

**DigitalPersona, Inc.**

# **One Touch<sup>®</sup> for Windows<sup>®</sup> SDK COM/ActiveX<sup>®</sup> Edition**

Version 1.1

## **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, One Touch, and U.are.U 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. Citrix is a trademark of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries. Microsoft, Visual Basic, Visual C++, Visual Studio, Vista, Windows, and Windows Server are registered trademarks of Microsoft Corporation in the United States and other countries. All other trademarks are the property of their respective owners.

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**

Maintenance and technical support are available for this product from DigitalPersona, its partners, and resellers. Once you have paid for maintenance and technical support and agreed to the support terms, you may obtain this support through a variety of mechanisms, including the online form explained in the next paragraph.

The DigitalPersona Web site provides an online technical support form at <http://www.digitalpersona.com/support/enterprise/chooseproduct.php>. Simply describe your issue and include your contact information, and a technical support representative will contact you shortly by email or by phone.

Phone support is available at (877) 378-2740 in the U.S. only.  
Outside the U.S., call +1 650-474-4000.

**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 .....	1
	Chapter Overview .....	1
	Document Conventions .....	2
	Notational Conventions .....	2
	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
	Install the Software .....	6
	Insert the Fingerprint Reader .....	6
	Using the Sample Application .....	6
3	Installation .....	12
	Installing the SDK .....	12
	Installing the Runtime Environment (RTE) .....	13
	Installing and Uninstalling the RTE Silently .....	14
4	Overview .....	15
	Biometric System .....	15
	Fingerprint .....	15
	Fingerprint Recognition .....	16
	Fingerprint Enrollment .....	17
	Fingerprint Verification .....	17
	False Positives and False Negatives .....	18
	Workflows .....	19
	Fingerprint Enrollment Workflow .....	19
	Fingerprint Enrollment with UI Support .....	22
	Fingerprint Verification .....	23
	Fingerprint Verification with UI Support .....	26
	Fingerprint Data Object Serialization/Deserialization .....	28

5	API Reference for Visual Basic Developers .....	30
	Component Objects .....	30
	DPFPCapture .....	32
	StartCapture Method .....	32
	StopCapture Method .....	32
	Priority Property .....	32
	ReaderSerialNumber Property .....	33
	OnComplete Event .....	34
	OnFingerGone Event .....	34
	OnFingerTouch Event .....	34
	OnReaderConnect Event .....	35
	OnReaderDisconnect Event .....	35
	OnSampleQuality Event .....	35
	DPFPData .....	36
	Deserialize Method .....	36
	Serialize Method .....	36
	DPFPEnrollment .....	37
	AddFeatures Method .....	37
	Clear Method .....	37
	FeaturesNeeded Property .....	38
	Template Property .....	38
	TemplateStatus Property .....	38
	DPFPEnrollmentControl .....	39
	EnrolledFingersMask Property .....	39
	MaxEnrollFingerCount Property .....	40
	ReaderSerialNumber Property .....	41
	OnDelete Event .....	42
	OnEnroll Event .....	43
	DPFPEventHandlerStatus .....	43
	Status Property .....	43
	DPFPFeatureExtraction .....	44
	CreateFeatureSet Method .....	44
	FeatureSet Property .....	45
	DPFPFeatureSet .....	45
	DPFPReaderDescription .....	46
	FirmwareRevision Property .....	46
	HardwareRevision Property .....	46
	Language Property .....	46
	ImpressionType Property .....	47
	ProductName Property .....	47

SerialNumber Property .....	47
SerialNumberType Property .....	48
Technology Property .....	48
Vendor Property .....	48
DPFPReadersCollection .....	49
Reader Method .....	49
Count Property .....	49
Item Property .....	50
_NewEnum Property .....	50
DPFPSample .....	51
DPFPSampleConversion .....	51
ConvertToANSI381 Method .....	51
ConvertToPicture Method .....	52
DPFPTemplate .....	52
DPFPVerification .....	52
Verify Method .....	53
FARRequested Property .....	53
DPFPVerificationControl .....	54
ReaderSerialNumber Property .....	54
OnComplete Event .....	55
DPFPVerificationResult .....	56
FARAchieved Property .....	56
Verified Property .....	56
Enumerations .....	57
DPFPCaptureFeedbackEnum Enumeration .....	58
DPFPCapturePriorityEnum Enumeration .....	59
DPFPEventHandlerStatusEnum Enumeration .....	60
DPFPDataPurposeEnum Enumeration .....	61
DPFPReaderImpressionTypeEnum Enumeration .....	62
DPFPReaderTechnologyEnum Enumeration .....	62
DPFPSerialNumberTypeEnum Enumeration .....	63
DPFPTemplateStatusEnum Enumeration .....	64
6 API Reference for C++ Developers .....	65
Interfaces .....	65
IDPFPCapture Interface .....	67
IDPFPCapture::Priority Property .....	67
IDPFPCapture::ReaderSerialNumber Property .....	68
IDPFPCapture::StartCapture Method .....	68
IDPFPCapture::StopCapture Method .....	69

_IDPFPCaptureEvents Interface .....	69
_IDPFPCaptureEvents::OnComplete Event .....	70
_IDPFPCaptureEvents::OnFingerGone Event .....	70
_IDPFPCaptureEvents::OnFingerTouch Event .....	70
_IDPFPCaptureEvents::OnReaderConnect Event .....	71
_IDPFPCaptureEvents::OnReaderDisconnect Event .....	71
_IDPFPCaptureEvents::OnSampleQuality Event .....	71
IDPFPPData Interface .....	72
IDPFPPData::Deserialize Method .....	72
IDPFPPData::Serialize Method .....	72
IDPFPEnrollment Interface .....	73
IDPFPEnrollment::AddFeatures Method .....	73
IDPFPEnrollment::Clear Method .....	74
IDPFPEnrollment::FeaturesNeeded Property .....	74
IDPFPEnrollment::Template Property .....	74
IDPFPEnrollment::TemplateStatus Property .....	75
IDPFPEnrollmentControl Interface .....	76
IDPFPEnrollmentControl::EnrolledFingersMask Property .....	76
IDPFPEnrollmentControl::MaxEnrollFingerCount Property .....	77
IDPFPEnrollmentControl::ReaderSerialNumber Property .....	78
_IDPFPEnrollmentControlEvents Interface .....	79
_IDPFPEnrollmentControlEvents::OnDelete Event .....	79
_IDPFPEnrollmentControlEvents::OnEnroll Event .....	80
IDPFPEventHandlerStatus Interface .....	81
IDPFPEventHandlerStatus::Status Property .....	81
IDPFPPFeatureExtraction Interface .....	82
IDPFPPFeatureExtraction::CreateFeatureSet Method .....	82
IDPFPPFeatureExtraction::FeatureSet Property .....	83
IDPFPPFeatureSet Interface .....	84
IDPFPPReaderDescription Interface .....	84
IDPFPPReaderDescription::FirmwareRevision Property .....	84
IDPFPPReaderDescription::HardwareRevision Property .....	85
IDPFPPReaderDescription::Language Property .....	85
IDPFPPReaderDescription::ImpressionType Property .....	86
IDPFPPReaderDescription::ProductName Property .....	86
IDPFPPReaderDescription::SerialNumber Property .....	86
IDPFPPReaderDescription::SerialNumberType Property .....	87
IDPFPPReaderDescription::Technology Property .....	87
IDPFPPReaderDescription::Vendor Property .....	88

IDPFPPReadersCollection Interface .....	88
IDPFPPReadersCollection::Reader Method .....	89
IDPFPPReadersCollection::Count Property .....	89
IDPFPPReadersCollection::Item Property .....	90
IDPFPPReadersCollection::_NewEnum Property .....	90
IDPFPSample Interface .....	91
IDPFPSampleConversion Interface .....	91
IDPFPSample::ConvertToANSI381 Method .....	92
IDPFPSample::ConvertToPicture Method .....	92
IDPFPTemplate Interface .....	93
IDPFPPVerification Interface .....	93
IDPFPPVerification::FARRequested Property .....	93
IDPFPPVerification::Verify Method .....	94
IDPFPPVerificationControl Interface .....	95
IDPFPPVerificationControl::ReaderSerialNumber Property .....	96
_IDPFPPVerificationControlEvents Interface .....	97
_IDPFPPVerificationControlEvents::OnComplete Event .....	97
IDPFPPVerificationResult Interface .....	97
IDPFPPVerificationResult::FARAchieved Property .....	97
IDPFPPVerificationResult::Verified Property .....	98
Enumerations .....	99
DPFPCaptureFeedbackEnum Enumerated Type .....	100
DPFPCapturePriorityEnum Enumerated Type .....	101
DPFPEventHandlerStatusEnum Enumerated Type .....	102
DPFPDataPurposeEnum Enumerated Type .....	103
DPFPReaderImpressionTypeEnum Enumerated Type .....	104
DPFPReaderTechnologyEnum Enumerated Type .....	104
DPFPSerialNumberTypeEnum Enumerated Type .....	105
DPFPTemplateStatusEnum Enumerated Type .....	106
7 User Interface .....	107
DPFPEnrollmentControl Object User Interface .....	107
Enrolling a Fingerprint .....	107
Deleting a Fingerprint Template .....	114
DPFPVerificationControl Object User Interface .....	116
8 Redistribution .....	117
RTE\Install Folder .....	117
Redist Folder .....	117

Fingerprint Reader Documentation .....	119
Hardware Warnings and Regulatory Information .....	119
Fingerprint Reader Use and Maintenance Guide .....	119
A Setting the False Accept Rate .....	120
False Accept Rate (FAR) .....	120
Representation of Probability .....	120
Requested FAR .....	121
Specifying the FAR in C++ .....	121
Specifying the FAR in Visual Basic .....	122
Achieved FAR .....	122
Testing .....	122
B Platinum SDK Registration Template Conversion .....	123
Platinum SDK Registration Template Conversion for Microsoft Visual C++ Applications .....	123
Platinum SDK Registration Template Conversion for Visual Basic 6.0 Applications .....	125
Glossary .....	126
Index .....	129



The One Touch® for Windows SDK 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 One Touch for Windows SDK 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 One Touch for Windows SDK 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 One Touch for Windows SDK 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 One Touch for Windows SDK product to add fast fingerprint identification capability to a developer's design.

## Target Audience

This guide is for developers who have a working knowledge of the C++ or Visual Basic programming language and the RPC paradigm as it applies to COM, or familiarity with OLE Automation model scripting and type libraries.

## 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 One Touch for Windows SDK: COM/ActiveX Edition; identifies the minimum system requirements needed to run the One Touch for Windows SDK: COM/ActiveX Edition; and lists the DigitalPersona products and fingerprint templates supported by the One Touch for Windows SDK: COM/ActiveX Edition.

Chapter 2, *Quick Start*, provides a quick introduction to the One Touch for Windows SDK: COM/ActiveX Edition using one 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 One Touch for Windows SDK: COM/ActiveX Edition terminology and concepts. This chapter also includes typical workflow diagrams and explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflows.

Chapter 5, *API Reference for Visual Basic Developers*, defines the component objects (including methods, properties, and events) and the enumerations that are used for developing applications based on the One Touch for Windows: COM/ActiveX Edition API in Microsoft® Visual Basic®.

Chapter 6, *API Reference for C++ Developers*, defines the interfaces (including their methods, properties, and events) and the enumerations that are used for developing applications based on the One Touch for Windows: COM/ActiveX Edition API in C++.

Chapter 7, *User Interface*, describes the functionality of the user interfaces included with the DPFPEnrollmentControl and DPFPVerificationControl controls.

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: COM/ActiveX Edition API.

Appendix A, *Setting the False Accept Rate*, provides information about determining and using specific values for the FAR and evaluating and testing achieved values.

Appendix B, *Platinum SDK Registration Template Conversion*, contains sample code for converting Platinum SDK registration templates for use with the One Touch for Windows SDK: COM/ActiveX Edition.

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
<b>Bold</b>	Used for keystrokes and window and dialog box elements and to indicate data types	Click <b>Fingerprint Enrollment</b> . The <b>Fingerprint Enrollment</b> dialog box appears. <b>String</b> that specifies a fingerprint reader serial number
<b>Courier bold</b>	Used to indicate computer programming code	When <code>SampleQualityGood</code> is returned, the <code>OnComplete</code> event is fired. Deserializes a data object returned by the <code>IDPFData::Serialize</code> method.
<i>Italics</i>	Used for emphasis or to introduce new terms If you are viewing this document online, clicking on text in italics may also activate a hypertext link to other areas in this guide or to URLs.	This section includes illustrations of <i>typical</i> fingerprint enrollment and fingerprint verification workflows. (emphasis) <i>A fingerprint</i> is an impression of the ridges on the skin of a finger. (new term) See <i>Installing the SDK</i> on page 8. (link to heading and page)

## Naming Conventions

The *DPFP* prefix used in API methods, properties, data types, and constants stands for *DigitalPersona Fingerprint*, and the *IDFPF* prefix is used for interfaces.

## Additional Resources

You can refer to the resources in this section to assist you in using the One Touch for Windows SDK: COM/ActiveX Edition.

### Related Documentation

Subject	Document
Fingerprint recognition, including the history and basics of fingerprint identification and the advantages of DigitalPersona's Fingerprint Recognition Algorithm	The DigitalPersona White Paper: Guide to Fingerprint Recognition (Fingerprint Guide.pdf located in the Docs folder on the One Touch for Windows SDK product CD)
Late-breaking news about the product	The Readme.txt files provided in the root directory of the product CD as well as in some subdirectories

### Online Resources

Web Site name	URL
DigitalPersona Developer Connection Forum for DigitalPersona Developers	<a href="http://www.digitalpersona.com/webforums/">http://www.digitalpersona.com/webforums/</a>
Latest updates for DigitalPersona software products	<a href="http://www.digitalpersona.com/support/downloads/software.php">http://www.digitalpersona.com/support/downloads/software.php</a>

## System Requirements

This section lists the minimum software and hardware requirements needed to run the One Touch for Windows SDK: COM/ActiveX Edition.

- x86-based processor or better
- CD-/DVD-ROM drive
- Microsoft® Windows® 2000 Professional SP4; Microsoft® Windows® XP Home, Professional, or Embedded<sup>1</sup>; Microsoft® Windows Server® 2003 SP1; or Microsoft® Windows Vista®; only 32-bit versions supported
- USB port on the computer where the fingerprint reader is to be connected
- DigitalPersona U.are.U 4000B Fingerprint Reader

---

1. A list of DLL dependencies for installation of your application on Microsoft Windows XP Embedded, One Touch for Windows XPE Dependencies.xls, is located in the Docs folder on the product CD.

## Supported DigitalPersona Products

The One Touch for Windows SDK: COM/ActiveX Edition supports the following DigitalPersona products:

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

## Fingerprint Template Compatibility

Fingerprint templates produced by the One Touch for Windows 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. See Appendix B on *page 123* for sample code that converts Platinum SDK templates to this format.

This chapter provides a quick introduction to the One Touch for Windows SDK: COM/ActiveX Edition using one of the sample applications provided as part of the One Touch for Windows SDK. This application is a Microsoft® Visual Basic® 6 project that demonstrates the functionality of the user interfaces included in the **DPFPEnrollmentControl** and **DPFPVerificationControl** component objects. The user interfaces are described in more detail in *DPFPEnrollmentControl Object User Interface on page 107* and *DPFPVerificationControl Object User Interface on page 116*.

## Install the Software

Before you can use the sample application, you must install the One Touch for Windows SDK: COM/ActiveX Edition, which includes the runtime environment (RTE).

### To install the One Touch for Windows SDK: COM/ActiveX Edition

1. Insert the One Touch for Windows product CD into your CD/DVD-ROM drive.
2. In the SDK folder, open the Setup.exe file, and then click **Next**.
3. Follow the installation instructions as they appear.
4. Restart your computer.

## Insert the Fingerprint Reader

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

## Using the Sample Application

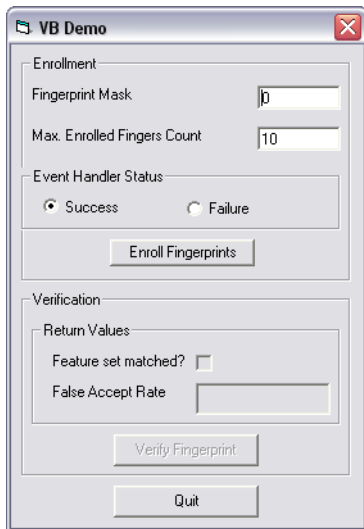
By performing the exercises in this section, you will

- Start the sample application
- Enroll a fingerprint
- Verify a fingerprint
- Unenroll (delete) a fingerprint
- Exit the sample application

## To start the sample application

- Open the UIVBDemo.exe file location in the <destination folder>One Touch SDK\COM-ActiveX\Samples\VB6\UI Support folder.

The **VB Demo** dialog box appears.

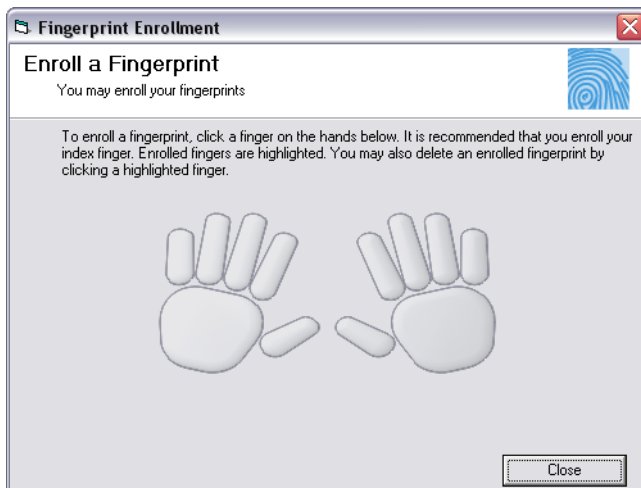


Enrolling a fingerprint consists of scanning your fingerprint four times using the fingerprint reader.

## To enroll a fingerprint

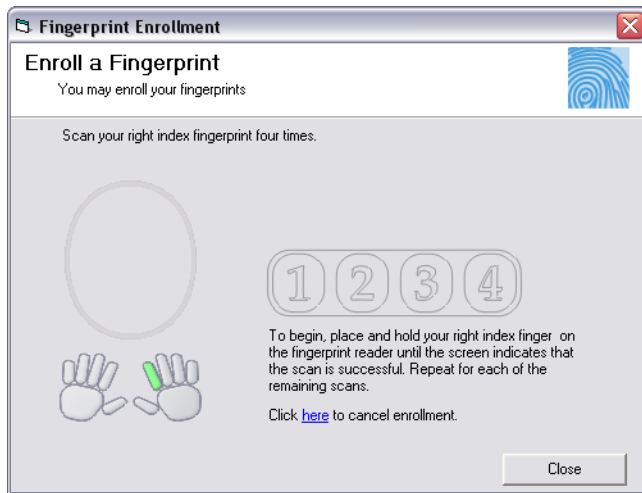
1. In the **VB Demo** dialog box, click **Enroll Fingers**.

The **Fingerprint Enrollment** dialog box appears.

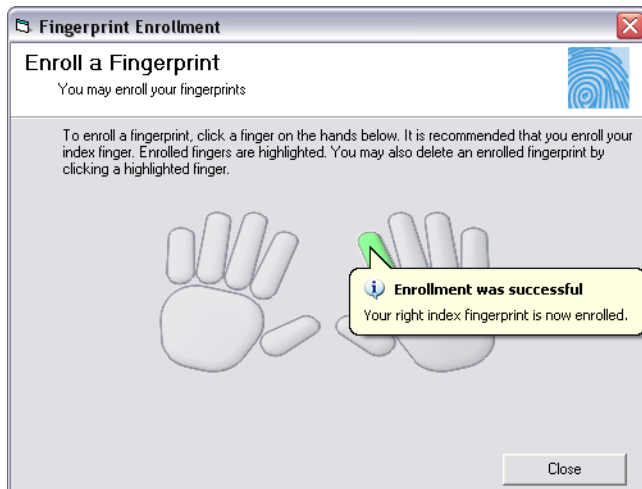


2. In the right “hand,” click the index finger.

A second **Fingerprint Enrollment** dialog box appears.



3. Using the fingerprint reader, scan your right index fingerprint.
4. Repeat step 3 until the **Enrollment was successful** message appears.



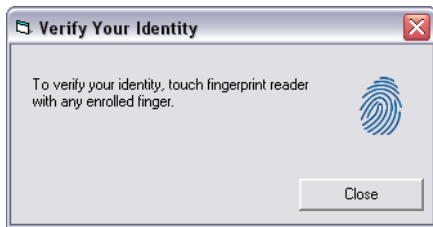
5. Click **Close**.



**To verify a fingerprint**

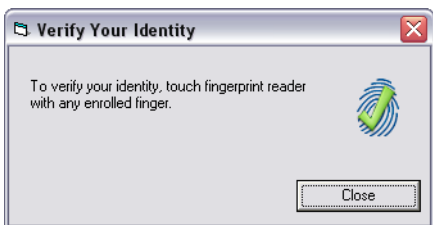
1. In the **VB Demo** dialog box, click **Verify Fingerprint**.

The **Verify Your Identity** dialog box appears.



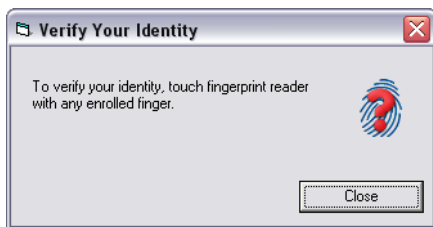
2. Using the fingerprint reader, scan your right index fingerprint.

In the **Verify Your Identity** dialog box, a green check mark appears over the fingerprint, which indicates that your fingerprint was verified.



3. Using the fingerprint reader, scan your right middle fingerprint.

In the **Verify Your Identity** dialog box, a red question mark appears over the fingerprint, which indicates that your fingerprint was not verified.

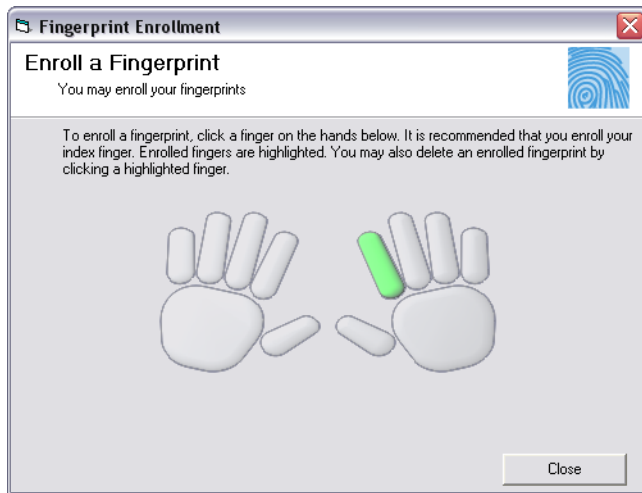


4. Click **Close**.

## To unenroll (delete) a fingerprint

1. In the **VB Demo** dialog box, click **Enroll Fingerprints**.

The **Fingerprint Enrollment** dialog box appears, indicating that you have enrolled your right index fingerprint.



2. On the right "hand," click the green index finger.

A message box appears, asking you to verify the deletion.



3. In the message box, click **Yes**.

The right index finger is no longer green, indicating that the fingerprint associated with that finger is not enrolled, or has been deleted.



**To exit the application**

- In the **VB Demo** dialog box, click **Quit**.

This chapter contains instructions for installing the various components of the One Touch for Windows SDK: COM/ActiveX Edition and identifies the files and folders that are installed on your hard disk.

The following two installations are located on the product CD:

- 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 implement the One Touch for Windows SDK: COM/ActiveX Edition interfaces, objects, methods, and properties. This installation is located in the RTE folder. (The RTE installation is also included in the SDK installation.)

## Installing the SDK

### To install the One Touch for Windows SDK: COM/ActiveX Edition

1. Insert the One Touch for Windows product CD into your CD/DVD-ROM drive.
2. In the SDK folder, open the Setup.exe file, and then click **Next**.
3. Follow the installation instructions as they appear.
4. Restart your computer.

Table 1 describes the files and folders that are installed in the <destination folder> folder on your hard disk. The RTE files and folders, which are listed in Table 2 on page 13, are also installed on your hard disk.

NOTE: All installations share the DLLs and the DPHostW.exe file that are installed with the C/C++ edition. Additional product-specific files are provided for other editions.

**Table 1.** One Touch for Windows SDK: COM/ActiveX Edition installed files and folders

Folder	File	Description
One Touch SDK\COM-ActiveX\Docs	One Touch for Windows SDK COM-ActiveX Developer Guide.pdf	DigitalPersona One Touch for Windows SDK: COM/ActiveX Edition Developer Guide
One Touch SDK\COM-ActiveX\Samples\VB6\Enrollment Sample	This folder contains a sample Microsoft Visual Basic 6 project that shows how to use the One Touch for Windows: COM/ActiveX Edition API for performing fingerprint enrollment and fingerprint verification	
One Touch SDK\COM-ActiveX\Samples\VB6\UI Support	This folder contains a sample Microsoft Visual Basic 6 project that demonstrates the functionality of the user interfaces included in the DPFPEnrollmentControl and DPFPVerificationControl component objects of the One Touch for Windows: COM/ActiveX Edition API	

## Installing the Runtime Environment (RTE)

When you develop a product based on the One Touch for Windows SDK: COM/ActiveX Edition, 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 on the product CD to create your own MSI installer. (See *Redistribution* on page 117 for licensing terms.)

If you created an application based on the One Touch for Windows: COM/ActiveX Edition APIs that does not include an installer, your end users must install the One Touch for Windows: COM/ActiveX 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 One Touch for Windows: COM/ActiveX Edition Runtime Environment

1. Insert the One Touch for Windows product CD into your CD-/DVD-ROM drive.
2. In the RTE folder, open the Setup.exe file.
3. Follow the installation instructions as they appear.

Table 2 identifies the files that are installed on your hard disk.

**Table 2.** One Touch for Windows SDK: COM/ActiveX Edition RTE installed files and folders

Folder	File	Description
<destination folder>\Bin	DPCOper2.dll DPDevice2.dll DPDevTS.dll DpHostW.exe DPmsg.dll DPMux.dll DpSvcInfo2.dll DPTSCInt.dll	DLLs and executable file used by the all of the One Touch for Windows APIs

**Table 2.** One Touch for Windows SDK: COM/ActiveX Edition RTE installed files and folders (*continued*)

Folder	File	Description
<destination folder>\Bin\ COM-ActiveX	DPFPShrX.dll DPFPDevX.dll DPFPEngX.dll DPFPCtlX.dll	DLLs used by the One Touch for Windows: COM/ ActiveX Edition API
<system folder>	DPFPApi.dll DpClback.dll dpHFtrEx.dll dpHMatch.dll DPFPUI.dll	DLLs used by all of the One Touch for Windows SDK APIs

## Installing and Uninstalling the RTE Silently

The One Touch for Windows project CD 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 product CD also contains a file, `UninstallOnly.bat`, that you can use to silently uninstall the RTE.

This chapter introduces One Touch for Windows SDK: COM/ActiveX Edition concepts and terminology. (For more details on the subject of fingerprint biometrics, refer to the “DigitalPersona White Paper: Guide to Fingerprint Recognition” included on the One Touch for Windows product CD.) This chapter also includes typical workflow diagrams and explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflows.

## Biometric System

A *biometric system* is 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 voice. Biometric identifiers are

- Universal
- Distinctive
- Persistent (sufficiently unchangeable over time)
- Collectable

Biometric systems have become an essential component of effective person recognition solutions because biometric identifiers cannot be shared or misplaced and they naturally represent an individual’s bodily identity. Substitute forms of identity, such as passwords (commonly used in logical access control) and identity cards (frequently used for physical access control), do not provide this level of authentication that strongly validates the link to the actual authorized user.

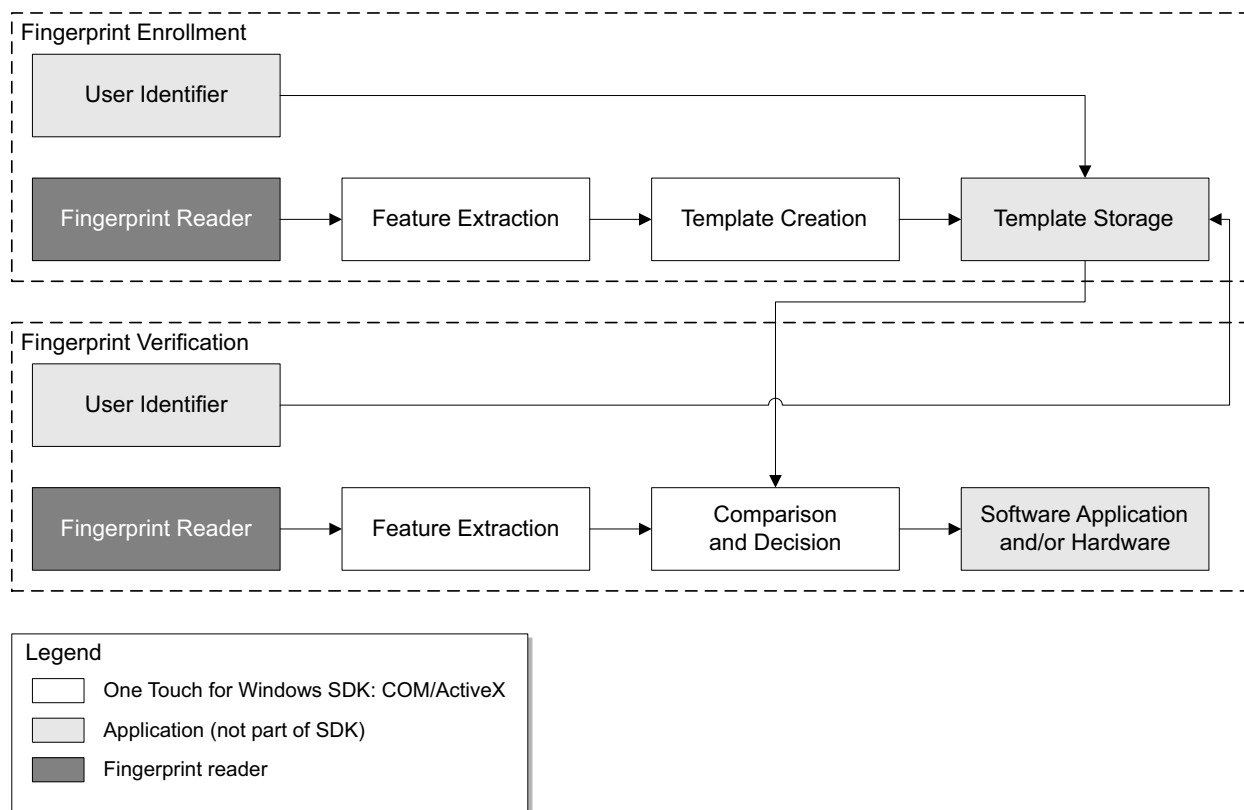
Fingerprint recognition is the most popular and mature biometric system used today. In addition to meeting the four criteria above, fingerprint recognition systems perform well (that is, they are accurate, fast, and robust), they are publicly acceptable, and they are hard to circumvent.

## Fingerprint

A *fingerprint* is an impression of the ridges on the skin of a finger. A *fingerprint recognition system* uses the distinctive and persistent characteristics from the ridges, also referred to as *fingerprint features*, to distinguish one finger (or person) from another. The One Touch for Windows SDK: COM/ActiveX Edition incorporates the *DigitalPersona Fingerprint Recognition Engine (Engine)*, which uses traditional as well as modern fingerprint recognition methodologies to convert these fingerprint features into a format that is compact, distinguishing, and persistent. The Engine then uses the converted, or extracted, fingerprint features in comparison and decision-making to provide reliable personal recognition.

## Fingerprint Recognition

The DigitalPersona fingerprint recognition system uses the processes of fingerprint enrollment and fingerprint verification, which are illustrated in the block diagram in *Figure 1*. Some of the tasks in these processes are done by the *fingerprint reader* and its driver; some are accomplished using One Touch for Windows: COM/ActiveX Edition API functions, which use the Engine; and some are provided by your software application and/or hardware.



**Figure 1.** DigitalPersona fingerprint recognition system



## Fingerprint Enrollment

*Fingerprint enrollment* is the initial process of collecting *fingerprint data* from a person by extracting the fingerprint features (performing *fingerprint feature extraction*) from the person's *fingerprint image* (or *fingerprint sample*) for the purpose of enrollment and then storing the resulting data in a template for later comparison. The following procedure for typical fingerprint enrollment incorporates the One Touch for Windows: COM/ActiveX Edition API fingerprint enrollment function that computes a *fingerprint template* from a required number of *fingerprint feature sets*. (Steps preceded by an asterisk are not done by the fingerprint reader or are not part of the One Touch for Windows SDK: COM/ActiveX Edition.)

1. \*Obtain the user ID of the person to be enrolled.
2. Capture the person's fingerprint as a digital image via the fingerprint reader.
3. Extract the fingerprint features from the image and create a fingerprint feature set for the purpose of enrollment.
4. Create a fingerprint template for the person's finger from a required number of fingerprint feature sets.
5. \*Associate the fingerprint template with the person through an identifier (user ID), such as a PIN, a password, or a user name.
6. \*Store the fingerprint template, along with the user ID, for later comparison.

Fingerprint templates can be stored in any type of repository that you choose, such as a *fingerprint capture device*, a smart card or a central database.

## Fingerprint Verification

*Fingerprint verification* is 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. The following procedure for typical fingerprint verification incorporates the One Touch for Windows: COM/ActiveX Edition API fingerprint verification function that performs a *one-to-one comparison* and makes a decision of *match* or *non-match*. (Steps preceded by an asterisk are not done by the fingerprint reader or are not part of the One Touch for Windows SDK: COM/ActiveX Edition.)

1. \*Obtain the user ID of the person to be verified.
2. Capture the person's fingerprint as a digital image via the fingerprint reader.
3. Extract the fingerprint features from the image and create a fingerprint feature set for the purpose of verification.
4. \*Retrieve the fingerprint template associated with the user ID from your repository.
5. Compare the fingerprint feature set and the fingerprint template, and make a decision of match or non-match.

6. \*Act on the decision accordingly, for example, unlock the door to a building for a match, or deny access to banking records for a non-match.

## False Positives and False Negatives

Fingerprint recognition systems provide many security and convenience advantages over traditional methods of recognition. However, they are essentially pattern recognition systems that inherently occasionally make certain errors because no two impressions of the same finger are identical. During verification, sometimes a person who is legitimately enrolled is rejected by the system (a false negative decision), and sometimes a person who is not enrolled is accepted by the system (a false positive decision).

The proportion of false positive decisions is known as the *false accept rate (FAR)*, and the proportion of false negative decisions is known as the *false reject rate (FRR)*. In fingerprint recognition systems, the FAR and the FRR are traded off against each other, that is, the lower the FAR, the higher the FRR, and the higher the FAR, the lower the FRR.

A One Touch for Windows: COM/ActiveX Edition API function enables you to set the value of the FAR, also referred to as the *security level*, to accommodate the needs of your application. In some applications, such as an access control system to a highly confidential site or database, a lower FAR is required. In other applications, such as an entry system to an entertainment theme park, security (which reduces ticket fraud committed by a small fraction of patrons by sharing their entry tickets) may not be as significant as accessibility for all of the patrons, and it may be preferable to decrease the FRR at the expense of an increased FAR.

It is important to remember that the accuracy of the fingerprint recognition system is largely related to the quality of the fingerprint. Testing with sizable groups of people over an extended period has shown that a majority of people have feature-rich, high-quality fingerprints. These fingerprints will almost surely be recognized accurately by the DigitalPersona Fingerprint Recognition Engine and practically never be falsely accepted or falsely rejected. The DigitalPersona fingerprint recognition system is optimized to recognize fingerprints of poor quality. However, a very small number of people may have to try a second or even a third time to obtain an accurate reading. Their fingerprints may be difficult to verify because they are either worn from manual labor or have unreadable ridges. Instruction in the proper use of the fingerprint reader will help these people achieve the desired results.

## Workflows

*Typical* workflows are presented in this section for the following operations:

- Fingerprint enrollment
- Fingerprint enrollment with UI support
- Fingerprint verification
- Fingerprint verification with UI support
- Fingerprint data object serialization and deserialization

NOTE: Steps preceded by a double dagger (‡) are done by a fingerprint reader, and steps preceded by an asterisk (\*) are performed by an application. "VB *page nn*" and "C++ *page nn*" indicate page references for the Visual Basic API reference and for the C++ API reference, respectively.

### Fingerprint Enrollment Workflow

This section contains a *typical* workflow for performing fingerprint enrollment. The workflow is illustrated in *Figure 2* and is followed by explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflow.

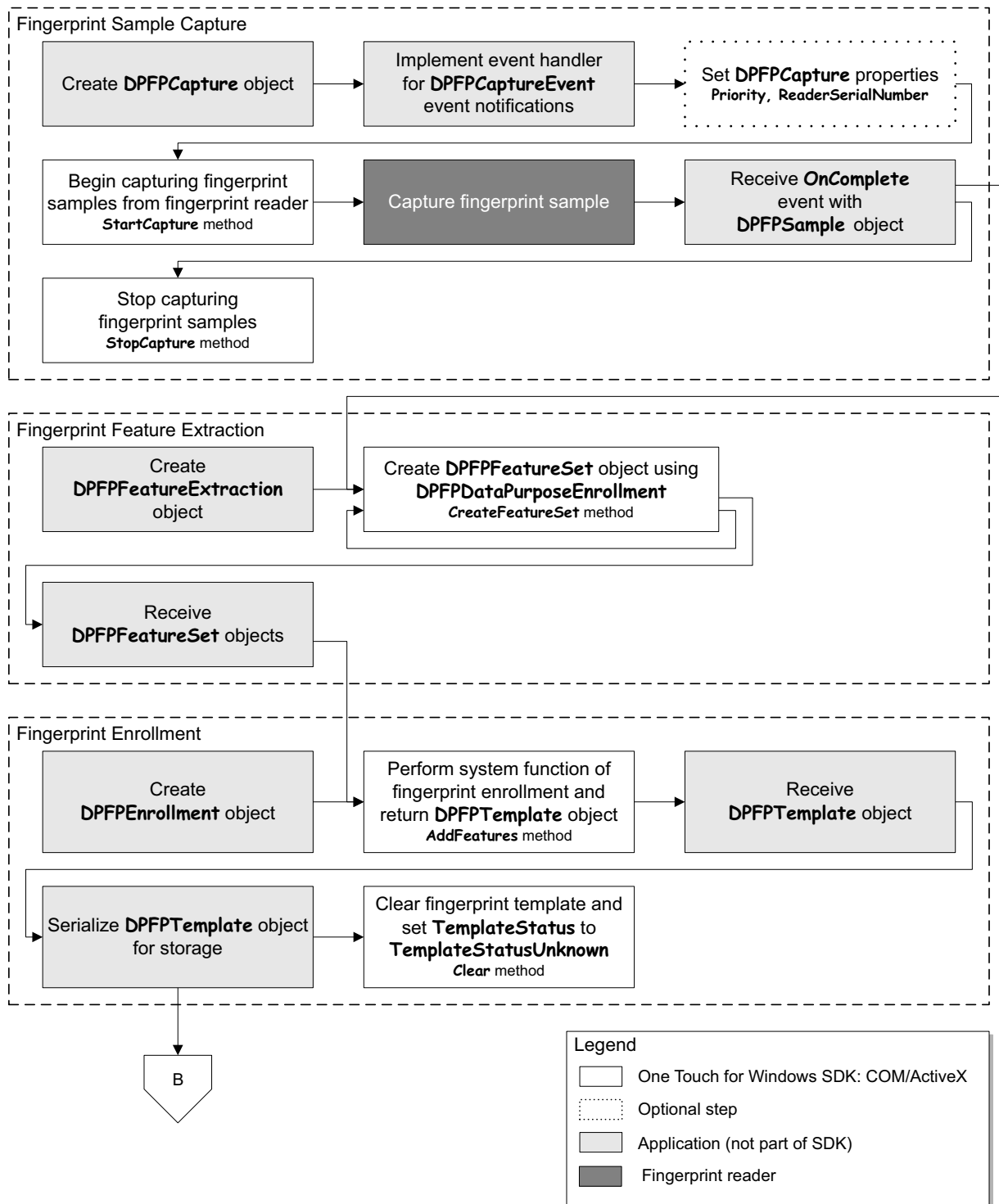


Figure 2. Typical fingerprint enrollment workflow

### ***Fingerprint Sample Capture***

1. \*Create an instance of a **DPFPCapture** object (VB *page 32*, C++ *page 67*).
2. \*Implement an event handler for **DPFPCaptureEvents** event notifications (VB *page 34*, C++ *page 69*).
3. Optionally, set the **Priority** and **ReaderSerialNumber** properties (VB *page 32* and *page 33*; C++ *page 67* and *page 68*).
4. Begin capturing fingerprint samples from the fingerprint reader by calling the **StartCapture** method (VB *page 32*, C++ *page 68*).
5. ‡Capture a fingerprint sample from a fingerprint reader.
6. \*Receive the **OnComplete** event with a **DPFPSample** object when the fingerprint sample is successfully captured by the fingerprint reader (VB *page 34* and *page 51*; C++ *page 70* and *page 91*).
7. Stop capturing fingerprint samples by calling the **StopCapture** method (VB *page 32*, C++ *page 69*).

### ***Fingerprint Feature Extraction***

1. \*Create an instance of a **DPFPFeatureExtraction** object (VB *page 44*, C++ *page 82*).
2. Create **DPFPFeatureSet** objects by calling the **CreateFeatureSet** method using the value **DPFPDataPurposeEnrollment** and passing a **DPFPSample** object (VB *page 44*, C++ *page 82*).
3. \*Receive the **DPFPFeatureSet** objects (VB *page 45*, C++ *page 84*).

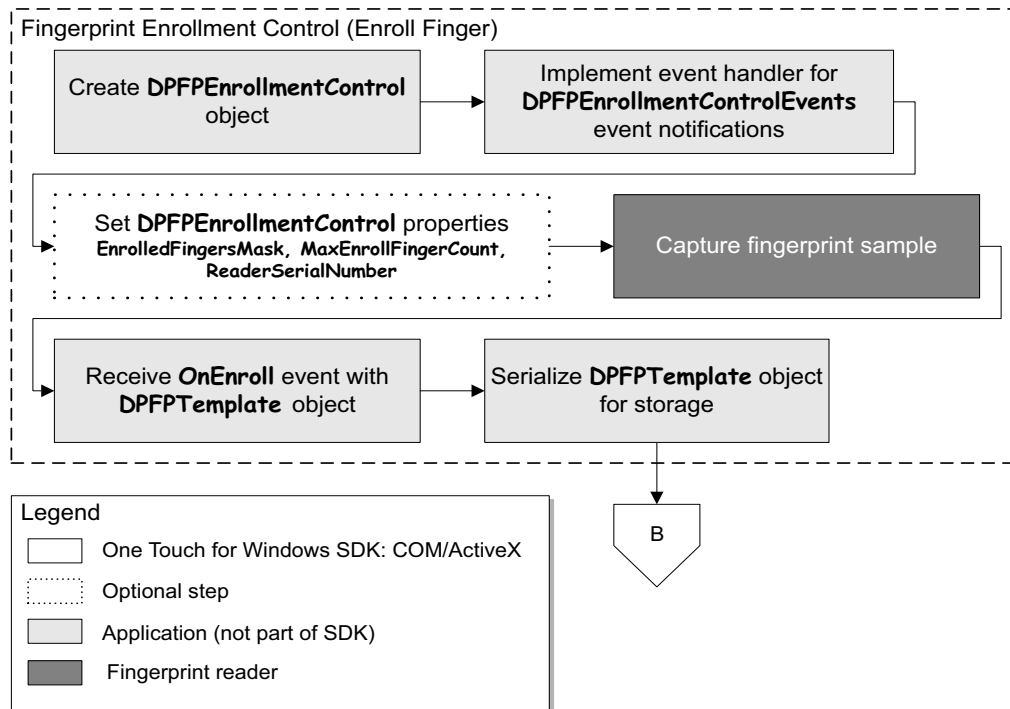
### ***Fingerprint Enrollment***

1. \*Create an instance of a **DPFPEnrollment** object (VB *page 37*, C++ *page 73*).
2. Perform the system function of fingerprint enrollment by calling the **AddFeatures** method and passing the **DPFPFeatureSet** objects (VB *page 37*, C++ *page 73*).  
When the **TemplateStatus** property returns the value **DPFPTemplateStatusReady**, a **DPFPTemplate** object is created (VB *page 38*, C++ *page 75*).
3. \*Receive the **DPFPTemplate** object (VB *page 52*, C++ *page 93*).
4. Serialize the **DPFPTemplate** object (see *Serializing a Fingerprint Data Object* on *page 28*).
5. \*Store the serialized fingerprint template data in a fingerprint storage data subsystem.

## Fingerprint Enrollment with UI Support

This section contains two *typical* workflows for performing fingerprint enrollment: one for enrolling a fingerprint and one for deleting a fingerprint template. The workflows are illustrated in *Figure 3* and *Figure 4* and are followed by explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflows.

### Enrolling a Fingerprint

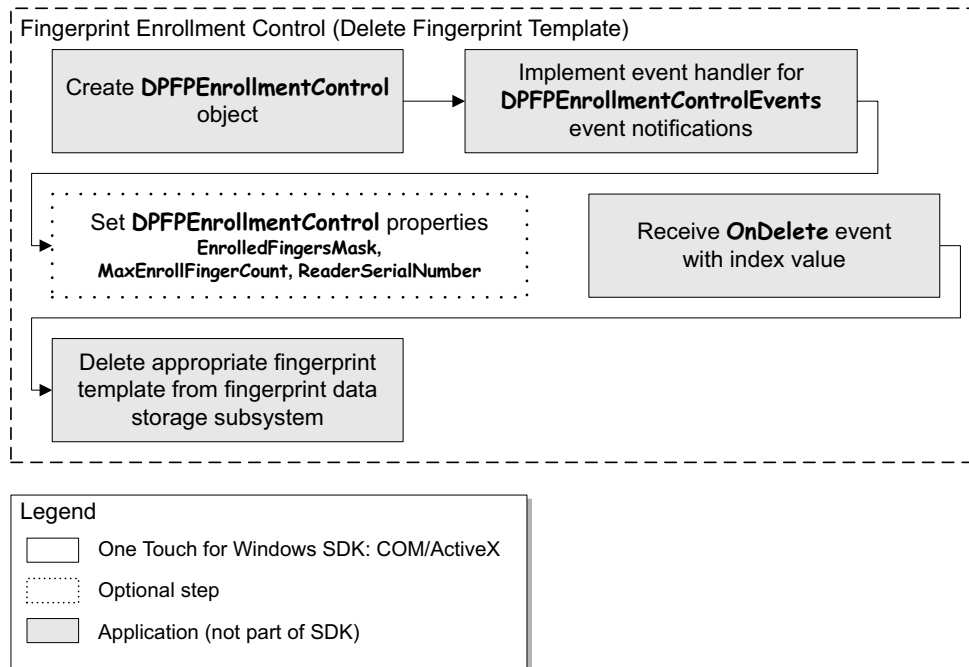


**Figure 3.** Typical fingerprint enrollment with UI support workflow: Enrolling a fingerprint

1. \*Create an instance of a **DPFPEnrollmentControl** object (VB page 39, C++ page 76).
2. \*Implement an event handler for **DPFPEnrollmentControlEvents** event notifications (VB page 42, C++ page 79).
3. Optionally, set the **EnrolledFingersMask**, **MaxEnrollFingerCount**, and **ReaderSerialNumber** properties (VB page 39, page 40, and page 41; C++ page 76, page 77, and page 78).
4. ‡Capture a fingerprint sample from a fingerprint reader.
5. \*Receive the **OnEnroll** event and the **DPFPTemplate** object (VB page 43 and page 52; C++ page 80 and page 93).

6. Serialize the **DPFPFTemplate** object (see *Serializing a Fingerprint Data Object* on page 28).
7. \*Store the serialized fingerprint template data in a fingerprint storage data subsystem.

## Deleting a Fingerprint Template



**Figure 4.** Typical fingerprint enrollment with UI support workflow: Deleting a fingerprint template

1. \*Create an instance of a **DPFPFEnrollmentControl** object (VB page 39, C++ page 76).
2. \*Implement an event handler for **DPFPFEnrollmentControlEvents** event notifications (VB page 42, C++ page 79).
3. Optionally, set the **EnrolledFingersMask**, **MaxEnrollFingerCount**, and **ReaderSerialNumber** properties (VB page 39, page 40, and page 41; C++ page 76, page 77, and page 78).
4. \*Receive the **OnDelete** event and the index value (VB page 42 and page 40; C++ page 79 and page 77).
5. \*Delete the appropriate fingerprint template from the fingerprint data storage subsystem.

## Fingerprint Verification

This section contains a *typical* workflow for performing fingerprint verification. The workflow is illustrated in *Figure 5* and is followed by explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflow.

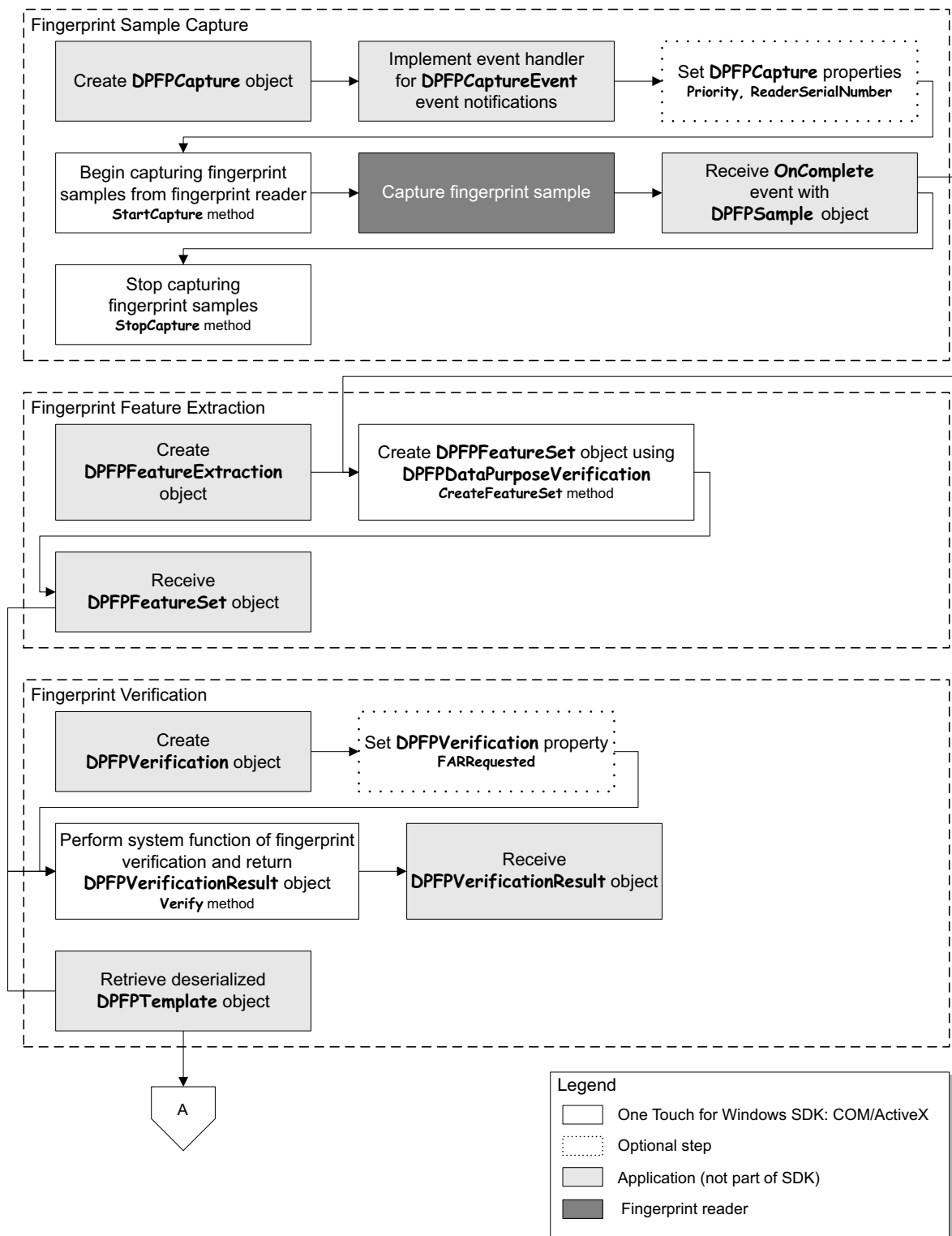


Figure 5. Typical fingerprint verification workflow



### ***Fingerprint Sample Capture***

1. \*Create an instance of a **DPFPCapture** object (VB *page 32*, C++ *page 67*).
2. \*Implement an event handler for **DPFPCaptureEvents** event notifications (VB *page 34*, C++ *page 69*).
3. Optionally, set the **Priority** and **ReaderSerialNumber** properties (VB *page 32* and *page 33*; C++ *page 67* and *page 68*).
4. Begin capturing fingerprint samples from the fingerprint reader by calling the **StartCapture** method (VB *page 32*, C++ *page 68*).
5. ‡Capture a fingerprint sample from a fingerprint reader.
6. \*Receive the **OnComplete** event with a **DPFPSample** object when the fingerprint sample is successfully captured by the fingerprint reader (VB *page 34* and *page 51*; C++ *page 70* and *page 91*).
7. Stop capturing fingerprint samples by calling the **StopCapture** method (VB *page 32*, C++ *page 69*).

### ***Fingerprint Feature Extraction***

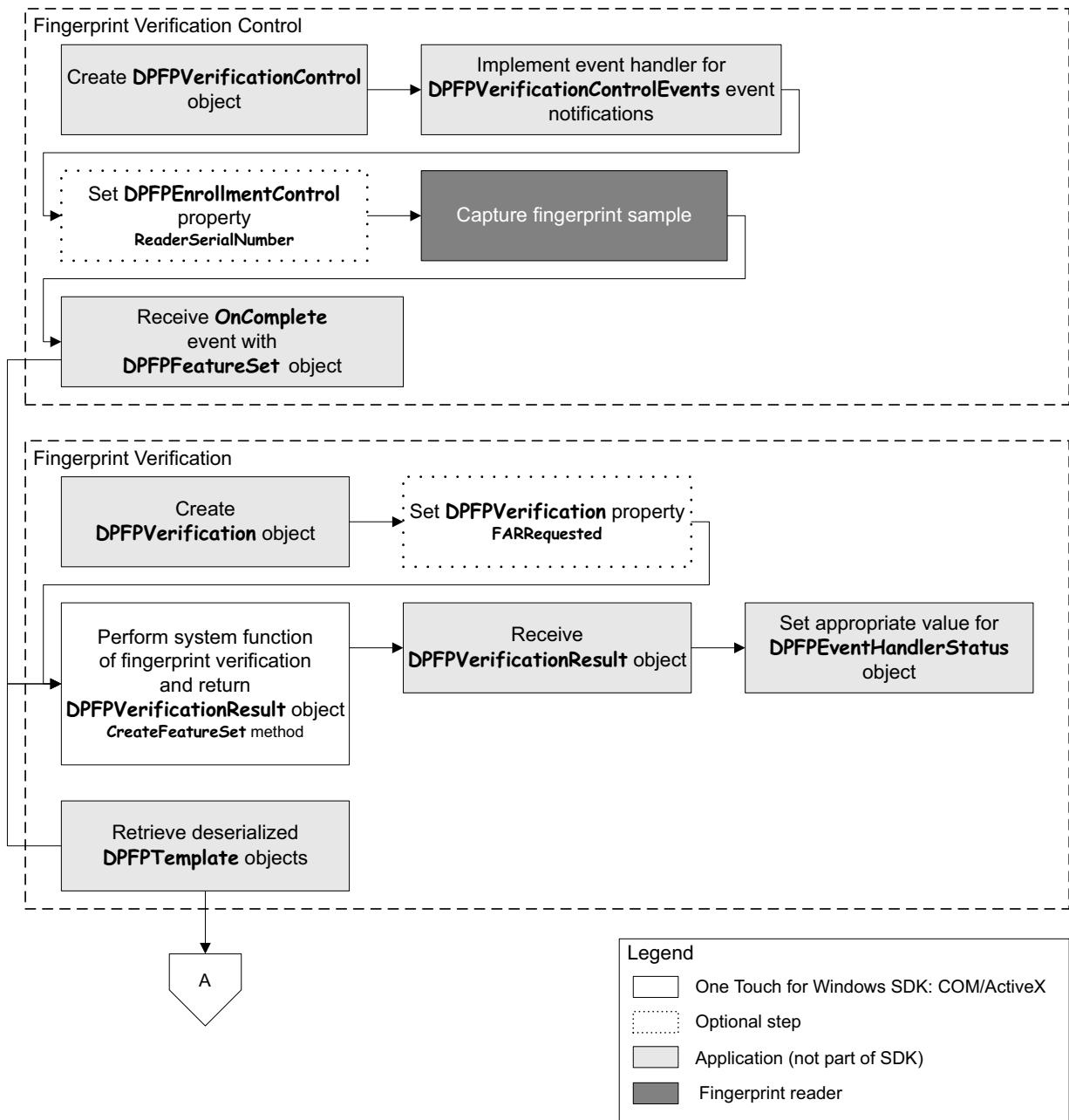
1. \*Create an instance of a **DPFPFeatureExtraction** object (VB *page 44*, C++ *page 82*).
2. Create a **DPFPFeatureSet** object by calling the **CreateFeatureSet** method using the value **DPFPDataPurposeVerification** and passing the **DPFPSample** object (VB *page 44*, C++ *page 82*).
3. \*Receive the **DPFPFeatureSet** object (VB *page 45*, C++ *page 84*).

### ***Fingerprint Verification***

1. \*Create an instance of a **DPFPVerification** object (VB *page 52*, C++ *page 93*).
2. Optionally, set the **FARRequested** property (VB *page 53*, C++ *page 93*).
3. \*Retrieve the serialized fingerprint template data from the fingerprint data storage subsystem.
4. Create a **DPFPTemplate** object from the serialized data (see *Deserializing a Serialized Fingerprint Data Object* on *page 29*).
5. Perform the system function of fingerprint verification by calling the **Verify** method and passing the **DPFPTemplate** and **DPFPFeatureSet** objects (VB *page 53*, C++ *page 94*).
6. \*Receive the **DPFPVerificationResult** object, which provides the comparison decision of match or non-match (VB *page 56*, C++ *page 97*).

## Fingerprint Verification with UI Support

This section contains a *typical* workflow for performing fingerprint verification with UI support. The workflow is illustrated in *Figure 6* and is followed by explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflow.



**Figure 6.** Typical fingerprint verification with UI support workflow

### ***Fingerprint Verification Control***

1. \*Create an instance of a **DPFPVerificationControl** object (VB *page 54*, C++ *page 95*).
2. Implement an event handler for **DPFPVerificationControlEvents** event notifications (VB *page 42*, C++ *page 97*).
3. Optionally, set the **ReaderSerialNumber** property (VB *page 54*, C++ *page 96*).
4. ‡Capture a fingerprint sample from a fingerprint reader.
5. Receive the **OnComplete** event with the **DPFPFeatureSet** object (VB *page 55* and *page 45*, C++ *page 97* and *page 84*).

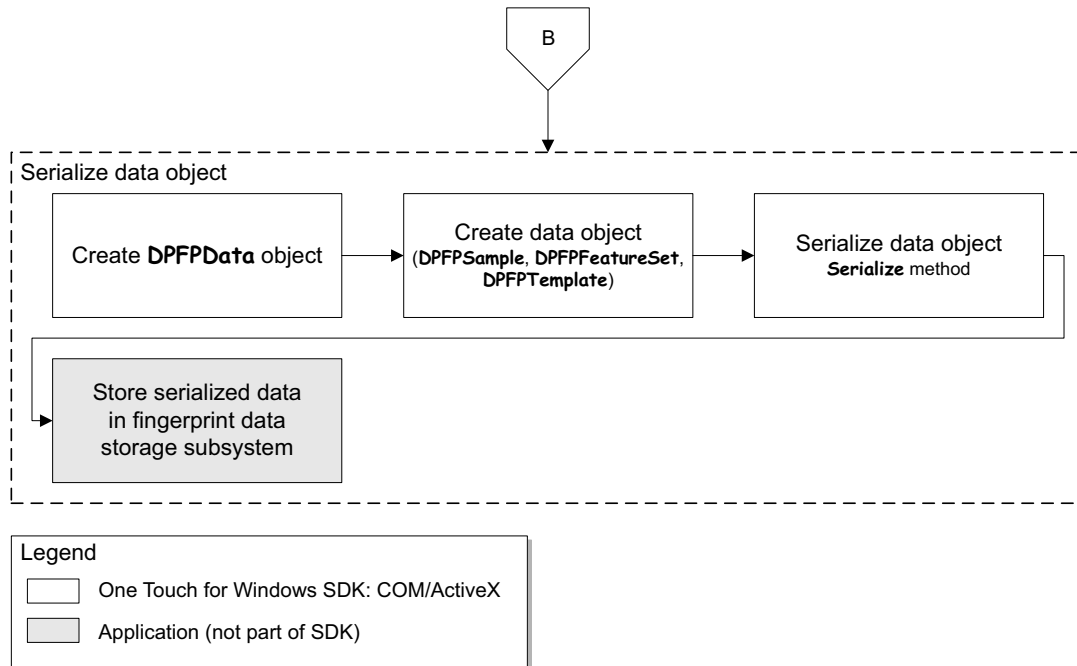
### ***Fingerprint Verification***

1. \*Create an instance of a **DPFPVerification** object (VB *page 52*, C++ *page 93*).
2. Optionally, set the **FARRequested** property (VB *page 53*, C++ *page 93*).
3. \*Retrieve the serialized fingerprint template data from the fingerprint data storage subsystem.
4. Create a **DPFPTemplate** object from the serialized data (see *Deserializing a Serialized Fingerprint Data Object* on *page 29*).
5. Perform the system function of fingerprint verification by calling the **Verify** method and passing the **DPFPTemplate** and **DPFPFeatureSet** objects (VB *page 53*, C++ *page 94*).
6. \*Receive the **DPFPVerificationResult** object, which provides the comparison decision of match or non-match (VB *page 56*, C++ *page 97*).

## Fingerprint Data Object Serialization/Deserialization

This section contains two workflows: one for serializing a fingerprint data object and one for deserializing a serialized fingerprint data object. The workflows are illustrated in *Figure 7* and *Figure 8* and are followed by explanations of the One Touch for Windows: COM/ActiveX Edition API functions used to perform the tasks in the workflows.

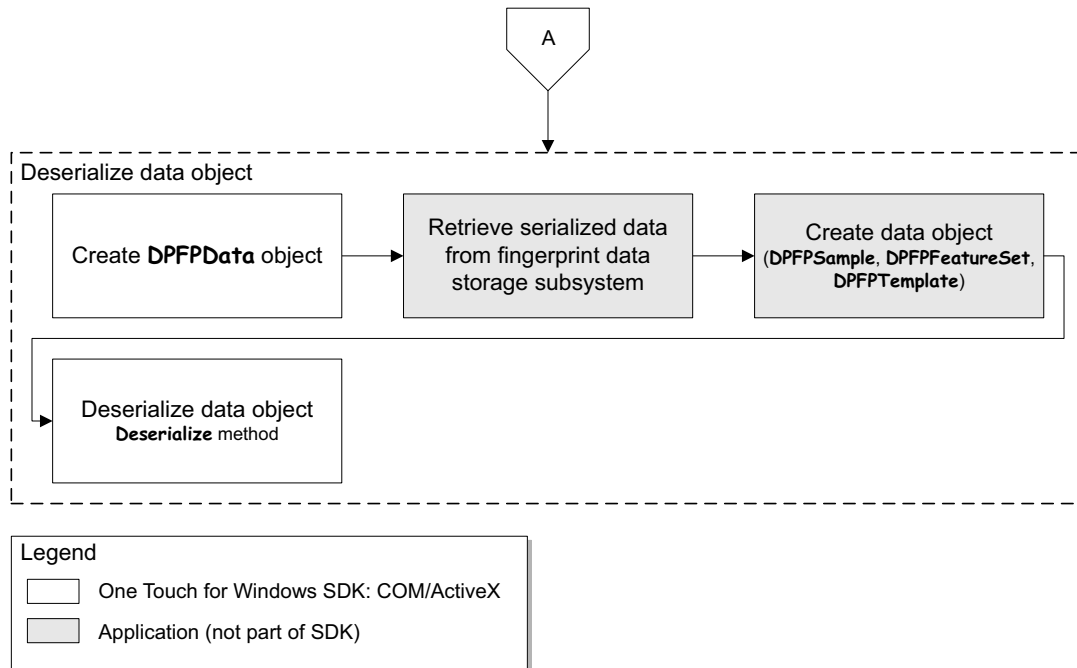
### Serializing a Fingerprint Data Object



**Figure 7.** Fingerprint data object serialization workflow

1. \*Create an instance of a **DPFPData** object (VB *page 36*, C++ *page 72*).
2. Create a fingerprint data object. (See the various methods and properties for creating and returning **DPFPsSample**, **DPFPFeatureSet**, and **DPFPTemplate** objects.)
3. Serialize the data object by calling the **Serialize** method (VB *page 36*, C++ *page 72*).
4. \*Store the serialized data in a fingerprint data storage subsystem.

## Deserializing a Serialized Fingerprint Data Object



**Figure 8.** Deserialization of serialized fingerprint data object workflow

1. \*Create an instance of a **DPFPData** object (VB *page 36*, C++ *page 72*).
2. \*Retrieve the serialized data from a fingerprint data storage subsystem.
3. \*Create an instance of a **DPFPsample**, **DPFPFeatureSet**, or **DPFPTemplate** object (VB *page 51*, *page 45*, and *page 52*; C++ *page 91*, *page 84*, and *page 93*).
4. Deserialize the fingerprint data object by calling the **Deserialize** method (VB *page 36*, C++ *page 72*).

This chapter defines the component objects (including methods, properties, and events) and the enumerations for developing applications that incorporate the functionality of the One Touch for Windows: COM/ActiveX Edition API in Visual Basic using the Component Object Model (COM) implementation.

## Component Objects

**IMPORTANT:** All of the read/write properties of the One Touch for Windows SDK API component objects are optional. If you do not set one of these properties, the default value is automatically used. When deciding whether to set a parameter, be aware that DigitalPersona may change the default values at any time without notice. If you want your application's functionality to remain consistent, you should set the properties accordingly.

The One Touch for Windows: COM/ActiveX Edition API COM implementation includes the component objects defined in this section. Use the following list to quickly locate an object by name, by page number, or by description.

Method	Page	Description
<b>DPFPCapture</b>	32	Captures a fingerprint sample from a fingerprint reader
<b>DPFPData</b>	36	Represents the data that is common to all fingerprint data objects
<b>DPFPEnrollment</b>	37	Performs the system function of fingerprint enrollment
<b>DPFPEnrollmentControl</b>	39	Contains an ActiveX control for creating and returning a fingerprint template and for deleting a fingerprint template, and provides a user interface
<b>DPFPEventHandlerStatus</b>	43	Returns codes that indicate the status of an operation
<b>DPFPFeatureExtraction</b>	44	Performs the system function of fingerprint feature extraction
<b>DPFPFeatureSet</b>	45	Represents a fingerprint feature set
<b>DPFPReaderDescription</b>	46	Provides information about a particular fingerprint reader
<b>DPFPReadersCollection</b>	49	Provides information about all of the fingerprint readers connected to a system
<b>DPFPSTemplate</b>	51	Represents a fingerprint sample
<b>DPFPSTemplateConversion</b>	51	Converts a fingerprint sample to an image for display
<b>DPFPSTemplate</b>	52	Represents a fingerprint template
<b>DPFPVerification</b>	52	Performs the system function of fingerprint verification

Method	Page	Description
<b>DPFPVerificationControl</b>	54	Contains an ActiveX control for creating and returning a fingerprint feature set created for the purpose of verification, and provides a user interface
<b>DPFPVerificationResult</b>	56	Represents the results of a fingerprint verification operation

## DPFPCapture

The **DPFPCapture** object captures a fingerprint sample from a fingerprint reader.

### Methods

#### StartCapture Method

Begins capturing a fingerprint sample from a fingerprint reader. A call to this method is asynchronous and returns immediately. The application continues to receive events until the **StopCapture** method is called or when the **DPFPCapture** object is destroyed.

#### Syntax

```
object.StartCapture()
```

#### Possible Errors

Error Code	Message	Description
-2147024809	One or more arguments are invalid.	A capture operation with the specified priority already exists. See <b>DPFPCapturePriorityEnum</b> on <i>page 59</i> for more information.
-2147024891	General access denied error.	The application does not have sufficient privileges to start capture operations with the specified priority. See <b>DPFPCapturePriorityEnum</b> on <i>page 59</i> for more information.

#### StopCapture Method

Stops the fingerprint sample capture operation started with a call to the **StartCapture** method. This method is optional.

#### Syntax

```
object.StopCapture()
```

### Properties

#### Priority Property

Gets or sets a value that specifies the priority of a fingerprint sample capture operation.

#### Syntax

```
DPFPCapture.Priority [ = enumValue ]  
[ enumValue = ] DPFPCapture.Priority
```



## Possible Values

<b>enumValue</b>	<b>Enum</b> that specifies or receives one of the <b>DPFPCapturePriorityEnum</b> enumeration values ( <i>page 59</i> )
------------------	--

This optional property is read/write. If you do not set it, the value **DPFPCapturePriorityNormal** is used.

## Possible Errors

Error Code	Message	Description
-2147352566	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

## ReaderSerialNumber Property

Gets or sets the serial number of a fingerprint reader that captures a fingerprint sample.

### Syntax

```
DPFPCapture.ReaderSerialNumber [ = bstrValue ]
[ bstrValue = ] DPFPCapture.ReaderSerialNumber
```

## Possible Values

<b>strValue</b>	<b>String</b> that specifies or receives a fingerprint reader serial number
-----------------	---

This optional property is read/write. If you do not set it, the following value is used: {00000000-0000-0000-0000-000000000000}. This means that the application will receive events from any of the fingerprint readers attached to the system.

## Possible Errors

Error Code	Message	Description
-2147024809	One or more arguments are invalid.	The format of the string containing the fingerprint reader serial number is incorrect. It should be in GUID format, for example, {A9EFB3F6-A8C8-4684-841E-4330973057C6}.

## Object Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## Events

### OnComplete Event

Fires when a fingerprint sample is successfully captured by a fingerprint reader.

#### Syntax

```
Private Sub object_OnComplete(  
    ByVal bstrReaderSerNum As String,  
    ByVal oFingerprintSample As Object)
```

#### Parameters

<b>bstrReaderSerNum</b>	<b>String</b> that specifies a fingerprint reader serial number
<b>oFingerprintSample</b>	A <b>DPFPFSample</b> object ( <i>page 51</i> )

### OnFingerGone Event

Fires when a user removes a finger from a fingerprint reader.

#### Syntax

```
Private Sub object_OnFingerGone(  
    ByVal bstrReaderSerNum As String)
```

#### Parameter

<b>bstrReaderSerNum</b>	<b>String</b> that specifies a fingerprint reader serial number
-------------------------	---

### OnFingerTouch Event

Fires when a user touches a fingerprint reader.

#### Syntax

```
Private Sub OnFingerTouch(  
    ByVal bstrReaderSerNum As String)
```

## Parameter

---

<b>bstrReaderSerNum</b>	<b>String</b> that specifies a fingerprint reader serial number
-------------------------	---

---

## OnReaderConnect Event

Fires when a fingerprint reader is attached to a system.

### Syntax

```
Private Sub object_OnReaderConnect(  
    ByVal bstrReaderSerNum As String)
```

## Parameter

---

<b>bstrReaderSerNum</b>	<b>String</b> that specifies a fingerprint reader serial number
-------------------------	---

---

## OnReaderDisconnect Event

Fires when a fingerprint reader is disconnected from a system.

### Syntax

```
Private Sub object_OnReaderDisconnect(  
    ByVal bstrReaderSerNum As String)
```

## Parameter

---

<b>bstrReaderSerNum</b>	<b>String</b> that specifies a fingerprint reader serial number
-------------------------	---

---

## OnSampleQuality Event

Fires when the quality of a fingerprint sample is verified. When **SampleQualityGood** is returned in the **SampleQuality** parameter, the **OnComplete** event is fired ([page 34](#)).

### Syntax

```
Private Sub object_OnSampleQuality(  
    ByVal bstrReaderSerNum As String,  
    ByVal enumSampleQuality As Enum)
```

## Parameters

<b>bstrReaderSerNum</b>	<b>String</b> that specifies a fingerprint reader serial number
<b>enumSampleQuality</b>	<b>Enum</b> that specifies one of the values, which provides feedback about a fingerprint sample capture operation, from the <b>DPFPCaptureFeedbackEnum</b> enumeration ( <a href="#">page 58</a> )

## DPFPData

Represents the data that is common to all *fingerprint data objects*. The **DPFPData** object also provides methods to serialize and deserialize the fingerprint data objects.

## Methods

### Deserialize Method

Deserializes a data object returned by the **Serialize** method.

#### Syntax

```
object.Deserialize(  
    ByRef aRawData() As Byte)
```

#### Parameter

<b>aRawData</b>	<b>Array of bytes</b> that specifies a deserialized data object
-----------------	---

## Possible Errors

Error Code	Message	Description
-2147024809	One or more arguments are invalid.	The format of the data passed to the <b>Deserialize</b> method is incorrect.

### Serialize Method

Serializes a data object and returns it as an array of bytes.

#### Syntax

```
Dim aRawData As Byte()  
aRawData = object.Serialize
```

## Parameter

<b>aRawData</b>	<b>Array of bytes</b> that receives a serialized data object
-----------------	--

## Object Information

Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## See Also

**DPFPFeatureSet** on page 45

**DPFPSample** on page 51

**DPFPTemplate** on page 52

# DPFPEnrollment

The **DPFPEnrollment** object performs the system function of *fingerprint enrollment*. This object creates a fingerprint template from a specified number of fingerprint feature sets created for the purpose of enrollment.

## Methods

### AddFeatures Method

Adds fingerprint feature sets, one-by-one, to a fingerprint template. The fingerprint template is complete when the **TemplateStatus** property is set to the value **TemplateStatusReady**.

#### Syntax

```
object.AddFeatures(  
    ByVal oFeatures As Object)
```

## Parameter

<b>oFeatures</b>	A <b>DPFPFeatureSet</b> object (page 45)
------------------	--

### Clear Method

Clears a fingerprint template and sets the value of the **TemplateStatus** property to **TemplateStatusUnknown** so an application can begin another fingerprint template creation operation.

#### Syntax

```
object.Clear()
```

## Properties

### FeaturesNeeded Property

Gets the number of fingerprint feature sets still needed to create a fingerprint template. When the value of **lValue** is equal to 0, the fingerprint template is created.

#### Syntax

```
[ lValue = ] DPFPEnrollment.FeaturesNeeded
```

#### Possible Values

<b>lValue</b>	<b>Long</b> that receives the value of the number of fingerprint feature sets
---------------	---

This property is read-only and has no default value.

### Template Property

Gets a **DPFPTemplate** object created during a fingerprint enrollment operation.

#### Syntax

```
[ oTemplate = ] DPFPEnrollment.Template
```

#### Possible Values

<b>oTemplate</b>	A <b>DPFPTemplate</b> object ( <i>page 52</i> )
------------------	---

This property is read-only and has no default value.

#### Possible Errors

<b>Error Code</b>	<b>Message</b>	<b>Description</b>
-2147352573	Member not found.	A fingerprint template has not been created yet.

### TemplateStatus Property

Gets a value that specifies the status of a fingerprint template creation operation.

#### Syntax

```
[ enumValue = ] DPFPEnrollment.TemplateStatus
```

## Possible Values

<b>enumValue</b>	<b>Enum</b> that receives one of the <b>DPFPCTemplateStatusEnum</b> enumeration values ( <i>page 64</i> )
------------------	---

This property is read-only and has no default value.

## Object Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## DPFPEnrollmentControl

The **DPFPEnrollmentControl** object contains an ActiveX control that implements a user interface (described in *DPFPEnrollmentControl Object User Interface on page 107*) and provides the following functionality:

- Captures a fingerprint sample from a fingerprint reader
- Creates a specified number of fingerprint feature sets for the purpose of enrollment
- Creates a fingerprint template
- Deletes a fingerprint template
- Fires events

## Properties

### EnrolledFingersMask Property

Gets or sets the mask representing the user's enrolled fingerprints. The enrollment mask is a combination of the values representing a user's enrolled fingerprints. For example, if a user's right index fingerprint and right middle fingerprint are enrolled, the value of this property is 00000000 011000000, or 192.

### Syntax

```
DPFPEnrollmentControl.EnrolledFingersMask [ = lValue ]
[ lValue = ] DPFPEnrollmentControl.EnrolledFingersMask
```

## Possible Values

<b>lValue</b>	<b>Long</b> that specifies or receives the value of the fingerprint mask. All possible values are listed in <i>Table 3</i> .
---------------	--

**Table 3.** Values for the enrollment mask

Finger	Binary Representation	Integer Representation
Left little finger	000000000 000000001	1
Left ring finger	000000000 000000010	2
Left middle finger	000000000 000000100	4
Left index finger	000000000 000001000	8
Left thumb	000000000 000010000	16
Right thumb	000000000 000100000	32
Right index finger	000000000 001000000	64
Right middle finger	000000000 010000000	128
Right ring finger	000000000 100000000	256
Right little finger	000000001 000000000	512

This optional property is read/write. If you do not set it, the value `0` is used, which means that no fingerprints have been enrolled.

### Possible Errors

Error Code	Message	Description
-2147352566	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

### MaxEnrollFingerCount Property

Gets or sets the value for the maximum number of fingerprints that can be enrolled.

#### Syntax

```
DPFPEnrollmentControl.MaxEnrollFingerCount [ = lValue ]
[ lValue = ] DPFPEnrollmentControl.MaxEnrollFingerCount
```

#### Possible Values

<b>lValue</b>	<b>Long</b> that specifies or receives the value for the maximum number of fingerprints that can be enrolled. Possible values are <code>1</code> through <code>10</code> .
---------------	--



This optional property is read/write. If you do not set it, the value `10` is used, which means the user can enroll all ten fingerprints.

### Possible Errors

Error Code	Message	Description
-2147352566	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

### ReaderSerialNumber Property

Gets or sets the serial number of the fingerprint reader from which a fingerprint sample is captured.

### Syntax

```
DPFPEnrollmentControl.ReaderSerialNumber [ = bstrValue ]
[ bstrValue = ] DPFPEnrollmentControl.ReaderSerialNumber
```

### Possible Values

<b>bstrValue</b>	<b>String</b> that specifies or receives the fingerprint reader serial number
------------------	---

This optional property is read/write. If you do not set it, the following value is used: `{00000000-0000-0000-0000-000000000000}`. This means that the application will receive events from any of the fingerprint readers attached to the system.

### Possible Errors

Error Code	Message	Description
-2147024809	One or more arguments are invalid.	The format of the string containing the fingerprint reader serial number is incorrect. It should be in GUID format, for example, <code>{A9EFB3F6-A8C8-4684-841E-4330973057C6}</code> .

### Object Information

Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPctIX.dll

## Events

### OnDelete Event

Fires when a user deletes a finger. The application handles the deletion of the fingerprint template from a fingerprint data storage subsystem and can display its own success or error messages.

#### Syntax

```
Private Sub object_OnDelete(  
    ByVal l1FingerMask As Long,  
    ByVal oStatus As Object)
```

#### Parameters

<b>l1FingerMask</b>	<b>Long</b> that specifies the index value of the (enrolled) fingerprint to be deleted. For possible values, see <i>Table 4</i> .
<b>oStatus</b>	A <b>DPFPEventHandlerStatus</b> object ( <i>page 43</i> )

The **l1FingerprintMask** parameter is the index value of the finger associated with a fingerprint to be enrolled or a fingerprint template to be deleted, as defined in ANSI/NIST-ITL 1. The index values are assigned to the graphical representation of the fingers on the hands in the user interface. All possible values are listed in *Table 4*.

**Table 4.** Finger index values in ANSI/NIST-ITL 1

Finger	Index Value	Finger	Index Value
Right thumb	1	Left thumb	6
Right index finger	2	Left index finger	7
Right middle finger	3	Left middle finger	8
Right index finger	4	Left ring finger	9
Right little finger	5	Left little finger	10

## OnEnroll Event

Fires when a user enrolls a fingerprint and returns a fingerprint template. The application handles the storage of the fingerprint template in a fingerprint data storage subsystem and can display its own success or error messages.

### Syntax

```
Private Sub object_OnEnroll(  
    ByVal llFingerMask As Long,  
    ByVal oFingerprintTemplate As Object,  
    ByVal oStatus As Object)
```

### Parameters

<b>llFingerMask</b>	<b>Long</b> that specifies the index value for the enrolled fingerprint. For possible values, see Table 4 on <i>page 42</i> .
<b>oFingerprintTemplate</b>	A <b>DPFPFTemplate</b> object ( <i>page 52</i> )
<b>oStatus</b>	A <b>DPFPEventHandlerStatus</b> object ( <i>page 43</i> )

## DPFPEventHandlerStatus

The **DPFPEventHandlerStatus** object returns codes that indicate the status of an operation.

### Properties

#### Status Property

Gets or sets the status of an operation performed by a **DPFPFEnrollmentControl** object (*page 39*) or by a **DPFPFVerificationControl** object (*page 54*).

### Syntax

```
DPFPEventHandlerStatus.Status [ = enumValue ]  
[ enumValue = ] DPFPEventHandlerStatus.Status
```

### Possible Values

<b>enumValue</b>	<b>Enum</b> that specifies or receives one of the values from the <b>DPFPEventHandlerStatusEnum</b> enumeration ( <i>page 60</i> )
------------------	--

This optional property is read/write. If you do not set it, the value **DPFPEventHandlerStatusSuccess** is used.

## Possible Errors

Error Code	Message	Description
-2147352566	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

## Object Information

Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPCtIX.dll

## DPFPFeatureExtraction

The **DPFPFeatureExtraction** object performs *fingerprint feature extraction*. This object creates a fingerprint feature set for the purpose of enrollment or verification by applying fingerprint feature extraction to a fingerprint sample.

## Method

### CreateFeatureSet Method

Applies fingerprint feature extraction to a fingerprint sample and then creates a fingerprint feature set for the specified purpose.

### Syntax

```
Dim enumSampleQuality As DPFPCaptureFeedbackEnum
enumSampleQuality = object.CreateFeatureSet(
    ByVal oFingerprintSample As Object,
    ByVal enumPurpose As Enum)
```

### Parameters

<b>oFingerprintSample</b>	A <b>DPFPSample</b> object ( <i>page 51</i> )
<b>enumPurpose</b>	<b>Enum</b> that specifies one of the values, which is for the specified purpose, from the <b>DPFPDataPurposeEnum</b> enumeration ( <i>page 61</i> )
<b>enumSampleQuality</b>	<b>Enum</b> the receives one of the values, which provides feedback about a fingerprint sample capture operation, from the <b>DPFPCaptureFeedbackEnum</b> enumeration ( <i>page 58</i> )

## Property

### FeatureSet Property

Retrieves a **DPFPFeatureSet** object created during a fingerprint feature extraction operation.

#### Syntax

```
[ oFeatureSet = ] DPFPFeatureExtraction.FeatureSet
```

#### Possible Values

<b>oFeatureSet</b>	A <b>DPFPFeatureSet</b> object (page 45)
--------------------	--

This property is read-only and has no default value.

#### Possible Errors

Error Code	Message	Description
-2147352573	Member not found.	A fingerprint feature set has not been created yet.

## Object Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## DPFPFeatureSet

The **DPFPFeatureSet** object represents a fingerprint feature set.

## Methods and Properties

None.

## Object Information

Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## DPFPReaderDescription

The **DPFPReaderDescription** object provides information about a particular fingerprint reader, such as its technology or serial number.

### Properties

#### FirmwareRevision Property

Gets the firmware revision number of a fingerprint reader.

##### Syntax

```
[ bstrValue = ] DPFPReaderDescription.FirmwareRevision
```

##### Possible Values

<b>bstrValue</b>	<b>String</b> the receives the fingerprint reader firmware revision number
------------------	--

This property is read-only and has no default value.

#### HardwareRevision Property

Gets the hardware revision number of a fingerprint reader.

##### Syntax

```
[ bstrValue = ] DPFPReaderDescription.HardwareRevision
```

##### Possible Values

<b>bstrValue</b>	<b>String</b> the receives the fingerprint reader hardware revision number
------------------	--

This property is read-only and has no default value.

#### Language Property

Gets the fingerprint reader language.

##### Syntax

```
[ bstrValue = ] DPFPReaderDescription.get_Language
```

##### Possible Values

<b>bstrValue</b>	<b>String</b> the receives the fingerprint reader language. The value of <b>bstrValue</b> is always 0x409, which is English.
------------------	--

This property is read-only and has no default value.

### ImpressionType Property

Gets a value that specifies the fingerprint reader impression type, for example, swipe reader or touch (area) reader.

#### Syntax

```
[ enumValue = ] DPFPReaderDescription.ImpressionType
```

#### Possible Values

<b>enumValue</b>	<b>Enum</b> that receives one of the values from the <b>DPFPReaderImpressionTypeEnum</b> enumeration ( <i>page 62</i> )
------------------	---

This property is read-only and has no default value.

### ProductName Property

Gets the product name of a fingerprint reader, for example, "U.are.U."

#### Syntax

```
[ bstrValue = ] DPFPReaderDescription.ProductName
```

#### Possible Values

<b>bstrValue</b>	<b>String</b> that receives the fingerprint reader product name
------------------	---

This property is read-only and has no default value.

### SerialNumber Property

Gets the serial number of a fingerprint reader. This property is read-only and has no default value.

#### Syntax

```
[ bstrValue = ] DPFPReaderDescription.SerialNumber
```

#### Possible Values

<b>bstrValue</b>	<b>String</b> the receives the fingerprint reader serial number
------------------	---

This property is read-only and has no default value.

## SerialNumberType Property

Gets a value that specifies the type of fingerprint reader serial number.

### Syntax

```
[ enumValue = ] DPFPReaderDescription.SerialNumberType
```

### Possible Values

enumValue	Enum that receives one of the values from the <code>DPFPSerialNumberTypeEnum</code> enumeration ( <i>page 63</i> )
-----------	--

This property is read-only and has no default value.

## Technology Property

Gets a value that specifies the fingerprint reader technology.

### Syntax

```
[ enumValue = ] DPFPReaderDescription.Technology
```

### Possible Values

enumValue	Enum that receives one of the values from the <code>DPFPReaderTechnologyEnum</code> enumeration ( <i>page 62</i> )
-----------	--

This property is read-only and has no default value.

## Vendor Property

Gets the vendor name for a fingerprint reader, for example, "DigitalPersona, Inc."

### Syntax

```
[ bstrValue = ] DPFPReaderDescription.Vendor
```

### Possible Values

bstrValue	String the receives the fingerprint reader vendor name
-----------	--

This property is read-only and has no default value.



## Object Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPReadersCollection

The **DPFPReadersCollection** object provides information about all of the fingerprint readers connected to a system.

### Method

#### Reader Method

Returns a **DPFPReaderDescription** object for a particular fingerprint reader using its serial number.

#### Syntax

```
Dim oReader As DPFPReaderDescription
Set oReader = object.Reader(
    ByVal bstrReaderSerialNum As String)
```

#### Parameters

<b>bstrReaderSerialNumber</b>	<b>String</b> that specifies a fingerprint reader serial number
<b>oReader</b>	A <b>DPFPReaderDescription</b> object ( <i>page 46</i> )

#### Possible Errors

Error Code	Message	Description
-2147024894	The system cannot find the specified file.	The fingerprint reader with the specified serial number cannot be found in the system.

## Properties

#### Count Property

Gets the total number of **DPFPReaderDescription** objects (items) connected to a system (a collection).

#### Syntax

```
[ lCount = ] DPFPReadersCollection.Count
```

## Possible Values

<b>lCount</b>	<b>Long</b> that receives the total number of <b>DPFPReaderDescription</b> objects
---------------	--

This property is read-only and has no default value.

## Item Property

Gets or sets a **DPFPReaderDescription** object (an item) from the fingerprint readers connected to a system (a collection) using its index.

## Syntax

```
[ lReader = ] DPFPReadersCollection.Item
```

## Possible Values

<b>lReader</b>	<b>Long</b> that specifies the index of the <b>DPFPReaderDescription</b> object to retrieve from the collection. The value of <b>lReader</b> starts with <b>1</b> .
----------------	---

This property is read-only and has no default value.

## Possible Errors

Error Code	Message	Description
-2147352565	Invalid index.	The specified index is not in the valid range from <b>1</b> to <b>Count</b> .

## \_NewEnum Property

Gets a **ReaderEnum** object (enumeration object), which is an array of **DPFPReaderDescription** objects.

## Syntax

```
[ aReaderEnum = ] DPFPReadersCollection._NewEnum
```

## Possible Values

<b>aReaderEnum</b>	<b>IUnknown</b> that receives the array of <b>DPFPReaderDescription</b> objects
--------------------	---

This property is read-only and has no default value.

## Object Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPSTDevX.dll

## DPFPSTSample

The **DPFPSTSample** object represents a fingerprint sample captured from a fingerprint reader.

## Methods and Properties

None.

## Object Information

Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPSTShrX.dll

## See Also

**DPFPSTData** on page 36

## DPFPSTSampleConversion

The **SampleConversion** object provides methods for returning a fingerprint sample as an **IPicture** object and as an image in ANSI 381 format that can be used for display.

## Methods

### ConvertToANSI381 Method

Converts a fingerprint sample to an image in ANSI 381 format.

```
Dim aAnsi As Byte()  
aAnsi = object.ConvertToANSI381(  
    ByVal oSample As Object)
```

## Parameters

<b>oSample</b>	A <b>DPFPSTSample</b> object (page 51)
<b>vAnsi</b>	<b>Variant</b> that receives an image in ANSI 381 format

## ConvertToPicture Method

Converts a fingerprint sample to an **IPicture** object.

### Syntax

```
Dim oPicture As IPictureDisp
Set oPicture = object.ConvertToPicture(
    ByVal oSample As Object)
```

### Parameters

<b>oSample</b>	A <b>DPFPSTemplate</b> object ( <i>page 51</i> )
<b>oPicture</b>	An <b>IPicture</b> object

### Object Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPSTemplate

The **DPFPSTemplate** object represents a fingerprint template.

### Methods and Properties

None.

### Object Information

Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## DPFPVerification

The **DPFPVerification** object performs the system function of *fingerprint verification*, which is a one-to-one comparison of a fingerprint feature set with a fingerprint template produced at enrollment that returns a decision of match or non-match.

## Method

### Verify Method

Performs the system function of fingerprint verification and specifies a comparison decision based on the requested FAR set by the **FARRequested** property.

#### Syntax

```
Dim oVerificationResult As DPFPVerificationResult
Set oVerificationResult = object.Verify(
    ByVal oVerificationFeatureSet As Object,
    ByVal oFingerprintTemplate As Object)
```

#### Parameters

<b>oFeatureSet</b>	A <b>DPFPFeatureSet</b> object, where the <b>enumPurpose</b> parameter of the <b>CreateFeatureSet</b> method of the <b>DPFPFeatureExtraction</b> object was set to the value <b>FeatureSetPurposeVerification</b> (page 44)
<b>oTemplate</b>	A <b>DPFPTemplate</b> object (page 52)
<b>oVerificationResult</b>	A <b>DPFPVerificationResult</b> object (page 56)

## Properties

### FARRequested Property

Gets or sets the requested false accept rate (FAR). For more information about FAR, see *False Positives and False Negatives* on page 18.

**IMPORTANT:** Although the default value is adequate for most applications, you might require a lower or higher value to meet your needs. If you decide to use a value other than the default, be sure that you understand the consequences of doing so. Refer to Appendix A on page 120 for more information about setting the value of the FAR.

#### Syntax

```
DPFPVerification.FARRequested [ = lValue ]
[ lValue = ] DPFPVerification.FARRequested
```

#### Possible Values

<b>lValue</b>	<b>Long</b> that receives the value of the requested FAR
---------------	--

This optional property is read/write. If you do not set it, the default value is used. You can use the **FARRequested** property accessor function to determine the current default value.

### Possible Errors

Error Code	Message	Description
-2147352566	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

### Object Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

### See Also

**DPFPVerificationResult** on page 56

## DPFPVerificationControl

The **DPFPVerificationControl** object is an ActiveX control that implements a user interface (described in *DPFPEnrollmentControl Object User Interface on page 107*) and provides the following functionality:

- Receives fingerprint reader connect and disconnect event notifications
- Captures a fingerprint sample from a fingerprint reader
- Creates a fingerprint feature set for the purpose of verification
- Fires an event

### Property

#### ReaderSerialNumber Property

Gets or sets the serial number of the fingerprint reader from which a fingerprint sample is captured.

#### Syntax

```
DPFPVerificationControl.ReaderSerialNumber [ = bstrValue ]  
[ bstrValue = ] DPFPVerificationControl.ReaderSerialNumber
```

#### Possible Values

<b>bstrValue</b>	<b>String</b> that receives the fingerprint reader serial number
------------------	--

This optional value is read/write. If you do not set it, the following value is used: `{00000000-0000-0000-0000-000000000000}`. This means that the application will receive events from any of the fingerprint readers attached to the system.

### Possible Errors

Error Code	Message	Description
-2147024809	One or more arguments are invalid.	The format of the string containing the fingerprint reader serial number is incorrect. It should be in GUID format, for example, {A9EFB3F6-A8C8-4684-841E-4330973057C6}.

### Object Information

Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPctIX.dll

### Event

#### OnComplete Event

Fires when a fingerprint feature set created for the purpose of verification is ready for comparison and returns the fingerprint feature set. The application handles the comparison of the fingerprint feature set with a fingerprint template(s).

#### Syntax

```
Private Sub object_OnComplete(  
    ByVal oVerificationFeatureSet As Object,  
    ByVal oStatus As Object)
```

#### Parameters

<b>oVerificationFeatureSet</b>	A <b>DPFPFeatureSet</b> object, which represents a fingerprint feature set created for the purpose of verification ( <i>page 45</i> )
<b>oStatus</b>	A <b>DPFPEventHandlerStatus</b> object ( <i>page 43</i> )

### Object Information

Type library
Library

# DPFPVerificationResult

The `DPFPVerificationResult` object represents the results of a fingerprint verification operation.

## Properties

### FARAchieved Property

Gets the value of the achieved FAR for a comparison operation.

#### Syntax

```
[ lValue = ] DPFPVerificationResult.FARAchieved
```

#### Possible Values

<b>lValue</b>	<b>Long</b> that receives the value of the FAR that was achieved for the comparison
---------------	---

This property is read-only and has no default value. See *Achieved FAR* on *page 122* for more information about this property.

### Verified Property

Gets the comparison decision, which indicates whether the comparison of a fingerprint feature set and a fingerprint template resulted in a decision of match or non-match. This decision is based on the value of the **FARRequested** property of the `DPFPVerification` object (*page 53*).

#### Syntax

```
[ vbValue = ] DPFPVerificationResult.Verified
```

#### Possible Values

<b>vbValue</b>	<b>Variant</b> of type <b>boolean</b> that receives the comparison decision. Possible values are true for a decision of match or false for a decision of non-match.
----------------	---

This property is read-only and has no default value.

## Object Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll



## Enumerations

The One Touch for Windows: COM/ActiveX Edition API COM implementation includes the enumerations defined in this section. Use the following list to quickly locate an enumeration by name, by page number, or by description.

Method	Page	Description
<b>DPFPCaptureFeedbackEnum</b>	58	Events returned by a fingerprint reader that provide feedback about a fingerprint sample capture operation
<b>DPFPCapturePriorityEnum</b>	59	Priority of a fingerprint sample capture operation
<b>DPFPEventHandlerStatusEnum</b>	60	Codes that are returned by the <b>DPFPEventHandlerStatus</b> object to indicate the status of an operation
<b>DPFPDataPurposeEnum</b>	61	Purpose for which a fingerprint feature set is to be used
<b>DPFPReaderImpressionTypeEnum</b>	62	Modality that a fingerprint reader uses to capture fingerprint samples
<b>DPFPReaderTechnologyEnum</b>	62	Fingerprint reader technology
<b>DPFPSerialNumberTypeEnum</b>	63	Fingerprint reader serial number persistence after reboot
<b>DPFPTemplateStatusEnum</b>	64	Status of a fingerprint template creation operation

## DPFPCaptureFeedbackEnum Enumeration

The `DPFPCaptureFeedbackEnum` enumeration defines the events returned by a fingerprint reader that provide feedback about a fingerprint sample capture operation.

### Syntax

```
Enum DPFPCaptureFeedbackEnum{
    CaptureFeedbackGood = 0,
    CaptureFeedbackNone = 1,
    CaptureFeedbackTooLight = 2,
    CaptureFeedbackTooDark = 3,
    CaptureFeedbackTooNoisy = 4,
    CaptureFeedbackLowContrast = 5,
    CaptureFeedbackNotEnoughFtrs = 6,
    CaptureFeedbackNoCentralRgn = 7,
    CaptureFeedbackNoFinger = 8,
    CaptureFeedbackTooHigh = 9,
    CaptureFeedbackTooLow = 10,
    CaptureFeedbackTooLeft = 11,
    CaptureFeedbackTooRight = 12,
    CaptureFeedbackTooStrange = 13,
    CaptureFeedbackTooFast = 14,
    CaptureFeedbackTooSkewed = 15,
    CaptureFeedbackTooShort = 16,
    CaptureFeedbackTooSlow = 17,
End Enum
```

### Constants

<code>CaptureFeedbackGood</code>	The fingerprint sample is of good quality.
<code>CaptureFeedbackNone</code>	There is no fingerprint sample.
<code>CaptureFeedbackTooLight</code>	The fingerprint sample is too light.
<code>CaptureFeedbackTooDark</code>	The fingerprint sample is too dark
<code>CaptureFeedbackTooNoisy</code>	The fingerprint sample is too noisy.
<code>CaptureFeedbackLowContrast</code>	The fingerprint sample contrast is too low.
<code>CaptureFeedbackNotEnoughFtrs</code>	The fingerprint sample does not contain enough information.
<code>CaptureFeedbackNoCentralRgn</code>	The fingerprint sample is not centered.

<b>CaptureFeedbackNoFinger</b>	The scanned object is not a finger.
<b>CaptureFeedbackTooHigh</b>	The finger was too high on the swipe sensor.
<b>CaptureFeedbackTooLow</b>	The finger was too low on the swipe sensor.
<b>CaptureFeedbackTooLeft</b>	The finger was too close to the left border of the swipe sensor.
<b>CaptureFeedbackTooRight</b>	The finger was too close to the right border of the swipe sensor.
<b>CaptureFeedbackTooStrange</b>	The scan looks strange.
<b>CaptureFeedbackTooFast</b>	The finger was swiped too quickly.
<b>CaptureFeedbackTooSkewed</b>	The fingerprint sample is too skewed.
<b>CaptureFeedbackTooShort</b>	The fingerprint sample is too short.
<b>CaptureFeedbackTooSlow</b>	The finger was swiped too slowly.

## Remarks

The members of this enumeration are called by the **CreateFeatureSet** method of the **DPFPFeatureExtraction** object (page 44) and by the **OnSampleQuality** event of the **DPFPCapture** object (page 35).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## DPFPCapturePriorityEnum Enumeration

The **DPFPCapturePriorityEnum** enumeration defines the priority of a fingerprint sample capture operation performed by a fingerprint reader.

## Syntax

```
Enum DPFPCapturePriorityEnum{
    CapturePriorityLow = 0,
    CapturePriorityNormal = 1,
    CapturePriorityHigh = 2,
End Enum
```

## Constants

<b>CapturePriorityLow</b>	Low priority. An application uses this priority to acquire events from the fingerprint reader only if there are no subscribers with high or normal priority. Only one subscriber with this priority is allowed.
<b>CapturePriorityNormal</b>	Normal priority. An application uses this priority to acquire events from the fingerprint reader only if the operation runs in a foreground process. Multiple subscribers with this priority are allowed.
<b>CapturePriorityHigh</b>	High priority. A subscriber uses this priority to acquire events from the fingerprint reader exclusively. Only one subscriber with this priority is allowed.

## Remarks

The members of this enumeration are called by the **Priority** property of the **DPFPCapture** object (page 32).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPEventHandlerStatusEnum Enumeration

The **DPFPEventHandlerStatusEnum** enumeration defines the codes that are returned by the **DPFPEventHandlerStatus** object to indicate the status of an operation.

## Syntax

```
Enum DPFPEventHandlerStatusEnum{
    EventHandlerStatusSuccess = 0,
    EventHandlerStatusFailure = 1,
End Enum
```

## Constants

<b>EventHandlerStatusSuccess</b>	An operation was performed successfully.
<b>EventHandlerStatusFailure</b>	An operation failed.

## Remarks

The members of this enumeration are called by the **Status** property of the **DPFPEventHandlerStatus** object (page 43).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPShrX.dll

## DPFPDataPurposeEnum Enumeration

The **DPFPDataPurposeEnum** enumeration defines the purpose for which a fingerprint feature set is to be used.

## Syntax

```
Enum DPFPDataPurposeEnum{
    DataPurposeUnknown = 0,
    DataPurposeVerification = 1,
    DataPurposeEnrollment = 2,
End Enum
```

## Constants

<b>DataPurposeUnknown</b>	The purpose is not known.
<b>DataPurposeVerification</b>	A fingerprint feature set to be used for the purpose of verification.
<b>DataPurposeEnrollment</b>	A fingerprint feature set to be used for the purpose of enrollment.

## Remarks

The members of this enumeration are called by the **CreateFeatureSet** method of the **DPFPFeatureExtraction** object (page 44).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## DPFPReaderImpressionTypeEnum Enumeration

The **DPFPReaderImpressionTypeEnum** enumeration defines the modality that a fingerprint reader uses to capture fingerprint samples.

### Syntax

```
Enum DPFPReaderImpressionTypeEnum{  
    ReaderImpressionTypeUnknown = 0,  
    ReaderImpressionTypeSwipe = 1,  
    ReaderImpressionTypeArea = 2,  
End Enum
```

### Constants

<b>ReaderImpressionTypeUnknown</b>	A fingerprint reader for which the modality is not known.
<b>ReaderImpressionTypeSwipe</b>	A swipe fingerprint reader.
<b>ReaderImpressionTypeArea</b>	An area (touch) sensor fingerprint reader.

### Remarks

The members of this enumeration are called by the **ImpressionType** property of the **DPFPReaderDescription** object (page 47).

### Enumeration Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPReaderTechnologyEnum Enumeration

The **DPFPReaderTechnologyEnum** enumeration defines the fingerprint reader technology.

### Syntax

```
Enum DPFPReaderTechnologyEnum{  
    ReaderTechnologyUnknown = 0,  
    ReaderTechnologyOptical = 1,  
    ReaderTechnologyCapacitive = 2,  
    ReaderTechnologyThermal = 3,  
    ReaderTechnologyPressure = 4,  
End Enum
```

## Constants

<b>ReaderTechnologyUnknown</b>	A fingerprint reader for which the technology is not known.
<b>ReaderTechnologyOptical</b>	An optical fingerprint reader.
<b>ReaderTechnologyCapacitive</b>	A capacitive fingerprint reader.
<b>ReaderTechnologyThermal</b>	A thermal fingerprint reader.
<b>ReaderTechnologyPressure</b>	A pressure fingerprint reader.

## Remarks

The members of this enumeration are called by the **Technology** property of the **DPFPReaderDescription** object (page 48).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPSerialNumberTypeEnum Enumeration

The **DPFPSerialNumberTypeEnum** enumeration defines whether a fingerprint reader serial number persists after reboot.

## Syntax

```
Enum DPFPSerialNumberTypeEnum{  
    SerialNumberTypePersistent = 0,  
    SerialNumberTypeVolatile = 1,  
End Enum
```

## Constants

<b>SerialNumberTypePersistent</b>	A persistent serial number.
<b>SerialNumberTypeVolatile</b>	A volatile serial number.

## Remarks

The members of this enumeration are called by the **SerialNumberType** property of the **DPFPReaderDescription** object (page 48).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPTemplateStatusEnum Enumeration

The **DPFPTemplateStatusEnum** enumeration defines the status of a fingerprint template creation operation.

### Syntax

```
Enum DPFPTemplateStatusEnum{
    TemplateStatusUnknown = 0,
    TemplateStatusInsufficient = 1,
    TemplateStatusFailed = 2,
    TemplateStatusReady = 3,
End Enum
```

### Constants

<b>TemplateStatusUnknown</b>	The status of a template creation operation is not know, probably because a fingerprint template does not exist yet.
<b>TemplateStatusInsufficient</b>	A fingerprint template exists, but more fingerprint feature sets are required to complete it.
<b>TemplateStatusFailed</b>	A fingerprint template creation operation failed.
<b>TemplateStatusReady</b>	A fingerprint template was created and is ready for use.

### Remarks

The members of this enumeration are called by the **TemplateStatus** property of the **DPFPEnrollment** object (*page 38*).

## Enumeration Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll



This chapter defines the interfaces (including methods, properties, and events) and the enumerations that are used for developing applications that incorporate the functionality of the One Touch for Windows: COM/ActiveX Edition API in C++ using the Component Object Model (COM) implementation.

## Interfaces

The One Touch for Windows: COM/ActiveX Edition API COM implementation includes the dual, nonextensible interfaces defined in this section. Use the following list to quickly locate an interface by name, by page number, or by description.

**IMPORTANT:** All of the read/write properties of the One Touch for Windows SDK API interfaces are optional. If you do not set one of these properties, the default value is automatically used. When deciding whether to set a property, be aware that DigitalPersona may change the default values at any time without notice. If you want your application's functionality to remain consistent, you should set the properties accordingly.

Interface	Page	Description
<b>IDPFPCapture</b>	67	Used by an application to capture a fingerprint sample from a fingerprint reader
<b>_IDPFPCaptureEvents</b>	69	Designates an event sink interface that an application must implement to receive event notifications from a <b>DPFPCapture</b> object
<b>IDPFPPData</b>	72	Represents the functionality of the data that is common to all fingerprint data objects
<b>IDPFPEnrollment</b>	73	Used by an application to perform the system function of fingerprint enrollment
<b>IDPFPEnrollmentControl</b>	76	Represents the functionality of an ActiveX control for creating and returning a fingerprint template or for deleting a fingerprint template, and provides a user interface
<b>_IDPFPEnrollmentControlEvents</b>	79	Designates an event sink interface that an application must implement to receive event notifications from a <b>DPFPEnrollmentControl</b> object
<b>IDPFPEventHandlerStatus</b>	81	Used by an application to retrieve codes that indicate the status of an operation
<b>IDPFPPFeatureExtraction</b>	82	Used by an application to perform the system function of fingerprint feature extraction

Interface	Page	Description
<b>IDPFPPFeatureSet</b>	84	Represents the functionality of a fingerprint feature set
<b>IDPFPPReaderDescription</b>	84	Used by an application to obtain information about a particular fingerprint reader connected to a system
<b>IDPFPPReadersCollection</b>	88	Represents a collection of fingerprint readers connected to a system
<b>IDPFPPSample</b>	91	Represents the functionality of a fingerprint sample
<b>IDPFPPSampleConversion</b>	91	Used by an application to convert a fingerprint sample to an image for display
<b>IDPFPPTemplate</b>	93	Represents the functionality of a fingerprint template
<b>IDPFPPVerification</b>	93	Used by an application to perform fingerprint verification
<b>IDPFPPVerificationControl</b>	95	Represents the functionality of an ActiveX control for creating and returning a fingerprint feature set, and provides a user interface
<b>_IDPFPPVerificationControlEvents</b>	97	Designates an event sink interface that an application must implement to receive event notifications from a <b>IDPFPPVerificationControl</b> object
<b>IDPFPPVerificationResult</b>	97	Represents the functionality of the results of a fingerprint verification operation

## IDPFPCapture Interface

Used by an application to capture a fingerprint sample from a fingerprint reader. The **IDPFPCapture** interface provides methods and properties for capturing a fingerprint sample from a fingerprint reader.

### IDPFPCapture Members

#### IDPFPCapture::Priority Property

Retrieves or returns a value that specifies the priority of a fingerprint sample capture operation.

This property is optional. If you do not set it, the value **DPFPCapturePriorityNormal** is used.

#### Syntax

```
HRESULT IDPFPCapture::get_Priority(  
    [out, retval] DPFPCapturePriorityEnum* pVal  
);  
  
HRESULT IDPFPCapture::put_Priority(  
    [in] DPFPCapturePriorityEnum newVal  
);
```

#### Parameters

<b>pVal</b>	[out, retval] Pointer to a variable that receives a value that specifies the priority of a fingerprint reader sample capture operation. For possible values, see <b>DPFPCapturePriorityEnum Enumerated Type</b> on page 101.
<b>newVal</b>	[in] Variable that contains the value that specifies the priority of a fingerprint reader sample capture operation

#### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_OVERFLOW</b>	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

## IDPFPCapture::ReaderSerialNumber Property

Retrieves or returns the serial number of a fingerprint reader that captures a fingerprint sample.

This property is optional. If you do not set it, the following value is used:

{00000000-0000-0000-0000-000000000000}. This means that the application will receive events from any of the fingerprint readers attached to the system.

### Syntax

```
HRESULT IDPFPCapture::get_ReaderSerialNumber(  
    [out, retval] BSTR* pVal  
);  
  
HRESULT IDPFPCapture::put_ReaderSerialNumber(  
    [in] BSTR newVal  
);
```

### Parameters

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>BSTR</b> that receives a fingerprint reader serial number
<b>newVal</b>	[in] Variable of type <b>BSTR</b> that contains the fingerprint reader serial number

### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>E_INVALIDARG</b>	One or more arguments are invalid.	The format of the string containing the fingerprint reader serial number is incorrect. It should be in GUID format, for example, {A9EFB3F6-A8C8-4684-841E-4330973057C6}.

## IDPFPCapture::StartCapture Method

Begins capturing a fingerprint sample from a fingerprint reader. A call to this method is asynchronous and returns immediately. The application continues to receive events until the **IDPFPCapture::StopCapture** method is called or when the **IDPFPCapture** object is destroyed.

### Syntax

```
HRESULT StartCapture(void);
```

## Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>E_INVALIDARG</b>	One or more arguments are invalid.	A capture operation with the specified priority already exists. See <b>DPFPCapturePriorityEnum</b> on <i>page 101</i> for more information.
<b>E_ACCESSDENIED</b>	General access denied error.	The application does not have sufficient privileges to start capture operations with the specified priority. See <b>DPFPCapturePriorityEnum</b> on <i>page 101</i> for more information.

## IDPFPCapture::StopCapture Method

Stops the fingerprint sample capture operation started with a call to the **IDPFPCapture::StartCapture** method.

### Syntax

```
HRESULT StopCapture(void);
```

### Return Value

Returns **S\_OK** if successful.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## \_IDPFPCaptureEvents Interface

Designates an event sink interface that an application must implement to receive event notifications from a **DPFPCapture** object, which implements the **IDPFPCapture** interface (*page 67*).

## \_IDPFPCaptureEvents Members

### \_IDPFPCaptureEvents::OnComplete Event

Fires when a fingerprint sample is successfully captured by a fingerprint reader.

#### Syntax

```
HRESULT OnComplete(
    [in] BSTR ReaderSerNum,
    [in] IDispatch* pFingerprintSample
);
```

#### Parameters

<b>ReaderSerNum</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
<b>pFingerprintSample</b>	[in] A <b>DPFPsample</b> object

### \_IDPFPCaptureEvents::OnFingerGone Event

Fires when a user removes a finger from a fingerprint reader.

#### Syntax

```
HRESULT OnFingerGone(
    [in] BSTR ReaderSerNum
);
```

#### Parameter

<b>ReaderSerNum</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
---------------------	--

### \_IDPFPCaptureEvents::OnFingerTouch Event

Fires when a user touches a fingerprint reader.

#### Syntax

```
HRESULT OnFingerTouch(
    [in] BSTR ReaderSerNum
);
```

#### Parameter

<b>ReaderSerNum</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
---------------------	--

**\_IDPFPCaptureEvents::OnReaderConnect Event**

Fires when a fingerprint reader is attached to a system.

**Syntax**

```
HRESULT OnReaderConnect(
    [in] BSTR ReaderSerNum
);
```

**Parameter**


---

<b>ReaderSerNum</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
---------------------	--

---

**\_IDPFPCaptureEvents::OnReaderDisconnect Event**

Fires when a fingerprint reader is disconnected from a system.

**Syntax**

```
HRESULT OnReaderDisconnect(
    [in] BSTR ReaderSerNum
);
```

**Parameter**


---

<b>ReaderSerNum</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
---------------------	--

---

**\_IDPFPCaptureEvents::OnSampleQuality Event**

Fires when the quality of a fingerprint sample is verified. When **SampleQualityGood** is returned by this event, the **\_IDPFPCaptureEvents::OnComplete** event is fired (*page 70*).

**Syntax**

```
HRESULT OnSampleQuality(
    [in] BSTR ReaderSerNum,
    [in] DPFPCaptureFeedbackEnum SampleQuality
);
```

**Parameters**


---

<b>ReaderSerNum</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
<b>SampleQuality</b>	[in] Variable that contains a value that provides feedback about a fingerprint sample capture operation. For possible values, see <b>DPFPCaptureFeedbackEnum Enumerated Type</b> on <i>page 100</i> .

---

## IDPFPPData Interface

Represents the functionality of the data that is common to all *fingerprint data objects*. The **IDPFPPData** interface also provides methods to serialize and deserialize the fingerprint data objects.

### IDPFPPData Members

#### IDPFPPData::Deserialize Method

Deserializes a fingerprint data object returned by the **IDPFPPData::Serialize** method.

##### Syntax

```
HRESULT Deserialize(
    [in] VARIANT RawData
);
```

##### Parameter

<b>RawData</b>	[in] <b>Variant array of bytes</b> ( <b>VT_U1</b> or <b>VT_ARRAY</b> ) that contains a deserialized fingerprint data object
----------------	---

##### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>E_INVALIDARG</b>	One or more arguments are invalid.	The format of the data passed to the <b>Deserialize</b> method is incorrect.
<b>S_FALSE</b>		Feature sets cannot be added because the fingerprint template has already been created.

#### IDPFPPData::Serialize Method

Serializes a fingerprint data object and returns it as an array of bytes.

##### Syntax

```
HRESULT Serialize(
    [out, retval] VARIANT* pRawData
);
```



### Parameter

<b>pRawData</b>	[out, retval] Pointer to a <b>variant array of bytes</b> ( <b>VT_U1</b> or <b>VT_ARRAY</b> ) that receives a serialized fingerprint data object
-----------------	---

### Return Value

Returns **S\_OK** if successful.

### Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

### See Also

**IDPFFeatureSet** Interface on page 84

**IDPFPSample** Interface on page 91

**IDPFPTemplate** Interface on page 93

## IDPFPEnrollment Interface

Used by an application to perform the system function of *fingerprint enrollment*. The **IDPFPEnrollment** interface provides methods and properties for creating a fingerprint template from a specified number of fingerprint feature sets created for the purpose of enrollment.

### IDPFPEnrollment Members

#### IDPFPEnrollment::AddFeatures Method

Adds fingerprint feature sets, one-by-one, to a fingerprint template. A call to this method creates an instance of **DPFPPTemplate**, which represents a fingerprint template. The **DPFPPTemplate** object implements the **IDPFPTemplate** interface (page 93). The fingerprint template is complete when the **TemplateStatus** property is set to the value **TemplateStatusReady**.

#### Syntax

```
HRESULT AddFeatures(  
    [in] IDispatch* pVal  
);
```

**Parameter**


---

<b>pVal</b>	[in] A <b>DPFPFeatureSet</b> object
-------------	-------------------------------------

---

**Return Value**

Returns **S\_OK** if successful.

**IDPFPEnrollment::Clear Method**

Clears a fingerprint template and sets the value of the **TemplateStatus** property to **TemplateStatusUnknown** so an application can begin another fingerprint template creation operation.

**Syntax**

```
HRESULT Clear(void);
```

**Return Value**

Returns **S\_OK** if successful.

**IDPFPEnrollment::FeaturesNeeded Property**

Retrieves the number of fingerprint feature sets still needed to create a fingerprint template. When the value of the **pVal** parameter is equal to 0, the fingerprint template is created. This property is read-only and has no default value.

**Syntax**

```
HRESULT IDPFPEnrollment::get_FeaturesNeeded(
    [out, retval] LONG* pVal
);
```

**Parameter**


---

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>long</b> that receives the value of the number of fingerprint feature sets
-------------	---

---

**Return Value**

Returns **S\_OK** if successful.

**IDPFPEnrollment::Template Property**

Retrieves a **DPFPTemplate** object created during a fingerprint enrollment operation. This property is read-only and has no default value.

Syntax

```
HRESULT IDPFPEnrollment::get_Template(  
    [out, retval] IDispatch** pVal  
);
```

Parameter

pVal	[out, retval] A <b>DPFPTemplate</b> object
------	--

Return Value

Returns **S\_OK** if successful.

IDPFPEnrollment::TemplateStatus Property

Retrieves a value that specifies the status of a fingerprint template creation operation. This property is read-only and has no default value.

Syntax

```
HRESULT IDPFPEnrollment::get_TemplateStatus(  
    [out, retval] DPFPTemplateStatusEnum* pVal  
);
```

Parameter

pVal	[out, retval] Pointer to a variable that receives a value that specifies the status of the fingerprint template creation operation. For possible values, see <b>DPFPTemplateStatusEnum Enumerated Type</b> on page 106.
------	---

Return Value

Returns **S\_OK** if successful.

Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

# IDPFPEnrollmentControl Interface

Represents the functionality of an ActiveX control, which implements a user interface (described in *DPFPEnrollmentControl Object User Interface on page 107*). The `IDPFPEnrollmentControl` interface provides the following functionality:

- Captures a fingerprint sample from a fingerprint reader
- Creates a specified number of fingerprint feature sets for the purpose of enrollment
- Creates a fingerprint template
- Deletes a fingerprint template
- Fires events

## IDPFPEnrollmentControl Members

### IDPFPEnrollmentControl::EnrolledFingersMask Property

Retrieves or returns the mask representing the user’s enrolled fingerprints. The enrollment mask is a combination of the values representing a user’s enrolled fingerprints. For example, if a user’s right index fingerprint and right middle fingerprint are enrolled, the value of this property is 00000000 011000000, or 192.

This property is optional. If you do not set it, the value `0` is used, which means that no fingerprints have been enrolled.

#### Syntax

```
HRESULT IDPFPEnrollmentControl::get_EnrolledFingersMask (
    [out, retval] LONG* pVal
);

HRESULT IDPFPEnrollmentControl::put_EnrolledFingersMask (
    [in] LONG newVal
);
```

#### Parameters

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>long</b> that receives the value of the fingerprint mask
<b>newVal</b>	[in] Variable of type <b>long</b> that contains the value of the fingerprint mask

## Possible Values

All possible values for the enrollment mask are listed in *Table 5*.

**Table 5.** Values for the enrollment mask

Finger	Binary Representation	Integer Representation
Left little finger	000000000 000000001	1
Left ring finger	000000000 000000010	2
Left middle finger	000000000 000000100	4
Left index finger	000000000 000001000	8
Left thumb	000000000 000010000	16
Right thumb	000000000 000100000	32
Right index finger	000000000 001000000	64
Right middle finger	000000000 010000000	128
Right ring finger	000000000 100000000	256
Right little finger	000000001 000000000	512

## Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_OVERFLOW</b>	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

## IDPFPEnrollmentControl::MaxEnrollFingerCount Property

Retrieves or returns the value for the maximum number of fingerprints that can be enrolled. Possible values for this parameter are **1** through **10**.

This property is optional. If you do not set it, the value **10** is used, which means the user can enroll all ten fingerprints.

### Syntax

```
HRESULT IDPFPEnrollmentControl::get_MaxEnrollFingerCount(  
    [out, retval] LONG* pVal  
);  
  
HRESULT IDPFPEnrollmentControl::put_MaxEnrollFingerCount(  
    [in] LONG newVal  
);
```

### Parameters

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>long</b> that receives the value for the maximum number of fingerprints that can be enrolled
<b>newVal</b>	[in] Variable of type <b>long</b> that contains the value for the maximum number of fingerprints that can be enrolled

### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_OVERFLOW</b>	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

### IDPFPEnrollmentControl::ReaderSerialNumber Property

Retrieves or returns the serial number of the fingerprint reader from which a fingerprint sample is captured.

This property is optional. If you do not set it, the following value is used:

{00000000-0000-0000-0000-000000000000}. This means that the application will receive events from any of the fingerprint readers attached to the system.

### Syntax

```
HRESULT IDPFPEnrollmentControl::get_ReaderSerialNumber(  
    [out, retval] BSTR* pVal  
);  
  
HRESULT IDPFPEnrollmentControl::put_ReaderSerialNumber(  
    [in] BSTR newVal  
);
```

## Parameters

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>BSTR</b> that receives the fingerprint reader serial number
<b>newVal</b>	[in] Variable of type <b>BSTR</b> that contains the fingerprint reader serial number

## Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>E_INVALIDARG</b>	One or more arguments are invalid.	The format of the string containing the fingerprint reader serial number is incorrect. It should be in GUID format, for example, {A9EFB3F6-A8C8-4684-841E-4330973057C6}.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPCTX.dll

## \_IDPFPEnrollmentControlEvents Interface

Designates an event sink interface that an application must implement to receive event notifications from a **DPFPEnrollmentControl** object, which implements the **IDPFPEnrollmentControlEvents** interface (page 76).

## \_IDPFPEnrollmentControlEvents Members

### \_IDPFPEnrollmentControlEvents::OnDelete Event

Fires when a user deletes a finger. The application handles the deletion of the fingerprint template from a fingerprint data storage subsystem and can display its own success or error messages.

## Syntax

```
HRESULT OnDelete(
    [in] LONG lFingerMask,
    [in] IDispatch* pStatus
);
```

## Parameters

<b>lFingerMask</b>	[in] Pointer to a variable of type <b>long</b> that contains the index value of the (enrolled) fingerprint to be deleted. For possible values, see <i>Table 6</i> .
<b>pStatus</b>	[in] A <b>DPFPEventHandlerStatus</b> object

The **uFingerprintMask** parameter is the index value of the finger associated with a fingerprint to be enrolled or a fingerprint template to be deleted, as defined in ANSI/NIST-ITL 1. The index values are assigned to the graphical representation of the fingers on the hands in the user interface. All possible values are listed in *Table 6*.

**Table 6.** Finger index values in ANSI/NIST-ITL 1

Finger	Index Value	Finger	Index Value
Right thumb	1	Left thumb	6
Right index finger	2	Left index finger	7
Right middle finger	3	Left middle finger	8
Right index finger	4	Left ring finger	9
Right little finger	5	Left little finger	10

## \_IDPFPEnrollmentControlEvents::OnEnroll Event

Fires when a user enrolls a fingerprint and returns a fingerprint template. The application handles the storage of the fingerprint template in a fingerprint data storage subsystem and can display its own success or error messages.

## Syntax

```
HRESULT OnEnroll(
    [in] LONG lFingerMask,
    [in] IDispatch* pFingerprintTemplate,
    [in] IDispatch* pStatus
);
```



## Parameters

<b>lFingerMask</b>	[in] Variable of type <b>long</b> that contains the index value for the enrolled fingerprint. For possible values, see Table 6 on <i>page 80</i> .
<b>pFingerprintTemplate</b>	[in] A <b>DPFPFTemplate</b> object
<b>pStatus</b>	[in] A <b>DPFPEventHandlerStatus</b> object

## IDPFPEventHandlerStatus Interface

Used by an application to retrieve codes that indicate the status of an operation.

### IDPFPEventHandlerStatus Member

#### IDPFPEventHandlerStatus::Status Property

Retrieves or returns the status of an operation performed by a **DPFPFEnrollmentControl** object, which implements the **IDPFPEnrollmentControl** interface (*page 76*), or a **DPFPVerificationControl** object, which implements the **IDPFVerificationControl** interface (*page 95*).

This property is optional. If you do not set it, the value **DPFPEventHandlerStatusSuccess** is used.

#### Syntax

```
HRESULT IDPFPEventHandlerStatus::get_Status(
    [out, retval] DPFPEventHandlerStatusEnum* pVal
);

HRESULT IDPFPEventHandlerStatus::put_Status(
    [in] DPFPEventHandlerStatusEnum newVal
);
```

## Parameters

<b>pVal</b>	[out, retval] Pointer to a variable that receives a value that indicates the status of an operation. For possible values, see <b>DPFPEventHandlerStatusEnum Enumerated Type</b> on <i>page 102</i> .
<b>newVal</b>	[in] Variable that contains the value that indicates the status of an operation

## Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_OVERFLOW</b>	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPCTX.dll

## IDPFPPFeatureExtraction Interface

Used by an application to perform *fingerprint feature extraction*. The **IDPFPPFeatureExtraction** interface provides a method and a property for creating a fingerprint feature set for the purpose of enrollment or verification by applying fingerprint feature extraction to a fingerprint sample.

## IDPFPPFeatureExtraction Members

### IDPFPPFeatureExtraction::CreateFeatureSet Method

Applies fingerprint feature extraction to a fingerprint sample and then creates a fingerprint feature set for the specified purpose. A call to this method creates an instance of **DPFPFeatureSet**, which represents a fingerprint feature set. The **DPFPFeatureSet** object implements the **IDPFPPFeatureSet** interface (page 84).

### Syntax

```
HRESULT CreateFeatureSet(
    [in] IDispatch* pFingerprintSample,
    [in] DPFPDataPurposeEnum Purpose,
    [out, retval] DPFPCaptureFeedbackEnum* pSampleQuality
);
```

## Parameters

<b>pFingerprintSample</b>	[in] A <b>DPFPSTSample</b> object
<b>Purpose</b>	[in] Variable that contains a value for the specified purpose. For possible values, see <b>DPFPDataPurposeEnum Enumerated Type</b> on page 103.
<b>pSampleQuality</b>	[out, retval] Pointer to a variable that receives a value that provides feedback about a fingerprint sample capture operation. For possible values, see <b>DPFPCaptureFeedbackEnum Enumerated Type</b> on page 100.

## Return Value

Returns **S\_OK** if successful.

## IDPFPPFeatureExtraction::FeatureSet Property

Retrieves a **DPFPFeatureSet** object created during a fingerprint feature extraction operation. This property is read-only and has no default value.

## Syntax

```
HRESULT IDPFPPFeatureExtraction::get_FeatureSet(
    [out, retval] IDispatch** pVal
);
```

## Parameter

<b>pVal</b>	[out, retval] A <b>DPFPFeatureSet</b> object
-------------	--

## Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_MEMBERNOTFOUND</b>	Member not found.	A fingerprint feature set has not been created yet.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## IDPFPPFeatureSet Interface

Represents the functionality of a fingerprint feature set. A **DPFPFeatureSet** object, which represents a fingerprint feature set, implements the **IDPFPPFeatureSet** interface.

### IDPFPPFeatureSet Members

None.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDPFPPData</b>
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## IDPFPPReaderDescription Interface

Used by an application to obtain information about a particular fingerprint reader connected to a system, such as its technology or serial number.

### IDPFPPReaderDescription Members

#### IDPFPPReaderDescription::FirmwareRevision Property

Retrieves the firmware revision number of a fingerprint reader. This property is read-only and has no default value.

#### Syntax

```
HRESULT IDPFPPReaderDescription::get_FirmwareRevision(
    [out, retval] BSTR* pVal
);
```

### Parameter

---

<b>pVal</b>	[in] Pointer to a variable of type <b>BSTR</b> that receives the fingerprint reader firmware revision number
-------------	--

---

### Return Value

Returns **S\_OK** if successful.

### IDPFPRReaderDescription::HardwareRevision Property

Retrieves the hardware revision number of a fingerprint reader. This property is read-only and has no default value.

### Syntax

```
HRESULT IDPFPRReaderDescription::get_HardwareRevision(  
    [out, retval] BSTR* pVal  
);
```

### Parameter

---

<b>pVal</b>	[in] Pointer to a variable of type <b>BSTR</b> that receives the fingerprint reader hardware revision number
-------------	--

---

### IDPFPRReaderDescription::Language Property

Retrieves the fingerprint reader language. The value of the **pVal** parameter is always 0x409, which is English. This property is read-only and has no default value.

### Syntax

```
HRESULT IDPFPRReaderDescription::get_Language(  
    [out, retval] LONG* pVal  
);
```

### Parameter

---

<b>pVal</b>	[in] Pointer to a variable of type <b>BSTR</b> that receives the fingerprint reader language
-------------	--

---

### Return Value

Returns **S\_OK** if successful.

### IDPFPRReaderDescription::ImpressionType Property

Retrieves a value that specifies the fingerprint reader impression type, for example, swipe reader or touch (area) reader. This property is read-only and has no default value.

#### Syntax

```
HRESULT IDPFPRReaderDescription::get_ImpressionType(  
    [out, retval] DPFPReaderImpressionTypeEnum* pVal  
);
```

#### Parameter

---

<b>pVal</b>	[in] Pointer to a variable that receives a value that specifies the fingerprint reader modality. For possible values, see <b>DPFPReaderImpressionTypeEnum Enumerated Type</b> on page 104.
-------------	--

---

#### Return Value

Returns **S\_OK** if successful.

### IDPFPRReaderDescription::ProductName Property

Retrieves the product name of a fingerprint reader, for example, "U.are.U." This property is read-only and has no default value.

#### Syntax

```
HRESULT IDPFPRReaderDescription::get_ProductName(  
    [out, retval] BSTR* pVal  
);
```

#### Parameter

---

<b>pVal</b>	[in] Pointer to a variable of type <b>BSTR</b> that receives the fingerprint reader product name
-------------	--

---

#### Return Value

Returns **S\_OK** if successful.

### IDPFPRReaderDescription::SerialNumber Property

Retrieves the serial number of a fingerprint reader. This property is read-only and has no default value.

### Syntax

```
HRESULT IDPFPPReaderDescription::get_SerialNumber(  
    [out, retval] BSTR* pVal  
);
```

### Parameter

---

<b>pVal</b>	[in] Pointer to a variable of type <b>BSTR</b> the receives the fingerprint reader serial number
-------------	--

---

### Return Value

Returns **S\_OK** if successful.

### IDPFPPReaderDescription::SerialNumberType Property

Retrieves a value that specifies the type of fingerprint reader serial number. This property is read-only and has no default value.

### Syntax

```
HRESULT IDPFPPReaderDescription::get_SerialNumberType(  
    [out, retval] DPFPSerialNumberTypeEnum* pVal  
);
```

### Parameter

---

<b>pVal</b>	[in] Pointer to a variable that receives a value that specifies the fingerprint reader serial number type. For possible values, see <b>DPFPSerialNumberTypeEnum Enumerated Type</b> on page 105.
-------------	--

---

### Return Value

Returns **S\_OK** if successful.

### IDPFPPReaderDescription::Technology Property

Retrieves a value that specifies the fingerprint reader technology. This property is read-only and has no default value.

### Syntax

```
HRESULT IDPFPPReaderDescription::get_Technology(  
    [out, retval] DPFPReaderTechnologyEnum* pVal  
);
```

**Parameter**

<b>pVal</b>	[in] Pointer to a variable that receives a value that specifies the fingerprint reader technology. For possible values, see <b>DPFPReaderTechnologyEnum Enumerated Type</b> on page 104.
-------------	--

**Return Value**

Returns **S\_OK** if successful.

**IDPFPRReaderDescription::Vendor Property**

Retrieves the vendor name for a fingerprint reader, for example, "DigitalPersona, Inc." This property is read-only and has no default value.

**Syntax**

```
HRESULT IDPFPRReaderDescription::get_Vendor(
    [out, retval] BSTR* pVal
);
```

**Parameter**

<b>pVal</b>	[in] Pointer to a variable of type <b>BSTR</b> the receives the fingerprint reader vendor name
-------------	--

**Return Value**

Returns **S\_OK** if successful.

**Interface Information**

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

**IDPFPRReadersCollection Interface**

Represents a collection of fingerprint readers connected to a system. The **IDPFPRReadersCollection** interface provides a method and properties for enumerating the fingerprint readers, for retrieving a particular fingerprint reader using its index value or its serial number, and for reporting the total number of fingerprint readers.



## IDPFPPReadersCollection Members

### IDPFPPReadersCollection::Reader Method

Creates an instance of **DPFPReaderDescription** for a particular fingerprint reader using its serial number. The **DPFPReaderDescription** object implements the **IDPFPPReaderDescription** interface ([page 84](#)).

#### Syntax

```
HRESULT Reader (
    [in] BSTR ReaderSerialNum,
    [out,retval] IDispatch** ppReader
);
```

#### Parameters

<b>ReaderSerialNumber</b>	[in] Variable of type <b>BSTR</b> that contains a fingerprint reader serial number
<b>ppReader</b>	[out, retval] A <b>DPFPReaderDescription</b> object

#### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>_HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND)</b>	The system cannot find the specified file.	The fingerprint reader with the specified serial number cannot be found in the system.

### IDPFPPReadersCollection::Count Property

Retrieves the total number of **DPFPReaderDescription** objects (items) connected to a system (a collection). This property is read-only and has no default value.

#### Syntax

```
HRESULT IDPFPPReadersCollection::get_Count (
    [out,retval] LONG* pVal
);
```

#### Parameter

<b>pVal</b>	[in] Pointer to a variable of type <b>long</b> that receives the total number of <b>DPFPReaderDescription</b> objects
-------------	---

### Return Value

Returns **S\_OK** if successful.

### IDPFPPReadersCollection::Item Property

Retrieves a **DPFPReaderDescription** object (an item) from the fingerprint readers connected to a system (a collection) using its index. The value of the **pVal** parameter starts with **1**.

#### Syntax

```
HRESULT IDPFPPReadersCollection::get_Item(
    [out,retval] IDispatch** pVal
);
```

#### Parameter

<b>pVal</b>	[out, retval] A <b>DPFPReaderDescription</b> object
-------------	---

#### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_BADINDEX</b>	Invalid index.	The specified index is not in the valid range from <b>1</b> to <b>Count</b> .

### IDPFPPReadersCollection::\_NewEnum Property

Retrieves an **IUnknown** pointer to the **ReaderEnum** object (enumeration object), which is an array of **DPFPReaderDescription** objects. This property is read-only and has no default value.

#### Syntax

```
HRESULT IDPFPPReadersCollection::get__NewEnum(
    [out,retval] IUnknown** pVal
);
```

#### Parameter

<b>pVal</b>	[in] Pointer to a variable of type <b>IUnknown</b> that receives the array of <b>DPFPReaderDescription</b> objects
-------------	--

#### Return Value

Returns **S\_OK** if successful.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## IDPFPSample Interface

Represents the functionality of a fingerprint sample captured from a fingerprint reader.

### IDPFPSample Members

None.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDPFPSData</b>
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

### See Also

**IDPFPSData Interface** on page 72

## IDPFPSampleConversion Interface

Used by an application to convert a fingerprint sample to an image for display. The **IDPFPSampleConversion** interface provides methods for returning a fingerprint sample as an **IPicture** object and as an image in ANSI 381 format that can be used for display.

## IDPFPSampleConversion Members

### IDPFPSample::ConvertToANSI381 Method

Converts a fingerprint sample to an image in ANSI 381 format that can be used by an application for display.

```
HRESULT ConvertToANSI381(
    [in] IDispatch* pSample,
    [out,retval] VARIANT* pAnsi
);
```

#### Parameters

<b>pSample</b>	[in] A <b>IDPFPSample</b> object
<b>pAnsi</b>	[out, retval] Pointer to a <b>variant array of bytes</b> ( <b>VT_U1</b> or <b>VT_ARRAY</b> ) that receives an image in ANSI 381 format

#### Return Value

Returns **S\_OK** if successful.

### IDPFPSample::ConvertToPicture Method

Converts a fingerprint sample to an **IPicture** object that can be used by an application as an image for display.

#### Syntax

```
HRESULT ConvertToPicture(
    [in] IDispatch* pSample,
    [out,retval] IDispatch** ppPicture
);
```

#### Parameters

<b>pSample</b>	[in] A <b>IDPFPSample</b> object
<b>ppPicture</b>	[out, retval] An <b>IPicture</b> object

#### Return Value

Returns **S\_OK** if successful.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## IDPFPTemplate Interface

Represents the functionality of a fingerprint template. A **DPFPTemplate** object, which represents a fingerprint template, implements the **IDPFPTemplate** interface.

### IDPFPTemplate Members

None.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDPFPPData</b>
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## IDFPFVerification Interface

Used by an application to perform the system function of *fingerprint verification*. The **IDFPFVerification** interface provides a method and a property for performing fingerprint verification, which is a one-to-one comparison of a fingerprint feature set with a fingerprint template produced at enrollment that returns a decision of match or non-match.

### IDFPFVerification Members

#### IDFPFVerification::FARRequested Property

Retrieves or returns the requested false accept rate (FAR). For a general explanation of FAR, see *False Positives and False Negatives* on page 18.

This property is optional. If you do not set it, the default value is used. You can use the **get\_FARRequested** property to determine the current value.

**IMPORTANT:** Although the default value is adequate for most applications, you might require a lower or higher value to meet your needs. If you decide to use a value other than the default, be sure that you understand the consequences of doing so. Refer to Appendix A on *page 120* for more information about setting the value of the FAR.

### Syntax

```
HRESULT IDFPFVerification::get_FARRequested(
    [out, retval] LONG* pVal
);

HRESULT IDFPFVerification::put_FARRequested(
    [in] LONG newVal
);
```

### Parameters

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>long</b> that receives the value of the requested FAR
<b>newVal</b>	[in] Variable of type <b>long</b> that contains the value of the requested FAR

### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>DISP_E_OVERFLOW</b>	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

### IDFPFVerification::Verify Method

Performs the system function of fingerprint verification and returns a comparison decision based on the requested FAR set by the **IDFPFVerification::FARRequested** property.

### Syntax

```
HRESULT Verify(
    [in] IDispatch* pVerificationFeatureSet,
    [in] IDispatch* pFingerprintTemplate,
    [out, retval] IDispatch** ppVerificationResult
);
```

## Parameters

<b>pFeatureSet</b>	[in] A <b>DPFPFeatureSet</b> object, where the <b>Purpose</b> parameter of the <b>IDPFPPFeatureExtraction::CreateFeatureSet</b> method was set to the value <b>FeatureSetPurposeVerification</b> ( <i>page 82</i> )
<b>pTemplate</b>	[in] A <b>DPFPTemplate</b> object
<b>ppVerificationResult</b>	[out, retval] A <b>DPFPVerificationResult</b> object

## Return Value

Returns **S\_OK** if successful.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## See Also

**IDPFPPVerificationResult** Interface on *page 97*

## IDPFPPVerificationControl Interface

Represents the functionality of an ActiveX control, which implements a user interface (described in *DPFPEnrollmentControl Object User Interface on page 107*). The **IDPFPPVerificationControl** interface provides the following functionality:

- Receives fingerprint reader connect and disconnect event notifications
- Captures a fingerprint sample from a fingerprint reader
- Creates a fingerprint feature set for the purpose of verification
- Fires an event

## IDPFVerificationControl Members

### IDPFVerificationControl::ReaderSerialNumber Property

Retrieves or returns the serial number of the fingerprint reader from which a fingerprint sample is captured.

This property is optional. If you do not set it, the following value is used:

{00000000-0000-0000-0000-000000000000}. This means that the application will receive events from any of the fingerprint readers attached to the system.

#### Syntax

```
HRESULT IDPFVerificationControl::get_ReaderSerialNumber (
    [out, retval] BSTR* pVal
);

HRESULT IDPFVerificationControl::put_ReaderSerialNumber (
    [in] BSTR newVal
);
```

#### Parameters

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>BSTR</b> that receives the fingerprint reader serial number
<b>newVal</b>	[in] Variable of type <b>BSTR</b> that contains the fingerprint reader serial number

#### Return Values

Returns **S\_OK** if successful, or the following error value otherwise:

Return Value	Message	Description
<b>E_INVALIDARG</b>	One or more arguments are invalid.	The format of the string containing the fingerprint reader serial number is incorrect. It should be in GUID format, for example, {A9EFB3F6-A8C8-4684-841E-4330973057C6}.

## Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPCtIX.dll



## \_IDFPFVerificationControlEvents Interface

Designates an event sink interface that an application must implement to receive event notifications from a **DPFPVerificationControl** object, which implements the **IDFPFVerificationControl** interface (page 95).

### \_IDFPFVerificationControlEvents Members

#### \_IDFPFVerificationControlEvents::OnComplete Event

Fires when a fingerprint feature set created for the purpose of verification is ready for comparison and returns the fingerprint feature set. The application handles the comparison of the fingerprint feature set with a fingerprint template(s).

#### Syntax

```
HRESULT OnComplete(
    [in] IDispatch* pVerificationFeatureSet,
    [in] IDispatch* pStatus
);
```

#### Parameters

<b>pVerificationFeatureSet</b>	[in] A <b>DPFPFeatureSet</b> object
<b>pStatus</b>	[in] A <b>DPFPEventHandlerStatus</b> object

#### Return Value

Returns **S\_OK** if successful.

## IDFPFVerificationResult Interface

Represents the functionality of the results of a fingerprint verification operation. A **DPFPVerificationResult** object, which represents the results of a fingerprint verification operation, implements the **IDFPFVerificationResult** interface. The **IDFPFVerificationResult** interface provides properties for retrieving the results of a fingerprint verification operation.

### IDFPFVerificationResult Members

#### IDFPFVerificationResult::FARAchieved Property

Retrieves the value of the achieved FAR for a comparison operation. This property is read-only and has no default value. See *Achieved FAR* on page 122 for more information about this property.

### Syntax

```
HRESULT IDPFPPVerificationResult::get_FARAchieved(
    [out, retval] LONG* pVal
);
```

### Parameter

<b>pVal</b>	[out, retval] Pointer to a variable of type <b>long</b> that receives the value of the FAR that was achieved for the comparison
-------------	---

### Return Value

Returns **S\_OK** if successful.

### IDPFPPVerificationResult::Verified Property

Retrieves the comparison decision, which indicates whether the comparison of a fingerprint feature set and a fingerprint template resulted in a decision of match or non-match. This decision is based on the value set by the **IDPFPPVerification::FARRequested** property (page 93). The **IDPFPPVerificationResult::Verified Property** property is read-only and has no default value.

### Syntax

```
HRESULT IDPFPPVerificationResult::get_Verified(
    [out, retval] VARIANT_BOOL* pVal
);
```

### Parameter

<b>pVal</b>	[out, retval] Pointer to a <b>variant</b> of type <b>boolean</b> that receives the comparison decision. Possible values are true for a decision of match or false for a decision of non-match.
-------------	--

### Return Value

Returns **S\_OK** if successful.

### Interface Information

Custom implementation	Yes
Inherits from	<b>IDispatch</b>
Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## Enumerations

The One Touch for Windows: COM/ActiveX Edition API COM implementation includes the enumerated types defined in this section. Use the following list to quickly locate an enumerated type by name, by page number, or by description.

Method	Page	Description
<b>DPFPCaptureFeedbackEnum</b>	100	Events returned by a fingerprint reader that provide feedback about a fingerprint sample capture operation
<b>DPFPCapturePriorityEnum</b>	101	Priority of a fingerprint sample capture operation
<b>DPFPEventHandlerStatusEnum</b>	102	Codes that are returned by the <b>DPFPEventHandlerStatus</b> object to indicate the status of an operation
<b>DPFPDataPurposeEnum</b>	103	Purpose for which a fingerprint feature set is to be used
<b>DPFPReaderImpressionTypeEnum</b>	104	Modality that a fingerprint reader uses to capture fingerprint samples
<b>DPFPReaderTechnologyEnum</b>	104	Fingerprint reader technology
<b>DPFPSerialNumberTypeEnum</b>	105	Fingerprint reader serial number persistence after reboot
<b>DPFPTemplateStatusEnum</b>	106	Status of a fingerprint template creation operation

## DPFPCaptureFeedbackEnum Enumerated Type

The `DPFPCaptureFeedbackEnum` enumerated type defines the events returned by a fingerprint reader that provide feedback about a fingerprint sample capture operation.

### Syntax

```
typedef enum DPFPCaptureFeedbackEnum{
    CaptureFeedbackGood = 0,
    CaptureFeedbackNone = 1,
    CaptureFeedbackTooLight = 2,
    CaptureFeedbackTooDark = 3,
    CaptureFeedbackTooNoisy = 4,
    CaptureFeedbackLowContrast = 5,
    CaptureFeedbackNotEnoughFtrs = 6,
    CaptureFeedbackNoCentralRgn = 7,
    CaptureFeedbackNoFinger = 8,
    CaptureFeedbackTooHigh = 9,
    CaptureFeedbackTooLow = 10,
    CaptureFeedbackTooLeft = 11,
    CaptureFeedbackTooRight = 12,
    CaptureFeedbackTooStrange = 13,
    CaptureFeedbackTooFast = 14,
    CaptureFeedbackTooSkewed = 15,
    CaptureFeedbackTooShort = 16,
    CaptureFeedbackTooSlow = 17,
} DPFPCaptureFeedbackEnum;
```

### Constants

<code>CaptureFeedbackGood</code>	The fingerprint sample is of good quality.
<code>CaptureFeedbackNone</code>	There is no fingerprint sample.
<code>CaptureFeedbackTooLight</code>	The fingerprint sample is too light.
<code>CaptureFeedbackTooDark</code>	The fingerprint sample is too dark
<code>CaptureFeedbackTooNoisy</code>	The fingerprint sample is too noisy.
<code>CaptureFeedbackLowContrast</code>	The fingerprint sample contrast is too low.
<code>CaptureFeedbackNotEnoughFtrs</code>	The fingerprint sample does not contain enough information.
<code>CaptureFeedbackNoCentralRgn</code>	The fingerprint sample is not centered.

<b>CaptureFeedbackNoFinger</b>	The scanned object is not a finger.
<b>CaptureFeedbackTooHigh</b>	The finger was too high on the swipe sensor.
<b>CaptureFeedbackTooLow</b>	The finger was too low on the swipe sensor.
<b>CaptureFeedbackTooLeft</b>	The finger was too close to the left border of the swipe sensor.
<b>CaptureFeedbackTooRight</b>	The finger was too close to the right border of the swipe sensor.
<b>CaptureFeedbackTooStrange</b>	The scan looks strange.
<b>CaptureFeedbackTooFast</b>	The finger was swiped too quickly.
<b>CaptureFeedbackTooSkewed</b>	The fingerprint sample is too skewed.
<b>CaptureFeedbackTooShort</b>	The fingerprint sample is too short.
<b>CaptureFeedbackTooSlow</b>	The finger was swiped too slowly.

## Remarks

The members of this enumerated type are called by the **IDPFPFFeatureExtraction::CreateFeatureSet** method (*page 82*) and by the **\_IDPFPCaptureEvents::OnSampleQuality** event (*page 71*).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

## DPFPCapturePriorityEnum Enumerated Type

The **DPFPCapturePriorityEnum** enumerated type defines the priority of a fingerprint sample capture operation performed by a fingerprint reader.

## Syntax

```
typedef enum DPFPCapturePriorityEnum{
    CapturePriorityLow = 0,
    CapturePriorityNormal = 1,
    CapturePriorityHigh = 2,
} DPFPCapturePriorityEnum;
```

## Constants

<b>CapturePriorityLow</b>	Low priority. An application uses this priority to acquire events from the fingerprint reader only if there are no subscribers with high or normal priority. Only one subscriber with this priority is allowed.
<b>CapturePriorityNormal</b>	Normal priority. An application uses this priority to acquire events from the fingerprint reader only if the operation runs in a foreground process. Multiple subscribers with this priority are allowed.
<b>CapturePriorityHigh</b>	High priority. A subscriber uses this priority to acquire events from the fingerprint reader exclusively. Only one subscriber with this priority is allowed.

## Remarks

The members of this enumerated type are called by the **IDPFPCapture::Priority** property (page 67).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPEventHandlerStatusEnum Enumerated Type

The **DPFPEventHandlerStatusEnum** enumerated type defines the codes that are returned by the **DPFPEventHandlerStatus** object to indicate the status of an operation.

## Syntax

```
typedef enum DPFPEventHandlerStatusEnum{
    EventHandlerStatusSuccess = 0,
    EventHandlerStatusFailure = 1,
} DPFPEventHandlerStatusEnum;
```

## Constants

<b>EventHandlerStatusSuccess</b>	An operation was performed successfully.
<b>EventHandlerStatusFailure</b>	An operation failed.

## Remarks

The members of this enumerated type are called by the `IDPFPEventHandlerStatus::Status` property (page 81).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPShrX.dll

## DPFPDataPurposeEnum Enumerated Type

The `DPFPDataPurposeEnum` enumerated type defines the purpose for which a fingerprint feature set is to be used.

## Syntax

```
typedef enum DPFPDataPurposeEnum{
    DataPurposeUnknown = 0,
    DataPurposeVerification = 1,
    DataPurposeEnrollment = 2,
} DPFPDataPurposeEnum;
```

## Constants

<code>DataPurposeUnknown</code>	The purpose is not known.
<code>DataPurposeVerification</code>	A fingerprint feature set to be used for the purpose of verification.
<code>DataPurposeEnrollment</code>	A fingerprint feature set to be used for the purpose of enrollment.

## Remarks

The members of this enumerated type are called by the `IDPFPPFeatureExtraction::CreateFeatureSet` method (page 82).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

## DPFPReaderImpressionTypeEnum Enumerated Type

The **DPFPReaderImpressionTypeEnum** enumerated type defines the modality that a fingerprint reader uses to capture fingerprint samples.

### Syntax

```
typedef enum DPFPReaderImpressionTypeEnum{
    ReaderImpressionTypeUnknown = 0,
    ReaderImpressionTypeSwipe = 1,
    ReaderImpressionTypeArea = 2,
} DPFPReaderImpressionTypeEnum;
```

### Constants

<b>ReaderImpressionTypeUnknown</b>	A fingerprint reader for which the modality is not known.
<b>ReaderImpressionTypeSwipe</b>	A swipe fingerprint reader.
<b>ReaderImpressionTypeArea</b>	An area (touch) sensor fingerprint reader.

### Remarks

The members of this enumerated type are called by the **IDPFPReaderDescription::ImpressionType** property (*page 86*).

### Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPReaderTechnologyEnum Enumerated Type

The **DPFPReaderTechnologyEnum** enumerated type defines the fingerprint reader technology.

### Syntax

```
typedef enum DPFPReaderTechnologyEnum{
    ReaderTechnologyUnknown = 0,
    ReaderTechnologyOptical = 1,
    ReaderTechnologyCapacitive = 2,
    ReaderTechnologyThermal = 3,
    ReaderTechnologyPressure = 4,
} DPFPReaderTechnologyEnum;
```



## Constants

<b>ReaderTechnologyUnknown</b>	A fingerprint reader for which the technology is not known.
<b>ReaderTechnologyOptical</b>	An optical fingerprint reader.
<b>ReaderTechnologyCapacitive</b>	A capacitive fingerprint reader.
<b>ReaderTechnologyThermal</b>	A thermal fingerprint reader.
<b>ReaderTechnologyPressure</b>	A pressure fingerprint reader.

## Remarks

The members of this enumerated type are called by the **IDPFPRReaderDescription::Technology** property (page 87).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFPSerialNumberTypeEnum Enumerated Type

The **DPFPSerialNumberTypeEnum** enumerated type defines whether a fingerprint reader serial number persists after reboot.

## Syntax

```
typedef enum DPFPSerialNumberTypeEnum{
    SerialNumberTypePersistent = 0,
    SerialNumberTypeVolatile = 1,
} DPFPSerialNumberTypeEnum;
```

## Constants

<b>SerialNumberTypePersistent</b>	A persistent serial number.
<b>SerialNumberTypeVolatile</b>	A volatile serial number.

## Remarks

The members of this enumerated type are called by the **IDPFPRReaderDescription::SerialNumberType** property (page 87).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

## DPFP\_TemplateStatusEnum Enumerated Type

The **DPFP\_TemplateStatusEnum** enumerated type defines the status of a fingerprint template creation operation.

### Syntax

```
typedef enum DPFP_TemplateStatusEnum{  
    TemplateStatusUnknown = 0,  
    TemplateStatusInsufficient = 1,  
    TemplateStatusFailed = 2,  
    TemplateStatusReady = 3,  
} DPFP_TemplateStatusEnum;
```

### Constants

<b>TemplateStatusUnknown</b>	The status of a template creation operation is not know, probably because a fingerprint template does not exist yet.
<b>TemplateStatusInsufficient</b>	A fingerprint template exists, but more fingerprint feature sets are required to complete it.
<b>TemplateStatusFailed</b>	A fingerprint template creation operation failed.
<b>TemplateStatusReady</b>	A fingerprint template was created and is ready for use.

### Remarks

The members of this enumerated type are called by the **IDPFPEnrollment::TemplateStatus** property ([page 75](#)).

## Enumerated Type Information

Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

This chapter describes the functionality of the user interfaces included in the following component objects:

- **DPFPEnrollmentControl**

This object includes the user interface described in the next section. The methods and properties for this object are described on *page 39* for Visual Basic and on *page 73* and *page 79* for C++.

- **DPFPVerificationControl**

This object includes the user interface described on *page 116*. The methods and properties for this object are described on *page 54* for Visual Basic and *page 95* and *page 97* for C++.

## DPFPEnrollmentControl Object User Interface

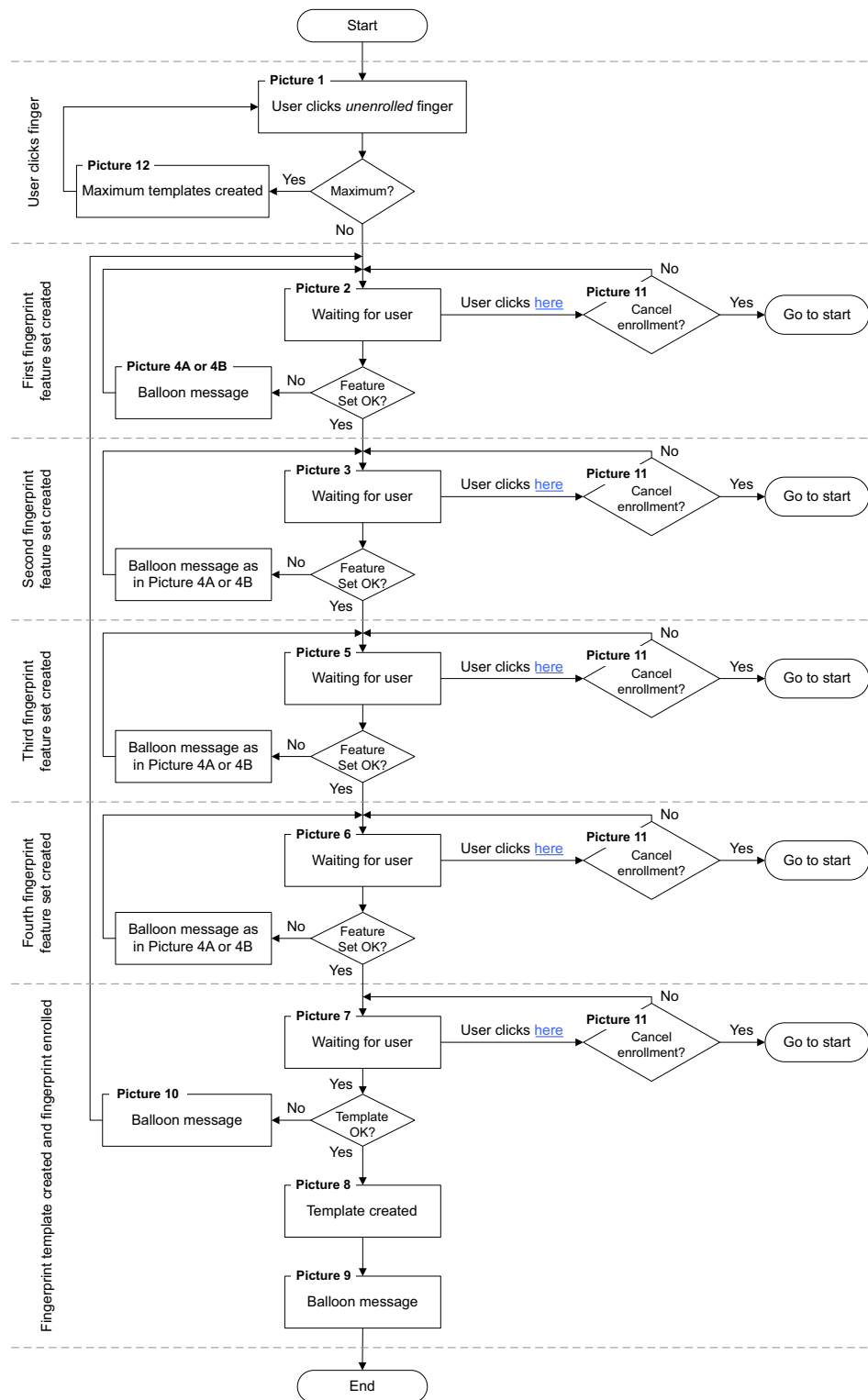
The user interface included with the **DPFPEnrollmentControl** object consists of two elements. The first element is used to provide instructions for selecting a fingerprint to enroll and for selecting a fingerprint template to delete, and is used to indicate already-enrolled fingerprints. The second element is used to provide instructions and feedback, both graphically and textually, about the enrollment process.

The tables and figure in this section describe the interaction between the user and the user interface during fingerprint enrollment and fingerprint template deletion.

NOTE: In the tables, the elements are referred to as the *hands element* and the *numbers element*.

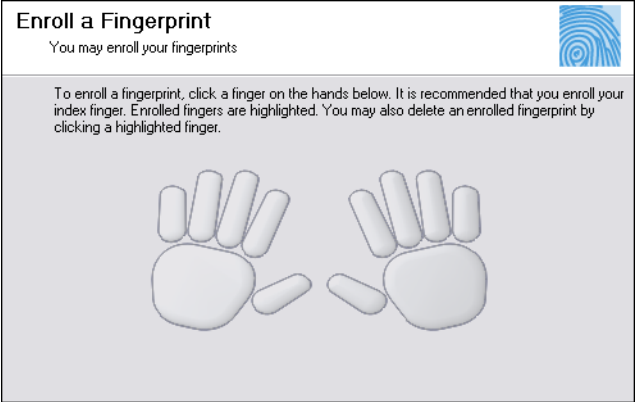
### Enrolling a Fingerprint

*Figure 9* illustrates the fingerprint enrollment process using the **DPFPEnrollmentControl** object interface. Picture numbers in the figure correspond to the pictures in *Table 7* on *page 109*. *Table 7* illustrates and describes the interaction between the user and the user interface during fingerprint enrollment.



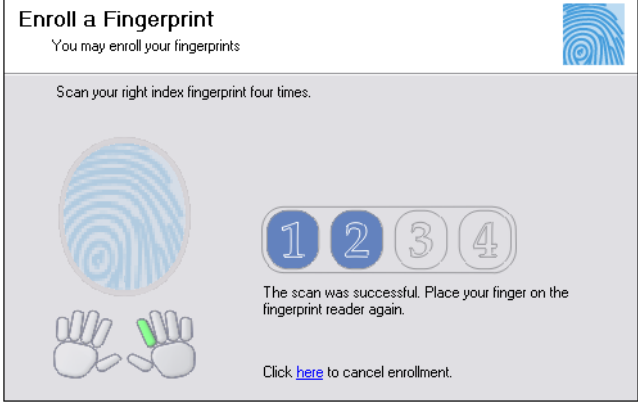


**Figure 9.** Enrolling a fingerprint using the `DPFPCControlEnrollment` object user interface

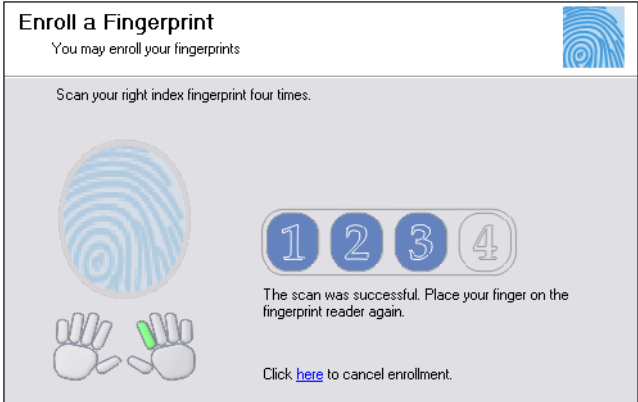
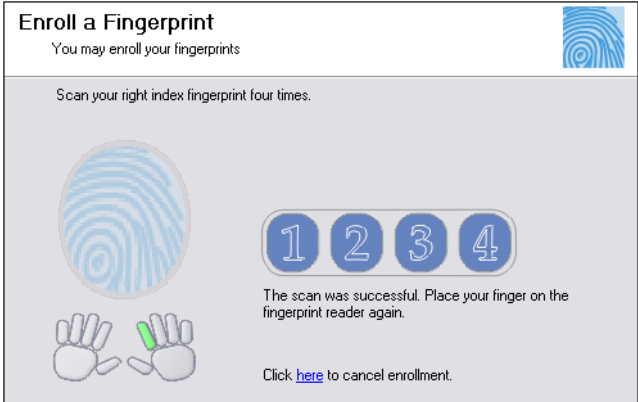
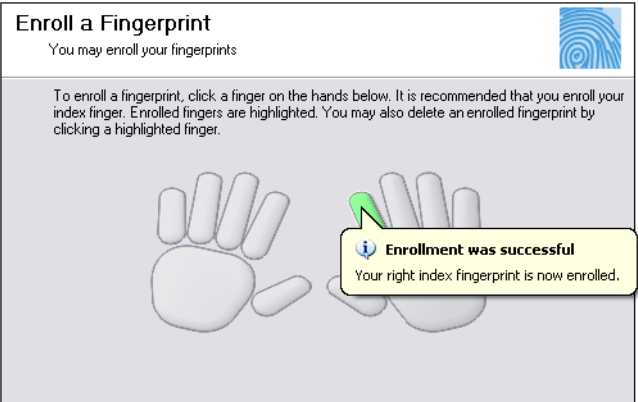
**Table 7.** `DPFPEnrollmentControl` object user interface: Enrolling a fingerprint

User interface	User actions and user interface feedback
<p><b>Picture 1</b></p> 	<p>This image indicates that no fingerprints have been enrolled, because the fingers associated with any enrolled fingerprints would be green.</p> <p>The user clicked the right index finger, and control was passed from the hands element to the numbers element.</p>
<p><b>Picture 2</b></p> here to cancel enrollment.'" data-bbox="100 422 488 612"/>	<p>The hands element is ready to enroll the user's right index fingerprint, as indicated by the green finger on the hand in the bottom left corner.</p>
<p><b>Picture 3</b></p> here to cancel enrollment.'" data-bbox="100 646 488 836"/>	<p>The user touched the fingerprint reader, and a fingerprint feature set was created.</p>

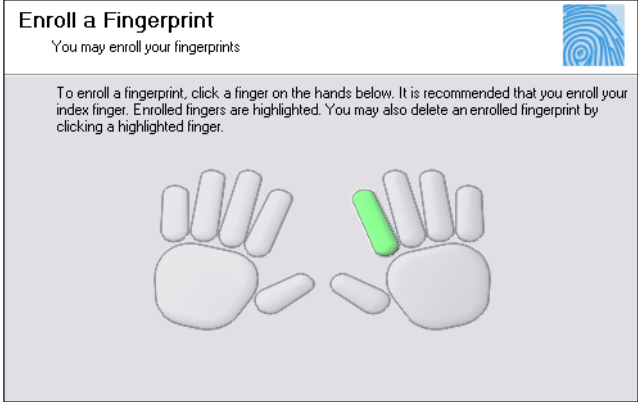
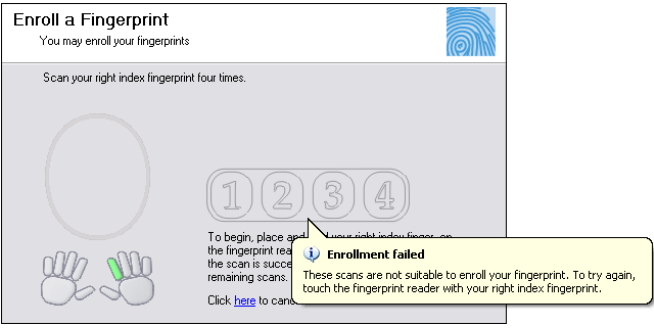

**Table 7. DPFPEnrollmentControl** object user interface: Enrolling a fingerprint (*continued*)

User interface	User actions and user interface feedback
<p><b>Picture 4A</b></p>  <p><b>Enroll a Fingerprint</b> You may enroll your fingerprints</p> <p>Scan your right index fingerprint four times.</p> <p>This scan was not successful. Try again. Place it flat on the fingerprint reader.</p> <p>Click <a href="#">here</a> to cancel enrollment.</p> <p>Close</p>	<p>The user touched the fingerprint reader, but a fingerprint feature set was not created. The message that is displayed depends on the quality of the fingerprint sample, as shown in Pictures 4A and 4B.</p>
<p><b>Picture 4B</b></p>  <p><b>Enroll a Fingerprint</b> You may enroll your fingerprints</p> <p>Scan your right index fingerprint four times.</p> <p>This scan was not successful. Lift your finger and try again. Place it flat on the fingerprint reader.</p> <p>Click <a href="#">here</a> to cancel enrollment.</p>	<p>The user touched the fingerprint reader, and a second fingerprint feature set was created.</p>
<p><b>Picture 5</b></p>  <p><b>Enroll a Fingerprint</b> You may enroll your fingerprints</p> <p>Scan your right index fingerprint four times.</p> <p>The scan was successful. Place your finger on the fingerprint reader again.</p> <p>Click <a href="#">here</a> to cancel enrollment.</p>	

**Table 7.** `DPFPEnrollmentControl` object user interface: Enrolling a fingerprint (*continued*)

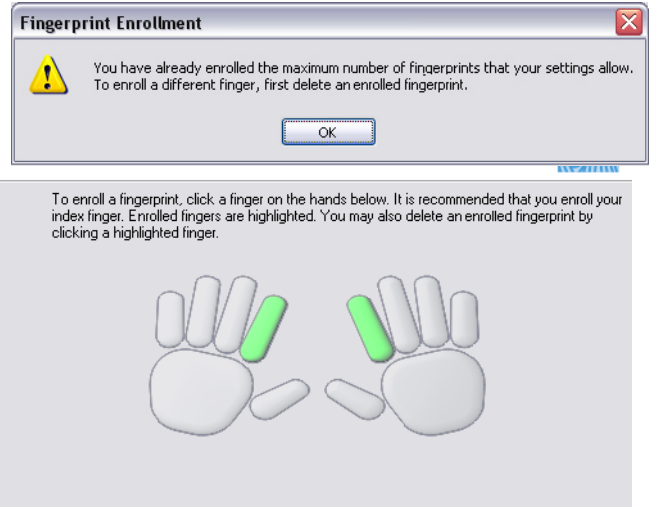
User interface	User actions and user interface feedback
<p><b>Picture 6</b></p> 	<p>The user touched the fingerprint reader, and a third fingerprint feature set was created.</p>
<p><b>Picture 7</b></p> 	<p>The user touched the fingerprint reader, and a fourth fingerprint feature set was created.</p> <p>Although the instructions to scan the finger again still appear, control is passed to the hands element almost immediately (see the next figure).</p>
<p><b>Picture 8</b></p> 	<p>A fingerprint template was created for the selected finger.</p> <p>This image appears when the <code>OnEnroll</code> event of the <code>DPFPEnrollmentControl</code> object is fired and returns a status of <code>EventHandlerStatusSuccess</code>.</p>

**Table 7.** `DPFPEnrollmentControl` object user interface: Enrolling a fingerprint (*continued*)

User interface	User actions and user interface feedback
<p>Picture 9</p> 	<p>The hands element indicates that the right index fingerprint is enrolled, that is, the finger is green.</p> <p>The value of the <code>EnrolledFingersMask</code> property is <code>000000000 001000000</code>, or 64.</p>
<p>Picture 10</p> 	<p>A fingerprint template was not created for the selected finger.</p> <p>The user is instructed to try again, and control remains with the numbers element.</p>
<p>Picture 11</p> 	<p>This message appears when the user clicks <b>here</b> in <b>Click here to cancel enrollment</b>. When the user clicks <b>No</b>, this message is dismissed and control is returned to the numbers element. When the user clicks <b>Yes</b>, this message is dismissed and control is passed to the hands element. The user can cancel enrollment at any time by clicking <b>here</b>.</p>



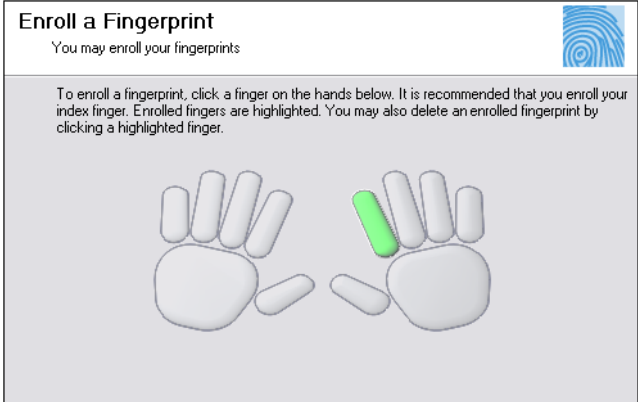

**Table 7. DPFPEnrollmentControl** object user interface: Enrolling a fingerprint (continued)

User interface	User actions and user interface feedback
<p>Picture 12</p> 	<p>This message is displayed when a user who has already enrolled the maximum allowed number of fingerprints (set by the <b>MaxEnrollFingerCount</b> property) clicks a finger associated with an unenrolled finger in the hands element. When the user clicks <b>OK</b>, control is returned to the hands element.</p>

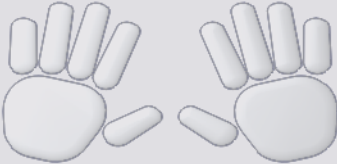

## Deleting a Fingerprint Template

Table 8 on *page 114* illustrates and describes the interaction between the user and the user interface during fingerprint template deletion.

**Table 8.** `DPFPEnrollmentControl` object user interface: Deleting a fingerprint template

User interface	User actions and user interface feedback
	<p>The hands element indicates that the right index fingerprint is enrolled, that is, the finger is green.</p> <p>The value of the <code>EnrolledFingersMask</code> property is 000000000 001000000, or 64.</p>
	<p>This message appears when the user clicks the right index fingerprint (which was previously enrolled) to delete its associated fingerprint template.</p> <p>When the user clicks <b>No</b>, this message is dismissed and control is returned to the hands element, which remains unchanged.</p>






**Table 8.** `DPFPEnrollmentControl` object user interface: Deleting a fingerprint template *(continued)*

User interface	User actions and user interface feedback
<div><div><div>Enroll a Fingerprint</div><div>You may enroll your fingerprints</div><div><div><div>To enroll a fingerprint, click a finger on the hands below. It is recommended that you enroll your index finger. Enrolled fingers are highlighted. You may also delete an enrolled fingerprint by clicking a highlighted finger.</div><div></div></div></div><div></div></div></div>	<p>When the user clicks <b>Yes</b>, this message is dismissed and control is returned to the hands element, where the green color is removed from the right index finger (the fingerprint template was deleted).</p> <p>The value of the <code>EnrolledFingersMask</code> property is now set to <code>000000000 000000000</code>, or 0.</p>

## DPFPVerificationControl Object User Interface

The user interface included with the **DPFPVerificationControl** object consists of one element. This element is used to indicate the connection status of the fingerprint reader and to provide feedback about the fingerprint verification process. *Table 9* illustrates and describes the interaction between the user and the user interface.

**Table 9.** DPFPVerificationControl object user interface

User interface	User actions and user interface feedback
	Indicates that the fingerprint reader is disconnected and ready for the user to scan a finger.
	Indicates that the fingerprint reader is disconnected.
	Indicates a comparison decision of match from a fingerprint verification operation.  This image appears when the <b>OnComplete</b> event of the <b>DPFPVerificationControl</b> object is fired and returns a status of <b>EventHandlerStatusSuccess</b> , and the value of the <b>Verified</b> property of the <b>DPFPVerificationResult</b> object is true.
	Indicates a comparison decision of non-match from a fingerprint verification operation.  This image appears when the <b>OnComplete</b> event of the <b>DPFPVerificationControl</b> object is fired and returns a status of <b>EventHandlerStatusSuccess</b> , and the value of the <b>Verified</b> property of the <b>DPFPVerificationResult</b> object is false.
	Indicates that the fingerprint verification operation failed.  This image appears when the <b>OnComplete</b> event of the <b>DPFPVerificationControl</b> object is fired and returns a status of <b>EventHandlerStatusFailure</b> .

You may redistribute the files in the RTE\Install and the Redist folders of the One Touch for Windows SDK product CD 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 on the product CD.

When you develop a product based on the One Touch for Windows 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 on the product CD 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 of the One Touch for Windows SDK product CD 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

- DpDrivers.msm

This merge module contains the following files:

- Dpd00701x64.dll
- Dpdevctlx64.dll
- Dpdevdatx64.dll
- Dpersona\_x64.cat
- Dpersona\_x64.inf
- Dpi00701x64.dll
- Dpinst32.exe
- Dpinst64.exe
- Usbdpfp.sys
- Dpersona.cat
- Dpersona.inf
- Dpdevctl.dll
- Dpdevdat.dll
- Dpk00701.sys
- Dpk00303.sys
- Dpd00303.dll
- Dpd00701.dll
- Dpi00701.dll

- DpFpRec.msm

This merge module contains the following files:

- Dphftrex.dll
- Dphmatch.dll

- DPFpUI.msm

This merge module contains the following file:

- Dpfpui.dll

- DpProCore.msm

This merge module contains the following files:

- Dpdevts.dll
- Dpsvinfo2.dll
- Dptsclnt.dll

- COM/ActiveX libraries

- DPFPShrX.dll
- DPFPPDevX.dll
- DPFPEngX.dll
- DPFPCtlX.dll

## Fingerprint Reader Documentation

You may redistribute the documentation included in the Redist folder on the One Touch for Windows SDK product CD 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 on the product CD.

## 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 on the One Touch for Windows SDK product CD. 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 folder on the One Touch for Windows SDK product CD. 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.

This appendix is for developers who want to specify a false accept rate (FAR) other than the default used by the DigitalPersona Fingerprint Recognition Engine.

## False Accept Rate (FAR)

The false accept rate (FAR), also known as the security level, is the proportion of fingerprint verification operations by authorized users that incorrectly returns a comparison decision of match. The FAR is typically stated as the ratio of the expected number of false accept errors divided by the total number of verification attempts, or the probability that a biometric system will falsely accept an unauthorized user. For example, a probability of 0.001 (or 0.1%) means that out of 1,000 verification operations by authorized users, a system is expected to return 1 incorrect match decision. Increasing the probability to, say, 0.0001 (or 0.01%) changes this ratio from 1 in 1,000 to 1 in 10,000.

Increasing or decreasing the FAR has the opposite effect on the false reject rate (FRR), that is, decreasing the rate of false accepts increases the rate of false rejects and vice versa. Therefore, a high security level may be appropriate for an access system to a secured area, but may not be acceptable for a system where convenience or easy access is more significant than security.

## Representation of Probability

The DigitalPersona Fingerprint Recognition Engine supports the representation for the FAR probability that fully conforms to the BIOAPI 1.1, BioAPI 2.0, and UPOS standard specifications. In this representation, the probability is represented as a positive 32-bit integer, or zero. (Negative values are reserved for special uses.)

The definition `PROBABILITY_ONE` provides a convenient way of using this representation. `PROBABILITY_ONE` has the value 0x7FFFFFFF (where the prefix 0x denotes base 16 notation), which is 2147483647 in decimal notation. If the probability ( $P$ ) is encoded by the value (`INT_N`), then

$$INT\_N = P \times PROBABILITY\_ONE$$

$$P = \frac{INT\_N}{PROBABILITY\_ONE}$$

Probability  $P$  should always be in the range from 0 to 1. Some common representations of probability are listed in column one of *Table 2*. The value in the third row represents the current default value used by the DigitalPersona Fingerprint Recognition Engine, which offers a mid-range security level. The value in the second row represents a typical high FAR/low security level, and the value in the fourth row represents a typical low FAR/high security level.

The resultant value of `INT_N` is represented in column two, in decimal notation.



**Table 2.** Common values of probability and resultant INT\_N values

Probability (P)	Value of INT_N in decimal notation
$0.001 = 0.1\% = 1/1000$	2147483
$0.0001 = 0.01\% = 1/10000$	214748
$0.00001 = 0.001\% = 1/100000$	21475
$0.000001 = 0.0001\% = 1/1000000$	2147

## Requested FAR

You specify the value of the FAR, which is INT\_N from the previous equation, using the **FARRequested** property. While you can request any value from 0 to the value PROBABILITY\_ONE, it is not guaranteed that the Engine will fulfill the request exactly. The Engine implementation makes the best effort to accommodate the request by internally setting the value closest to that requested within the restrictions it imposes for security.

## Specifying the FAR in C++

If you are developing your application in C++, you specify the value of the FAR (INT\_N) in the **pVal** parameter of the **IDPFVerification::FARRequested** property. The following sample code sets the FAR to a value of 0.000001, or 0.0001%.

```
#define PROBABILITY_ONE (0x7FFFFFFF)

IDPFVerification* verification;
...

//Sets the FAR to 0.0001%
rc = verification -> put_FARRequested (PROBABILITY_ONE / 1000000);
```

## Specifying the FAR in Visual Basic

If you are developing your application in Visual Basic, you specify the value of the FAR (INT\_N) in the **lValue** parameter in the **FARRequested** property of the **DPFPVerification** object. The following sample code sets the FAR to a value of 0.0001, or 0.01%.

```
Const PROBABILITY_ONE as Long = &H7FFFFFFF

Dim verification as DPFPVerification
...

' Sets the FAR to 0.01%
verification.FARRequested = PROBABILITY_ONE / 10000
```

## Achieved FAR

The actual value of the FAR achieved for a particular verification operation is returned in the **pVal** parameter of **IDPFPVerificationResult::FARAchieved** property in C++ or in the **lValue** parameter of the **FARAchieved** property of the **DPFPVerificationResult** object in Visual Basic. This value is typically much smaller than the requested FAR due to the accuracy of the DigitalPersona Fingerprint Recognition Engine. The requested FAR specifies the maximum value of the FAR to be used by the Engine in making the verification decision. The actual FAR achieved by the Engine when conducting a legitimate comparison is usually a much lower value. The Engine implementation may choose the range and granularity for the achieved FAR. If you make use of this value in your application, for example, by combining it with other achieved FARs, you should use it with caution, as the granularity and range may change between versions of DigitalPersona SDKs without notice.

## Testing

Although you may achieve the desired values of the FAR in your development environment, it is not guaranteed that your application will achieve the required security level in real-world situations. Even though the Engine is designed to make its best effort to accurately implement the probability estimates, it is recommended that you conduct system-level testing to determine the actual operating point and accuracy in a given scenario. This is even more important in systems where multiple biometric factors are used for identification.

This appendix is for Platinum SDK users who need to convert their Platinum SDK registration templates to a format that is compatible with the SDKs that are listed in *Fingerprint Template Compatibility* on page 5. You can use the following sample codes for this purpose.

## Platinum SDK Registration Template Conversion for Microsoft Visual C++ Applications

Use *Code Sample 1* in applications developed in Microsoft Visual C++ to convert DigitalPersona Platinum SDK registration templates.

### Code Sample 1. Platinum SDK Template Conversion for Microsoft Visual C++ Applications

```
#import "DpSdkEng.tlb" no_namespace, named_guids, raw_interfaces_only
#include <atlbase.h>

bool PlatinumTOGold(unsigned char* platinumBlob, int platinumBlobSize,
                    unsigned char* goldBlob, int goldBufferSize,
                    int* goldTemplateSize)
{
    // Load the byte array into FPTemplate Object
    // to create Platinum template object
    SAFEARRAYBOUND rgsabound;
    rgsabound.lLbound = 0;
    rgsabound.cElements = platinumBlobSize;

    CComVariant varVal;
    varVal.vt = VT_ARRAY | VT_UI1;
    varVal.parray = SafeArrayCreate(VT_UI1, 1, &rgsabound);

    unsigned char* data;
    if (FAILED(SafeArrayAccessData(varVal.parray, (void**)&data)))
        return false;

    memcpy(data, platinumBlob, platinumBlobSize);
    SafeArrayUnaccessData(varVal.parray);

    IFPTemplatePtr pIFPTemplate(__uuidof(FPTemplate));

    if (pIFPTemplate == NULL)
        return false;
```

**Code Sample 1.** Platinum SDK Template Conversion for Microsoft Visual C++ Applications (*continued*)

```

    HRESULT error;
    if (FAILED(pIFPTemplate->Import(varVal, &error)))
        return false;

    if (error != Er_OK)
        return false;

    // Now pIFPTemplate contains the Platinum template.
    // Use TemplData property to get the Gold Template out.
    CComVariant varValGold;

    if (FAILED(pIFPTemplate->get_TemplData(&varValGold)))
        return false;

    unsigned char* dataGold;
    if (FAILED(SafeArrayAccessData(varValGold.parray, (void**)&dataGold)))
        return false;

    int blobSizeRequired = varValGold.parray->rgsabound->cElements *
                           varValGold.parray->cbElements;
    *goldTemplateSize = blobSizeRequired;

    if (goldBufferSize < blobSizeRequired) {
        SafeArrayUnaccessData(varValGold.parray);
        return false;
    }

    memcpy(goldBlob, dataGold, blobSizeRequired);

    SafeArrayUnaccessData(varValGold.parray);

    return true;
}

```

## Platinum SDK Registration Template Conversion for Visual Basic 6.0 Applications

Use *Code Sample 2* in applications developed in Microsoft Visual Basic 6.0 to convert DigitalPersona Platinum SDK registration templates.

### Code Sample 2. Platinum SDK Template Conversion for Visual Basic 6.0 Applications

```
Public Function PlatinumToGold(platinumTemplate As Variant) As Byte()  
    Dim pTemplate As New FPTemplate  
    Dim vGold As Variant  
    Dim bGold() As Byte  
  
    Dim er As DpSdkEngLib.AIErrors  
    er = pTemplate.Import(platinumTemplate)  
    If er <> Er_OK Then PlatinumToGold = "": Exit Function  
    vGold = pTemplate.TemplData  
    bGold = vGold  
    PlatinumToGold = bGold  
End Function
```

# Glossary

## **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).

## **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 object**

An object that inherits the properties of a `DPFPData` object. Fingerprint data objects include `DPFPDataSample` (represents a fingerprint sample), `DPFPFeatureSet` (represents a fingerprint feature set), and `DPFPDataTemplate` (represents a fingerprint template).

## **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 enrollment subject**

The fingerprint data subject whose fingerprint template(s) are held in the fingerprint data storage subsystem.

**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**.



# Index

## Symbols

- \_IDPFPCaptureEvents interface for C++ 69
- \_IDPFPEnrollmentControlEvents interface for C++ 79
- \_IDPFVerificationControlEvents interface for C++ 97
- \_NewEnum property
  - DPFPCReadersCollection object for Visual Basic 50
  - IDPFPCReadersCollection interface for C++ 90

## A

- AddFeatures method
  - calling in typical fingerprint enrollment workflow 21
  - DPFPCEnrollment object for Visual Basic 37
  - IDPFPCEnrollment interface for C++ 73
- additional resources 4
  - online resources 4
  - related documentation 4
- API reference
  - C++ 65–106
  - Visual Basic 30–64
- audience for this guide 1

## B

- biometric system
  - defined 126
  - explained 15
- bold typeface, uses of 3

## C

- chapters, overview of 1
- Clear method
  - DPFPCEnrollment object for Visual Basic 37
  - IDPFPCEnrollment interface for C++ 74
- comparison, defined 126
- compatible fingerprint templates
  - See fingerprint template compatibility matrix
- component objects (Visual Basic) 30–56
  - See also individual components objects by name
- conventions, document
  - See document conventions
- converting Platinum SDK registration templates
  - for Microsoft Visual Basic 6.0 applications 125
  - for Microsoft Visual C++ applications 123
- ConvertToANSI381 method
  - DPFPCSampleConversion object for Visual Basic 51
  - IDPFPCSampleConversion interface for C++ 92
- ConvertToPicture method
  - DPFPCSampleConversion object for Visual Basic 52
  - IDPFPCSampleConversion interface for C++ 92

- Count property
  - DPFPCReadersCollection object for Visual Basic 49
  - IDPFPCReadersCollection interface for C++ 89
- Courier bold typeface, use of 3
- CreateFeatureSet method
  - calling
    - in typical fingerprint enrollment workflow 21
    - in typical fingerprint verification workflow 25
  - DPFPCFeatureExtraction object for Visual Basic 44
  - IDPFPCFeatureExtraction interface for C++ 82

## D

- data object
  - See fingerprint data object
- Deserialize method
  - calling in fingerprint data object deserialization workflow 29
  - DPFPCData object for Visual Basic 36
  - IDPFPCData interface for C++ 72
- deserializing fingerprint data object workflow 29
  - illustrated 29
- developer guide
  - overview of chapters 1
  - overview of concepts and terminology 15
- DigitalPersona Developer Connection Forum, URL to 4
- DigitalPersona Fingerprint Recognition Engine 15
- DigitalPersona fingerprint recognition system 16
  - illustrated 16
- DigitalPersona products, supported 5
- document conventions 2
  - naming 3
  - notational 2
  - typographical 3
- documentation, related 4
- DPFPCapture object for Visual Basic 32
- DPFPCapture object, creating
  - in typical fingerprint enrollment workflow 21
  - in typical fingerprint verification workflow 25
- DPFPCaptureEvents event handler, implementing
  - in typical fingerprint enrollment workflow 21
  - in typical fingerprint verification workflow 25
- DPFPCaptureFeedbackEnum
  - enumerated type for C++ 100
  - enumeration for Visual Basic 58
- DPFPCapturePriorityEnum
  - enumerated type for C++ 101
  - enumeration for Visual Basic 59

- DPFPData object for Visual Basic 36
  - DPFPData object, creating
    - in fingerprint data object deserialization workflow 29
    - in fingerprint data object serialization workflow 28
  - DPFPDataPurposeEnum
    - enumerated type for C++ 103
    - enumeration for Visual Basic 61
  - DPFPEnrollment object for Visual Basic 37
  - DPFPEnrollment object, creating, in typical fingerprint enrollment workflow 21
  - DPFPEnrollmentControl Events event handler, implementing, in typical fingerprint template deletion with UI support workflow 23
  - DPFPEnrollmentControl object for Visual Basic 39
  - DPFPEnrollmentControl object, creating
    - in typical fingerprint enrollment with UI support workflow 22
    - in typical fingerprint template deletion with UI support workflow 23
  - DPFPEnrollmentControlEvents event handler, implementing, in typical fingerprint enrollment with UI support workflow 22
  - DPFPEventHandlerStatus object for Visual Basic 43
  - DPFPEventHandlerStatusEnum
    - enumerated type for C++ 102
    - enumeration for Visual Basic 60
  - DPFPFeatureExtraction object for Visual Basic 44
  - DPFPFeatureExtraction object, creating
    - in typical fingerprint enrollment workflow 21
    - in typical fingerprint verification workflow 25
  - DPFPFeatureSet object
    - creating
      - in fingerprint data object deserialization workflow 29
      - in typical fingerprint enrollment workflow 21
      - in typical fingerprint verification workflow 25
    - receiving, in typical fingerprint verification workflow 27
  - DPFPFeatureSet object for Visual Basic 45
  - DPFPReaderDescription object for Visual Basic 46
  - DPFPReaderImpressionTypeEnum
    - enumerated type for C++ 104
    - enumeration for Visual Basic 62
  - DPFPReadersCollection object for Visual Basic 49
  - DPFPReaderTechnologyEnum
    - enumerated type for C++ 104
    - enumeration for Visual Basic 62
  - DPFPSample object for Visual Basic 51
  - DPFPSample object, creating, in fingerprint data object deserialization workflow 29
  - DPFPSampleConversion object for Visual Basic 51
  - DPFPSerialNumberTypeEnum
    - enumerated type for C++ 105
    - enumeration for Visual Basic 63
  - DPFPTemplate object
    - creating
      - from serialized data
        - in typical fingerprint verification with UI support workflow 27
      - in typical fingerprint verification workflow 25
      - in fingerprint data object deserialization workflow 29
      - in typical fingerprint enrollment workflow 21
    - serializing
      - in typical fingerprint enrollment with UI support workflow 23
      - in typical fingerprint enrollment workflow 21
    - storing
      - in typical fingerprint enrollment with UI support workflow 23
      - in typical fingerprint enrollment workflow 21
  - DPFPTemplate object for Visual Basic 52
  - DPFPTemplateStatusEnum
    - enumerated type for C++ 106
    - enumeration for Visual Basic 64
  - DPFPVerification object for Visual Basic 52
  - DPFPVerification object, creating
    - in typical fingerprint verification with UI support workflow 27
    - in typical fingerprint verification workflow 25
  - DPFPVerificationControl object for Visual Basic 54
  - DPFPVerificationControl object, creating, in typical fingerprint verification with UI support workflow 27
  - DPFPVerificationControlEvents event handler, implementing, in typical fingerprint verification with UI support workflow 27
  - DPFPVerificationResult object for Visual Basic 56
  - DPFPVerificationResult object, receiving
    - in typical fingerprint verification with UI support workflow 27
    - in typical fingerprint verification workflow 25
- ## E
- Engine
    - See DigitalPersona Fingerprint Recognition Engine
  - EnrolledFingersMask property
    - DPFPEnrollmentControl object for Visual Basic 39
    - IDFPEnrollmentControl interface for C++ 76

- setting
  - in typical fingerprint enrollment with UI support workflow 22
  - in typical fingerprint template deletion with UI support workflow 23

## enrollment

- See fingerprint enrollment
- enrollment mask, possible values for
  - C++ 77
  - Visual Basic 40, 77

## enumerations

- C++ 99–106
  - See also individual enumerated types by name
- Visual Basic 57–64
  - See also individual enumerations by name

## F

- false accept rate 18
  - defined 126
  - setting to value other than the default 120

- false negative decision 18

- false negative decision, proportion of 18
  - See also false accept rate

- false positive decision 18

- false positive decision, proportion of 18
  - See also false accept rate

- false positives and false negatives 18

- false reject rate 18
  - defined 126

## FAR

- See false accept rate

## FARAchieved property

- DPFPVerificationResult object for Visual Basic 56
- explanation of 122
- IDFPFVerificationResult interface for C++ 97

## FARRequested property

- DPFPVerification object for Visual Basic 53
- IDFPFVerification interface for C++ 93
- important notice to read Appendix A before setting 53, 94
- setting
  - in typical verification with UI support workflow 27
  - in typical verification workflow 25
- setting to other than the default 121

## features

- See fingerprint features

## FeatureSet property

- DPFPFeatureExtraction object for Visual Basic 45
- IDFPFFeatureExtraction interface for C++ 83

## FeaturesNeeded property

- DPFPEnrollment object for Visual Basic 38
- IDFPFEnrollment interface for C++ 74

## files and folders

- installed for RTE 13
- installed for SDK 12

## fingerprint 15

- defined 126

## fingerprint capture device 17

- defined 126
- See fingerprint reader

## fingerprint characteristics, defined 126

## fingerprint data 17

- defined 126

## fingerprint data object 36, 72

- creating, in fingerprint data object serialization workflow 28
- defined 126
- retrieving serialized data from storage 29
- serialization/deserialization workflow 28
- storing serialized data, in fingerprint data object serialization workflow 28

## fingerprint data storage subsystem, defined 126

## fingerprint enrollment 17

- defined 127
- with UI support workflows 22
- workflow 19
  - illustration of typical 20

## fingerprint feature extraction 17

- defined 127

## fingerprint feature set 17

- defined 127
- See also DPFPFeatureSet object

## fingerprint features, defined 127

## fingerprint image 17

- defined 127
- See also fingerprint sample

## fingerprint mask, possible values for

- C++ 80
- Visual Basic 42

## fingerprint reader 16

- defined 127
- driver 16
- redistributing documentation for 119
- use and maintenance guide
  - redistributing 119

## fingerprint recognition 16

## fingerprint recognition system 15

- defined 127
- See also DigitalPersona fingerprint recognition system

fingerprint recognition, guide to 4

fingerprint sample

- capturing
  - in typical fingerprint enrollment with UI support workflow 22
  - in typical fingerprint enrollment workflow 21
  - in typical fingerprint verification with UI support workflow 27
  - in typical fingerprint verification workflow 25
- defined 127
- See also* DPFP Sample object
- See also* fingerprint image

fingerprint template 17

- defined 127
- deleting from storage, in typical fingerprint template deletion workflow 23
- retrieving serialized data from storage
  - in typical fingerprint verification with UI support workflow 27
  - in typical fingerprint verification workflow 25
- See also* DPFP Template object
- serializing, in typical fingerprint enrollment workflow 21
- storing
  - in typical fingerprint enrollment with UI support workflow 23
  - in typical fingerprint enrollment workflow 21
- workflow for deleting with UI support 23
  - illustrated 23
- workflow for enrolling 19
  - illustrated 20
- workflow for enrolling with UI support 22
  - illustrated 22

fingerprint template compatibility matrix 5

fingerprint verification 17

- defined 127

fingerprint verification with UI support workflow 26

- illustrated 26

fingerprint verification workflow 23

- illustrated 24

FirmwareRevision property

- DPFPReaderDescription object for Visual Basic 46
- IDFPFReaderDescription interface for C++ 84

folders and files

- installed for RTE 13
- installed for SDK 12

FRR

- See* false reject rate

## H

hardware warnings and regulatory information redistributing 119

HardwareRevision property

- DPFPReaderDescription object for Visual Basic 46
- IDFPFReaderDescription interface for C++ 85

## I

IDFPFCapture interface for C++ 67

IDFPFData interface for C++ 72

IDFPFEnrollment interface for C++ 73

IDFPFEnrollmentControl interface for C++ 76

IDFPFEventHandlerStatus interface for C++ 81

IDFPFFeatureExtraction interface for C++ 82

IDFPFFeatureSet interface for C++ 84

IDFPFReaderDescription interface for C++ 84

IDFPFReadersCollection interface for C++ 88

IDFPFSample interface for C++ 91

IDFPFSampleConversion interface for C++ 91

IDFPFTemplate interface for C++ 93

IDFPFVerification interface for C++ 93

IDFPFVerificationControl interface for C++ 95

IDFPFVerificationResult interface for C++ 97

image

- See* fingerprint image

important notation, defined 2

important notice

- read Appendix A before setting FARRequested 53
- read Appendix A before setting FARRequested property 94

ImpressionType property

- DPFPReaderDescription object for Visual Basic 47
- IDFPFReaderDescription interface for C++ 86

installation 12

installation files for redistributables

- contents of RTE\Install folder 117
- redistributing 117

installing

- RTE 13
- RTE silently 14
- SDK 12

interfaces (C++) 65–98

- See also* individual interfaces by name

italics typeface, uses of 3

Item property

- DPFPReadersCollection object for Visual Basic 50
- IDFPFReadersCollection interface for C++ 90

**L**

## Language property

DPFPReaderDescription object for Visual Basic 46

IDFPFReaderDescription interface for C++ 85

**M**

## match 17

defined 127

## MaxEnrollFingerCount property

DPFPEnrollmentControl object for Visual Basic 40

IDFPFEnrollmentControl interface for C++ 77

## setting

in typical fingerprint enrollment with UI support  
workflow 22in typical fingerprint template deletion with UI  
support workflow 23

## merge modules

contents of 117

redistributing 117

**N**

## naming conventions 3

## non-match 17

defined 128

## notational conventions 2

## note notation, defined 2

**O**

## OnComplete event

\_IDFPFCaptureEvents interface for C++ 70

\_IDFPFVerificationControlEvents interface for C++ 97

DPFPCapture object for Visual Basic 34

DPFPVerificationControl object for Visual Basic 55

of DPFPCaptureEvents, receiving

in typical fingerprint enrollment workflow 21

in typical fingerprint verification workflow 25

of DPFPVerificationControlEvents, receiving, in typical

fingerprint verification with UI support

workflow 27

## OnDelete event

\_IDFPFEnrollmentControlEvents interface for C++ 79

DPFPEnrollmentControl object for Visual Basic 42

of DPFPEnrollmentControlEvents, receiving, in typical

fingerprint template with UI support

workflow 23

## OnEnroll event

\_IDFPFEnrollmentControlEvents interface for C++ 80

DPFPEnrollmentControl object for Visual Basic 43

of DPFPEnrollmentControlEvents, receiving, in typical

fingerprint template with UI support

workflow 22

## one-to-one comparison 17

defined 128

## OnFingerGone event

\_IDFPFCaptureEvents interface for C++ 70

DPFPCapture object for Visual Basic 34

## OnFingerTouch event

\_IDFPFCaptureEvents interface for C++ 70

DPFPCapture for Visual Basic 34

## online resources 4

## OnReaderConnect event

\_IDFPFCaptureEvents interface for C++ 71

DPFPCapture object for Visual Basic 35

## OnReaderDisconnect event

\_IDFPFCaptureEvents interface for C++ 71

DPFPCapture object for Visual Basic 35

## OnSampleQuality event

\_IDFPFCaptureEvents interface for C++ 71

DPFPCapture object for Visual Basic 35

## overview

of chapters 1

of concepts and terminology 15

**P**

## Platinum SDK registration template conversion 123

## Priority property

DPFPCapture object for Visual Basic 32

IDFPFCapture interface for C++ 67

## setting

in typical fingerprint enrollment workflow 21

in typical fingerprint verification workflow 25

## product compatibility

See fingerprint template compatibility matrix

## ProductName property

DPFPReaderDescription object for Visual Basic 47

IDFPFReaderDescription interface for C++ 86

**Q**

## quick start, introduction to SDK 6

**R**

## Reader method

DPFPReadersCollection object for Visual Basic 49

IDFPFReadersCollection interface for C++ 89

## ReaderSerialNumber property

DPFPCapture object for Visual Basic 33

DPFPEnrollmentControl object for Visual Basic 41

DPFPVerificationControl object for Visual Basic 54

IDFPFCapture interface for C++ 68

IDFPFEnrollmentControl interface for C++ 78

IDFPFVerificationControl interface for C++ 96

- of DPFPCapture, setting
  - in typical fingerprint enrollment workflow 21
  - in typical fingerprint verification workflow 25
- of DPFPEnrollmentControl
  - setting
    - in typical fingerprint enrollment with UI support workflow 22
    - in typical fingerprint template deletion with UI support workflow 23
- of DPFPVerificationControl, setting, in typical fingerprint verification with UI support workflow 27
- redistributable files
  - contents of 117
  - redistributing 117
- redistributables, redistributing 117
- redistribution of files 117
- regulatory information, requirement to advise end users of 119
- repository 17
- requirements, system
  - See system requirements
- resources, additional
  - See additional resources
- resources, online
  - See online resources
- RTE
  - installing 13
  - installing/uninstalling silently 14
  - redistributing 117
- runtime environment
  - See RTE

## S

- sample code for converting Platinum SDK registration templates
  - for Microsoft Visual Basic 6.0 applications 125
  - for Microsoft Visual C++ applications 123
- SDK
  - files and folders installed 12
  - installing 12
  - quick start 6
- security level 18
- Serialize method
  - calling in fingerprint data object serialization workflow 28
  - DPFPData object for Visual Basic 36
  - IDFPData interface for C++ 72
- serializing fingerprint data object workflow 28
  - illustrated 28

- SerialNumber property
  - DPFPReaderDescription object for Visual Basic 47
  - IDFPReaderDescription interface for C++ 86
- SerialNumberType property
  - DPFPReaderDescription object for Visual Basic 48
  - IDFPReaderDescription interface for C++ 87
- silently installing RTE 14
- StartCapture method
  - calling
    - in typical fingerprint enrollment workflow 21
    - in typical fingerprint verification workflow 25
  - DPFPCapture object for Visual Basic 32
  - IDFPCapture interface for C++ 68
- Status property
  - DPFPEventHandlerStatus object for Visual Basic 43
  - IDFPEventHandlerStatus interface for C++ 81
- StopCapture method
  - calling
    - in typical fingerprint enrollment workflow 21
    - in typical fingerprint verification workflow 25
  - DPFPCapture object for Visual Basic 32
  - IDFPCapture interface for C++ 69
- supported DigitalPersona products 5
- system requirements 4

## T

- target audience for this guide 1
- Technology property
  - DPFPReaderDescription object for Visual Basic 48
  - IDFPReaderDescription interface for C++ 87
- template compatibility matrix
  - See fingerprint template compatibility matrix
- Template property
  - DPFPEnrollment object for Visual Basic 38
  - IDFPEnrollment interface for C++ 74
- TemplateStatus property
  - DPFPEnrollment object for Visual Basic 38
  - IDFPEnrollment interface for C++ 75
- typefaces, uses of
  - bold 3
  - Courier bold 3
  - italics 3
- typographical conventions 3

## U

- uninstalling RTE silently 14
- updates for DigitalPersona software products, URL for downloading 4
- URL
  - DigitalPersona Developer Connection Forum 4
  - Updates for DigitalPersona Software Products 4

use and maintenance guide for fingerprint reader  
redistributing 119

## **V**

Vendor property

DPFPReaderDescription object for Visual Basic 48

IDFPFPReaderDescription interface for C++ 88

verification

See fingerprint verification

Verified property

DPFPVerificationResult object for Visual Basic 56

IDFPFPVerificationResult interface for C++ 98

Verify method

calling

in typical fingerprint verification with UI support  
workflow 27

in typical fingerprint verification workflow 25

DPFPVerification object for Visual Basic 53

IDFPFPVerification interface for C++ 94

## **W**

Web site

DigitalPersona Developer Connection Forum 4

Updates for DigitalPersona Software Products 4

workflows 19–29