's enrolled fingers.
Location() DataLocation Public	Information location.
MaxEnrollFingerCount() long Public	Maximum number of fingerprints that can be enrolled.

This chapter describes the graphical user interfaces available through the following controls.

- **DPPRO.VerifyUser**

This control includes the graphical user interface described below. Available methods are given on page 38 and following.

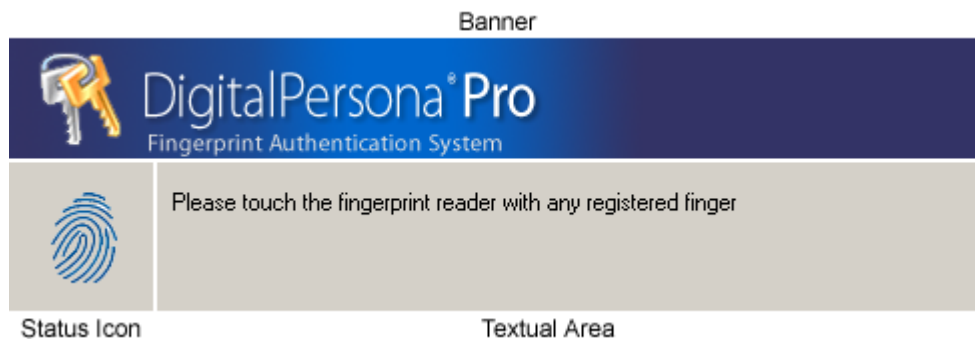
- **DPPRO.IdentifyUser**

This control includes the graphical user interface described beginning on page low. Available methods are given on page 36 and following.

DPPRO.VerifyUser Graphical User Interface

The graphical user interface included with the **DPPRO.VerifyUser** control can be used to verify a user's fingerprint.

The visual element is composed of three parts:



The control has four properties:

Property	Description	Default
Active	Defines whether or not the control is actively awaiting input.	False
UserName	Name of the user whose fingerprint is to be verified. If null, uses currently logged in user.	Null
UsernameType	Type of UserName, as defined in DPPRO.User.UserNameType	SamCompatible

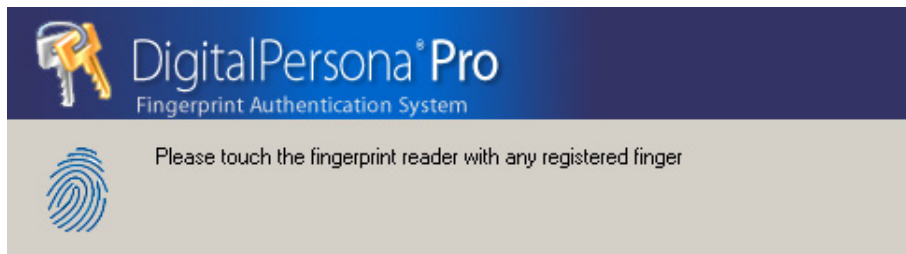
Property	Description	Default
Mode	Defines which parts of the control to display. Professional - Banner, Status Icon and Textual Area Standard - Status Icon and Textual Area Minimal - Status Icon only Hidden - No visual element is displayed	Professional

The control has two events:

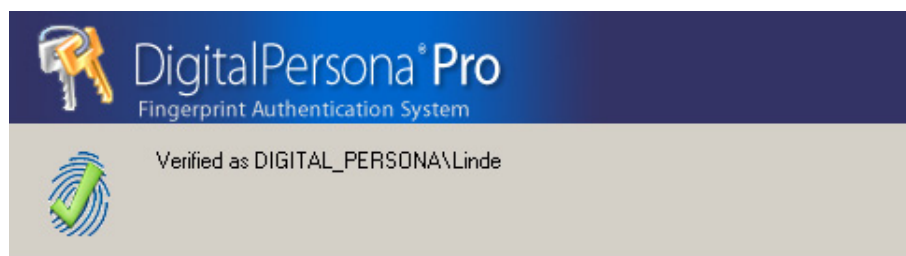
Event	Description
OnVerify(Object Control, User User)	Fired on successful user verification.
OnStatus(Object Control, Status Status, String Message)	Fired on status change.

The Status Icon has four statuses, with associated text. These directly relate to VerifyUser.Status.

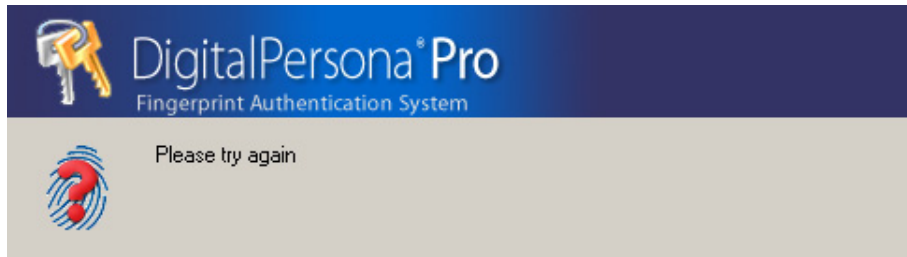
Waiting for input



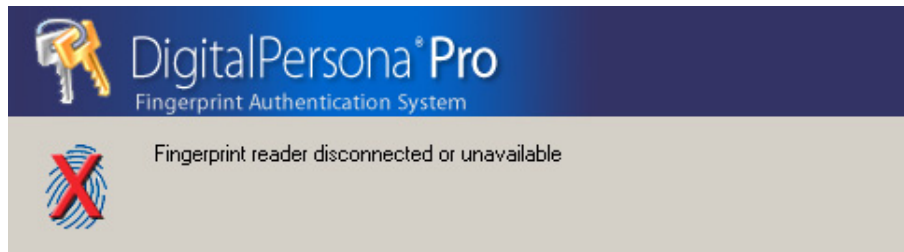
Verified



Failed to verify



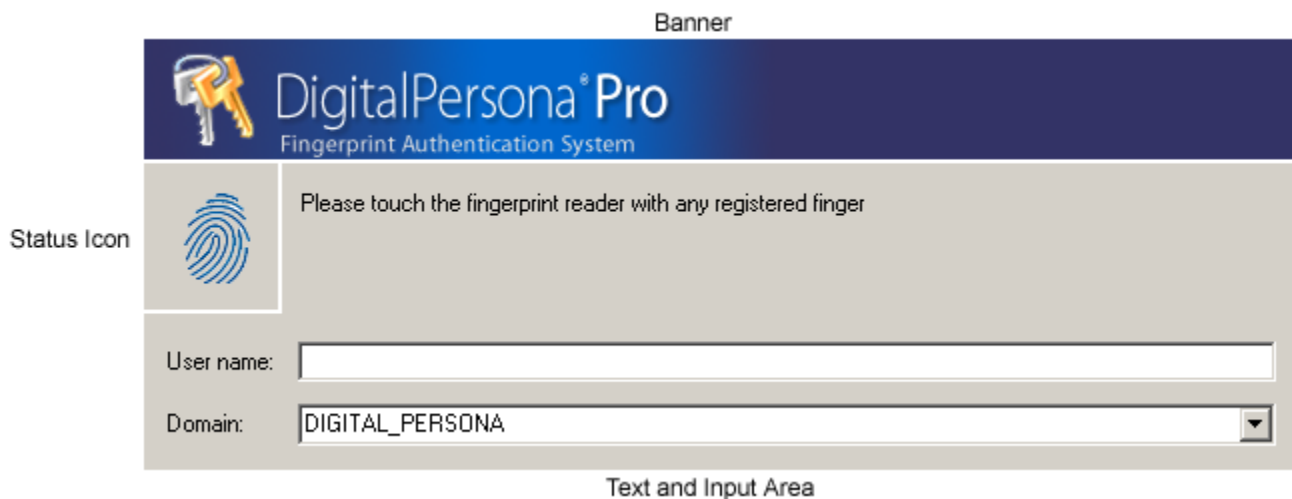
Reader Unavailable



DPPRO.IdentifyUser Graphical User Interface

The graphical user interface included with the `DPPRO.IdentifyUser` control can be used to identify a user by comparing their captured fingerprint to their previously enrolled fingerprint.

The visual element is composed of three parts:

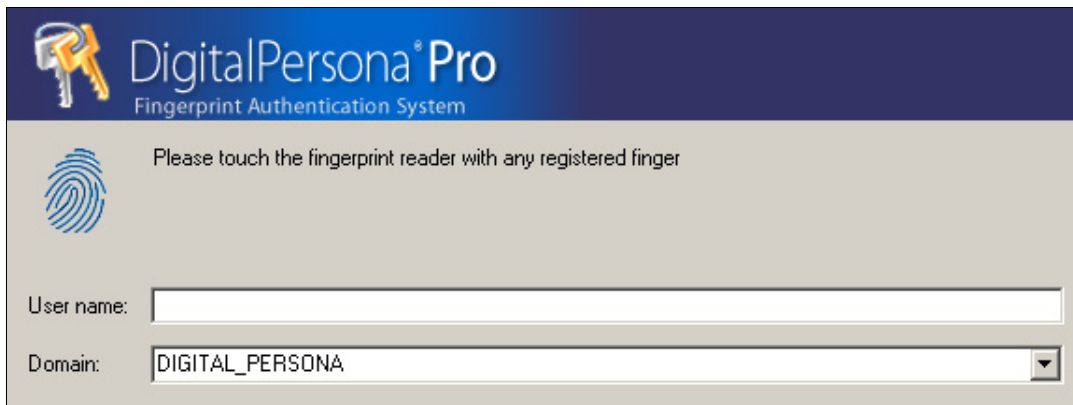


The control has four properties:

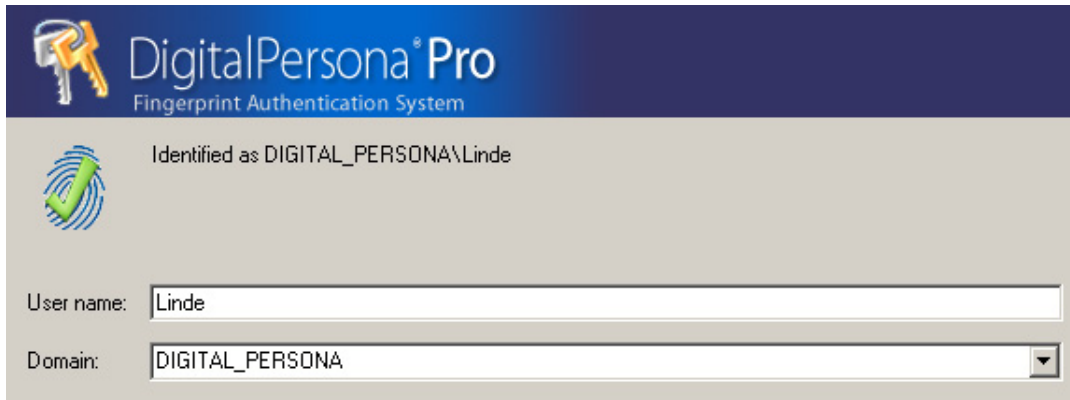
Name	Description	Default
Active	Defines whether or not the control is actively awaiting input.	False
Mode	Defines which parts of the control to display: Professional - Banner, Status Icon, Text and Input Area Standard - Status Icon, Text and Input Area Minimal - Status Icon only Hidden - No visual element is displayed	Professional
Userdomain	Name of the domain that the UserName belongs to. If null, uses currently logged in domain.	Null
Username	Name of the user whose fingerprint is to be verified. If null, uses currently logged in user.	Null

The Status Icon has four statuses, with associated text.

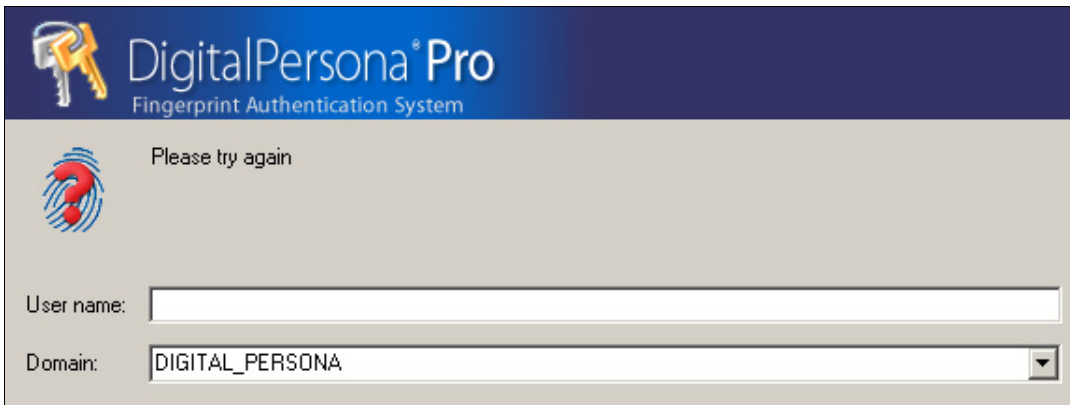
Waiting for input



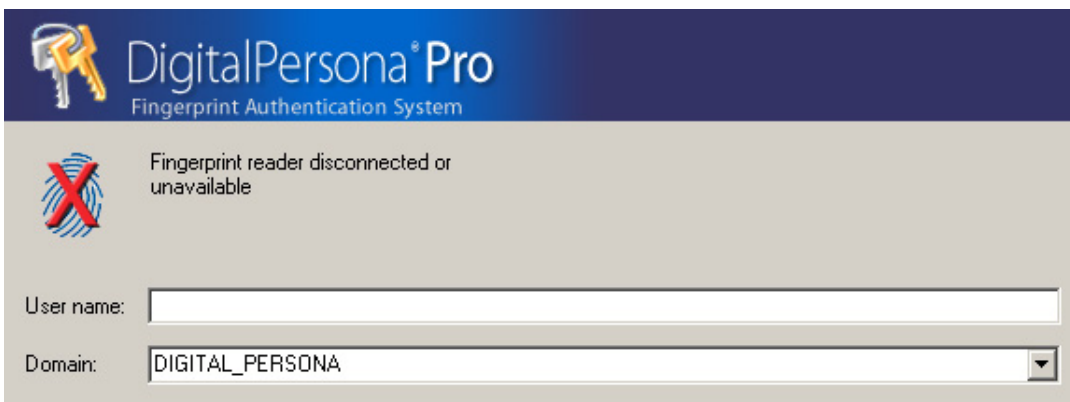
User Identified



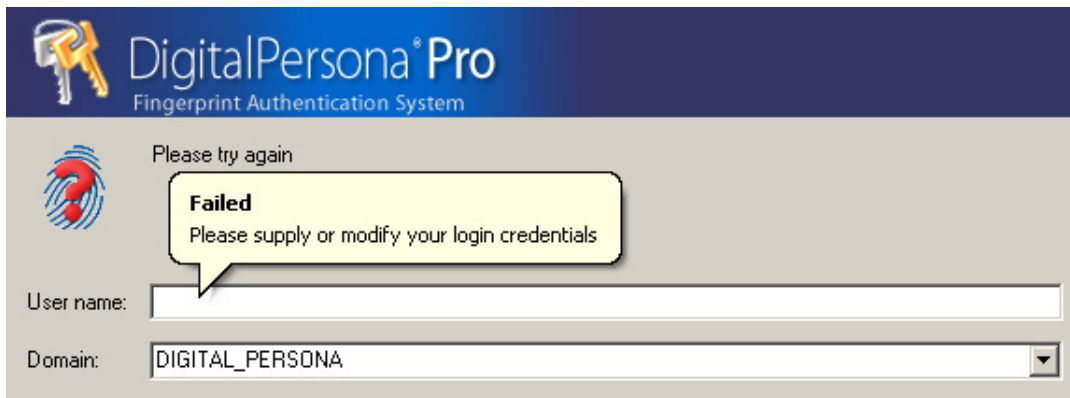
User failed to identify



Reader Unavailable



If the user identification process fails three times in a row, the following balloon message is displayed.



This SDK includes support for fingerprint authentication through Windows Terminal Services (including Remote Desktop Connection) and through a Citrix connection to Metaframe Presentation Server using a client from the Citrix Presentation Server Client package.

The following types of Citrix clients are supported for fingerprint authentication:

- Program Neighborhood
- Program Neighborhood Agent
- Web Client

In order to utilize this support, your application (or the end-user) will need to copy a file to the client computer and register it. The name of the file is DPICACnt.dll, and it is located in the "Misc\Citrix Support" folder in the product package.

To deploy the DP library for Citrix support:

1. - Locate the DPICACnt.dll file on the DP Product CD in the "Misc\Citrix Support" folder.
2. Copy the file to the folder on the client computer where the Citrix client components are located (i.e. for the Program Neighborhood client it might be the "Program Files\Citrix\ICA Client" folder).
3. Using the regsvr32.exe program, register the DPICACnt.dll library.

If you have several Citrix clients installed on a computer, deploy the DPICACnt.dll library to the Citrix client folder for each client.

If your application will also be working with Pro Workstation 4.2.0 and later or Pro Kiosk 4.2.0 and later, you will need to inform the end-user's administrator that they will need to enable two Group Policy Objects (GPOs), "Use DigitalPersona Pro Server for authentication" and "Allow Fingerprint Data Redirection". For information on how to enable these policies, see the "DigitalPersona Pro for AD Guide.pdf" located on the DigitalPersona Pro Server installation CD.

You may redistribute the files in the RTE\Install and the Redist folders in the DigitalPersona Pro SDK software package to your end users pursuant to the terms of the end user license agreement (EULA), attendant to the software and located in the Docs folder in the SDK software package.

When you develop a product based on the DigitalPersona Pro SDK, you need to provide the redistributables to your end users. These files are designed and licensed for use with your application. You may include the installation files located in the RTE\Install folder in your application, or you may incorporate the redistributables directly into your installer. You may also use the merge modules located in the Redist folder in the SDK software package to create your own MSI installer.

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 RTE\Install and Redist folders in the DigitalPersona Pro SDK software package to your end users and to incorporate these files into derivative works for sale and distribution:

RTE\Install Folder

- InstallOnly.bat
- Setup.exe
- Setup.msi
- UninstallOnly.bat

Redist Folder

- DpCore.msm

This merge module contains the following files:

- DPCOper2.dll
- DPDevice2.dll
- DPFPApi.dll
- DpHostW.exe
- DPMux.dll
- DPmsg.dll
- DpClback.dll
- DPCrStor.dll

- DpCore_x64.msm

This merge module contains the following files:

- DPFPAapi.dll
- DPCLback.dll

- DpProCore.msm

This merge module contains the following files:

- DPDevTS.dll
- DPSvInfo2.dll
- DPTSCInt.dll

- DpProCore_x64.msm

This merge module contains the following files:

- DPDevTS.dll
- DPSvInfo2.dll
- DPTSCInt.dll

- DpDrivers.msm

This merge module contains the following files:

- dpd00701.dll
- dpd00701x64.dll
- dpd01001.dll
- dpd01001x64.dll
- dpDevCtl.dll
- dpDevCtlx64.dll
- dpDevDat.dll
- dpDevDatx64.dll
- dPersona.cat
- dPersona.inf
- dpersona_x64.cat
- dPersona_x64.inf
- dpi00701.dll
- dpi00701x64.dll
- dpi01001.dll
- dpi01001x64.dll

- DPlnst32.exe
- DPlnst64.exe
- dpK00701.sys
- dpk00701.sys
- dpk01001.sys
- UsbdpFP.sys
- usbdpfp.sys
- DpFpRec.msm
 - This merge module contains the following files:
 - DpHFtrEx.dll
 - DpHMatch.dll
- DpFpRec_x64.msm
 - This merge module contains the following files:
 - <system folder> (x86)\DpHFtrEx.dll
 - <system folder> (x86)\DpHMatch.dll
 - <system64 folder>\DpHFtrEx.dll
 - <system64 folder>\DpHMatch.dll
- DpProSDKRTE.msm
 - This merge module contains the following file:
 - Bsapi.dll
 - DpBranding.dll
 - DPPS2.dll
 - DPDeviceAfss.cat
 - DPDeviceAfss.dll
 - DPDeviceAuthentec.dll
 - DPDeviceUpekBs.dll
 - DPDeviceValidity.dll
 - DpFnd2.dll
 - DPILPro.dll
 - KfsLib.dll
 - DpFailureScan.wav

- DpSuccessScan.wav
- DPSDApi.dll
- DPICACnt.dll
- DpProSDKRTE_x64.msm

This merge module contains the following file:

- Bsapi.dll
- DpBranding.dll
- DPPS2.dll
- DPDeviceAfss.cat
- DPDeviceAfss.dll
- DPDeviceAuthentec.dll
- DPDeviceUpekBs.dll
- DPDeviceValidity.dll
- DpFnd2.dll
- DPILPro.dll
- KfsLib.dll
- DpFailureScan.wav
- DpSuccessScan.wav
- DpBranding.dll
- DPSDApi.dll
- DPICACnt.dll
- DpProSDKDotNET.msm

This merge module contains the following file:

- DPPROShrNET.dll
- DPPRODevNET.dll
- DPPROEngNET.dll
- DPPROGuiNET.dll
- DpDatabase.msm

This merge module contains the following file:

- DPDB.dll

The following table indicates which merge modules should be used for a specific language and platform.

Merge Module	C/C++		.NET	
	32bit	64bit	32bit	64bit
DpDrivers.msm	x	x	x	x
DpPolicies_OTW.msm	x	x	x	x
DPDatabase.msm	x	x	x	x
DpCore.msm	x	x	x	x
DpCore_x64.msm		x		x
DpProCore.msm	x		x	
DpProCore_x64.msm		x		x
DpFpRec.msm	X		x	
DpFpRec_x64.msm		x		x
DpProSDKRTE.msm	x		x	
DpProSDKRTE_x64.msm		x		x
DpProSDKDotNET.msm			x	x

Fingerprint Reader Documentation

You may redistribute the documentation included in the Redist folder in the DigitalPersona Pro SDK software package to your end users pursuant to the terms of this section and of the EULA, attendant to the software and located in the Docs folder in the SDK software package.

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 folder in the DigitalPersona Pro SDK software package. 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 guides, DigitalPersona Reader Maintenance Touch.pdf and DigitalPersona Reader Maintenance Swipe.pdf, are located in the Redist folder in the DigitalPersona Pro SDK software package. You may copy and redistribute the DigitalPersona Reader Maintenance Touch.pdf and the DigitalPersona Reader Maintenance Swipe.pdf files, including the copyright and trademark notices, to those who purchase a U.are.U module or fingerprint reader from you.

biometric system

An automatic method of identifying a person based on the person's unique physical and/or behavioral traits, such as a fingerprint or an iris pattern, or a handwritten signature or a voice.

comparison

The estimation, calculation, or measurement of similarity or dissimilarity between fingerprint feature set(s) and fingerprint template(s).

comparison score

The numerical value resulting from a comparison of fingerprint feature set(s) with fingerprint template(s). Comparison scores can be of two types: similarity scores or dissimilarity scores.

DigitalPersona Fingerprint Recognition Engine

A set of mathematical algorithms formalized to determine whether a fingerprint feature set matches a fingerprint template according to a specified security level in terms of the false accept rate (FAR).

enrollee

See **fingerprint data subject**.

enrollment

See **fingerprint enrollment**.

false accept rate (FAR)

The proportion of fingerprint verification transactions by fingerprint data subjects not enrolled in the system where an incorrect decision of match is returned.

false reject rate (FRR)

The proportion of fingerprint verification transactions by fingerprint enrollment subjects against their own fingerprint template(s) where an incorrect decision of non-match is returned.

features

See **fingerprint features**.

fingerprint

An impression of the ridges on the skin of a finger.

fingerprint capture device

A device that collects a signal of a fingerprint data subject's fingerprint characteristics and converts it to a fingerprint sample. A device can be any piece of hardware (and supporting software and firmware). In some systems, converting a signal from fingerprint characteristics to a fingerprint sample may include multiple components such as a camera, photographic paper, printer, digital scanner, or ink and paper.

fingerprint characteristic

Biological finger surface details that can be detected and from which distinguishing and repeatable fingerprint feature set(s) can be extracted for the purpose of fingerprint verification or fingerprint enrollment.

fingerprint data

Either the fingerprint feature set, the fingerprint template, or the fingerprint sample.

fingerprint data storage subsystem

A storage medium where fingerprint templates are stored for reference. Each fingerprint template is associated with a fingerprint enrollment subject. Fingerprint templates can be stored within a fingerprint capture device; on a portable medium such as a smart card; locally, such as on a personal computer or a local server; or in a central database.

fingerprint data subject

A person whose fingerprint sample(s), fingerprint feature set(s), or fingerprint template(s) are present within the fingerprint recognition system at any time. Fingerprint data can be either from a person being recognized or from a fingerprint enrollment subject.

fingerprint enrollment

a. In a fingerprint recognition system, the initial process of collecting fingerprint data from a person by extracting the fingerprint features from the person's fingerprint image for the purpose of enrollment and then storing the resulting data in a template for later comparison.

b. The system function that computes a fingerprint template from a fingerprint feature set(s).

fingerprint feature extraction

The system function that is applied to a fingerprint sample to compute repeatable and distinctive information to be used for fingerprint verification or fingerprint enrollment. The output of the fingerprint feature extraction function is a fingerprint feature set.

fingerprint features

The distinctive and persistent characteristics from the ridges on the skin of a finger. *See also* **fingerprint characteristics**.

fingerprint feature set

The output of a completed fingerprint feature extraction process applied to a fingerprint sample. A fingerprint feature set(s) can be produced for the purpose of fingerprint verification or for the purpose of fingerprint enrollment.

fingerprint image

A digital representation of fingerprint features prior to extraction that are obtained from a fingerprint reader. *See also* **fingerprint sample**.

fingerprint reader

A device that collects data from a person's fingerprint features and converts it to a fingerprint sample.

fingerprint recognition system

A biometric system that uses the distinctive and persistent characteristics from the ridges of a finger, also referred to as *fingerprint features*, to distinguish one finger (or person) from another.

fingerprint sample

The analog or digital representation of fingerprint characteristics prior to fingerprint feature extraction that are obtained from a fingerprint capture device. A fingerprint sample may be raw (as captured), intermediate (after some processing), or processed.

fingerprint template

The output of a completed fingerprint enrollment process that is stored in a fingerprint data storage subsystem. Fingerprint templates are stored for later comparison with a fingerprint feature set(s).

fingerprint verification

a. In a fingerprint recognition system, the process of extracting the fingerprint features from a person's fingerprint image provided for the purpose of verification, comparing the resulting data to the template generated during enrollment, and deciding if the two match.

b. The system function that performs a one-to-one comparison and makes a decision of match or non-match.

match

The decision that the fingerprint feature set(s) and the fingerprint template(s) being compared are from the same fingerprint data subject.

non-match

The decision that the fingerprint feature set(s) and the fingerprint template(s) being compared are not from the same fingerprint data subject.

one-to-one comparison

The process in which recognition fingerprint feature set(s) from one or more fingers of one fingerprint data subject are compared with fingerprint template(s) from one or more fingers of one fingerprint data subject.

repository

See **fingerprint data storage subsystem**.

security level

The target false accept rate for a comparison context. See also **FAR**.

verification

See **fingerprint verification**.

Comparative Glossary

This comparative glossary is for those who are already familiar with the terminology and concepts presented in previous versions of this SDK and the *DigitalPersona Pro for Active Directory Administrator Guide*.

Some of the terms used in those documents have been changed to make fingerprint authentication easier to understand and to conform to current biometrics industry standards. The following table lists the previous terms, the corresponding new terms used in this developer guide, and the definition or new meaning of each new term.

Previous Term	New Term	Definition
biometric authentication	fingerprint authentication	See fingerprint verification.
fingerprint identification	The term is the same, but the meaning is slightly different.	The process of comparing a supplied fingerprint credential with one or more enrollment fingerprint credentials based on a fingerprint provided by the user. If a matching stored fingerprint credential is found, the operation returns the user name.
fingerprint registration	fingerprint enrollment operation	An operation created for the purpose of enrolling a user's fingerprint. When the successful-operation message is received by the fingerprint-enabled application, a stored fingerprint credential is created, added to the user record, and stored for later comparison with a supplied fingerprint credential. The fingerprint enrollment operation can also be used to delete a stored fingerprint credential from a user record.
fingerprint verification	fingerprint authentication	The process of comparing a supplied fingerprint credential with a stored fingerprint credential based on the user name and a fingerprint provided by the user. If the two fingerprint credentials match, the operation returns the DigitalPersona Pro Secret or a message confirming that the fingerprint is from the user.
registration template	stored fingerprint credential	The output of a fingerprint enrollment operation, which is added to the user record and stored for later comparison with a supplied fingerprint credential.
verification template	supplied fingerprint credential	Data acquired by a fingerprint acquisition operation that is used for performing fingerprint authentication and fingerprint identification and for managing user data and secrets.

Index

A

- Active Directory
 - role in typical fingerprint authentication operation 19
- additional resources 4
 - online resources 4
 - related documentation 4
- Allow Fingerprint Data Redirection 54
- API
 - list of components 24
 - See also individual components by name
 - reference 24–??

B

- biometric system
 - defined 60
- bold typeface, uses of 3

C

- Citrix 1
- Citrix Web Client 1
- Citrix, developing for 54
- comparison, defined 60
- compatible fingerprint templates
 - See fingerprint template compatibility
- components of API
 - list of 24
 - See also individual components by name
- concepts used in Pro SDK 4.2 17
- conventions, document
 - See document conventions
- Courier bold typeface, use of 3

D

- data flow
 - for Pro SDK 18
 - for typical fingerprint authentication operation
 - explained 19
 - illustrated 19
- dialog box, displaying
 - for fingerprint authentication 18
 - for fingerprint identification 18
- DigitalPersona Developer Connection Forum, URL to 4
- DigitalPersona Pro for Active Directory 4.2
 - terminology in administrator guide compared with Pro SDK 4.2 17
- DigitalPersona Pro Server, in Pro SDK data flow 18
- DigitalPersona Pro Workstation, in Pro SDK data flow 18
- DigitalPersona products, supported 5

- document conventions 3
- documentation, related 4
- DPFP.Gui.Enrollment graphical user interface 48, 50

E

- enrollee
 - defined 60
- enrollment
 - See fingerprint enrollment
- exceptions, discussion of 24

F

- false accept rate
 - defined 60
- false reject rate
 - defined 60
- FAR
 - See false accept rate
- features
 - See fingerprint features
- files and folders
 - installed for RTE, 32-bit installation 13
 - installed for RTE, 64-bit installation 15
 - installed for SDK 12
- fingerprint
 - defined 60
- fingerprint acquisition operation 17
 - priorities 18
 - workflows 20–??
- fingerprint authentication 17
 - performing
 - with UI support 18
- fingerprint capture device
 - defined 60
 - See fingerprint reader
- fingerprint characteristics, defined 61
- fingerprint data
 - defined 61
- fingerprint data storage subsystem, defined 61
- fingerprint data subject, defined 61
- fingerprint enrollment
 - defined 61
- fingerprint feature extraction
 - defined 61
- fingerprint feature set
 - defined 61
- fingerprint features, defined 61
- fingerprint identification 17

fingerprint image, defined 61
See also fingerprint sample

fingerprint reader
 defined 61
 redistributing documentation for 59
 use and maintenance guides, redistributing 59

fingerprint recognition system
 defined 62

fingerprint recognition, guide to 4

fingerprint sample
 defined 62
See also fingerprint image

fingerprint template
 defined 62

fingerprint template compatibility 5

fingerprint verification
 defined 62

fingerprint-enabled application 17
 role in typical Pro SDK fingerprint authentication
 operation 19

folders and files
 installed for RTE, 32-bit installation 13
 installed for RTE, 64-bit installation 15
 installed for SDK 12

FRR
See false reject rate

G

graphical user interfaces 48

Group Policy Objects 54

H

hardware warnings and regulatory information,
 redistributing 59

I

image
See fingerprint image

important notation, defined 3

important notice
 developer must provide functionality for acquiring
 user name 19, 21

installation 11

installation files for redistributables
 redistributing 55

installing
 RTE 13
 RTE silently 16
 SDK 11

introduction to SDK 6

italics typeface, uses of 3

L

Local Biometric Authentication Service
 in operations 17
 operation-processing hierarchy 18
 role in typical Pro SDK fingerprint authentication
 operation 19

M

managing
 secrets 18
 user data 18

match
 defined 62

merge modules
 contents of 55
 redistributing 55

Metaframe Presentation Server 1

N

naming conventions 3

non-match
 defined 62

notational conventions 3

note notation, defined 3

O

one-to-one comparison
 defined 62

online resources 4

operation 17
 fingerprint acquisition 17
 workflows 20–??

overview
 of SDK 17

P

performing
 fingerprint authentication 17
 with UI support 22
 without UI support 20
 fingerprint identification 17
 fingerprint identification with authentication
 with UI support 23
 without UI support 23

priorities 18

product compatibility
See fingerprint template compatibility

Program Neighborhood 1

Program Neighborhood Agent 1

Q

quick start 6

DigitalPersona Pro SDK: NET Edition | Developer Guide

65

R

Redist folder, redistributing contents of 55
 redistributables, redistributing 55
 redistribution of files 55
 regulatory information, requirement to advise end users
 of 59
 remote authentication 1
 Remote Desktop Connection 1
 requirements, system
 See system requirements
 resources, additional
 See additional resources
 resources, online
 See online resources
 RTE
 installing 13
 installing/uninstalling silently 16
 redistributing 55
 RTE\Install folder, redistributing contents of 55
 runtime environment
 See RTE

S

sample application
 using VB.NET UI Demo 7, 9
 SDK
 concepts and terminology 17
 data flow 18
 illustrated 19
 files and folders installed 12
 installing 11
 quick start 6
 secrets, managing 18
 Server Biometric Authentication Service
 role in typical Pro SDK fingerprint authentication
 operation 19
 silently installing RTE 16
 supplied fingerprint credential
 creating
 in typical fingerprint authentication workflow
 without UI support 21
 in typical Pro SDK fingerprint authentication
 operation 19
 purpose of 17
 supported DigitalPersona products 5
 system requirements 4

T

template compatibility
 See fingerprint template compatibility
 terminology used in Pro SDK 4.2 17

 compared with Pro for Active Directory 4.2
 Administrator Guide 17
 typefaces, uses of 3
 typographical conventions 3

U

UI support 18
 for fingerprint authentication 18
 for fingerprint identification 18
 performing fingerprint authentication 22
 performing fingerprint identification with
 authentication 23
 uninstalling RTE silently 16
 updates for DigitalPersona software products, URL for
 downloading 4
 URL
 DigitalPersona Developer Connection Forum 4
 Updates for DigitalPersona Software Products 4
 use and maintenance guides for fingerprint readers,
 redistributing 59
 Use DigitalPersona Pro Server for authentication 54
 user data, managing 18

V

VB.NET UI Demo sample application
 using 7, 9
 verification
 See fingerprint verification

W

Web site
 DigitalPersona Developer Connection Forum 4
 Updates for DigitalPersona Software Products 4
 Windows Terminal Services 1
 workflow
 for performing fingerprint authentication
 with UI support 22
 without UI support 20
 for performing fingerprint identification with
 authentication
 with UI support 23
 without UI support 23