DigitalPersona, Inc.

One Touch® for Windows® SDK

COM/ActiveX® Edition

Version 1.4

Developer Guide

digitalPersona.

DigitalPersona, Inc.
© 1996 – 2009 DigitalPersona, Inc. All Rights Reserved.

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

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

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

Except as permitted by such license or by the terms of this guide, no part of this document may be reproduced, stored, transmitted, and translated, in any form and by any means, without the prior written consent of DigitalPersona. The contents of this guide are furnished for informational use only and are subject to change without notice. Any mention of third-party companies and products is for demonstration purposes only and constitutes neither an endorsement nor a recommendation. DigitalPersona assumes no responsibility with regard to the performance or use of these third-party products. DigitalPersona makes every effort to ensure the accuracy of its documentation and assumes no responsibility or liability for any errors or inaccuracies that may appear in it.

Technical Support

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

Feedback

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

TechPubs@digitalpersona.com

or

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

Document Publication Date: May 22, 2009

Table of Contents

Introduction 1

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.

Fingerprint Authentication on a Remote Computer

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

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

The following types of Citrix clients are supported:

- Program Neighborhood
- Program Neighborhood Agent
- Web Client

Note that to take advantage of this feature, your fingerprint-enabled application must run on the Terminal Services or Citrix server, not on the client. If you are developing a Citrix-aware application, see additional information in the *Developing Citrix-aware applications* chapter on page *141*.

Target Audience

This guide is for developers who have a working knowledge of the C++ or Microsoft® 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 API components 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 API components 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 fingerprint enrollment and fingerprint verification ActiveX controls.

Chapter 9, *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 Enrollment 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
Bold	Used for keystrokes and window and dialog box elements and to indicate data types	Click **Fingerprint Enrollment**. The **Fingerprint Enrollment** dialog box appears. **String** that specifies a fingerprint reader serial number
`Courier bold`	Used to indicate computer programming code	When **`SampleQualityGood`** is returned, the **`OnComplete`** event is fired. Deserializes a data object returned by the **`IDPFPData::Serialize`** method.
Italics	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 *typical* fingerprint enrollment and fingerprint verification workflows. (emphasis) A *fingerprint* is an impression of the ridges on the skin of a finger. (new term) See *Installing the SDK* on *page 8*. (link to heading and page)

Naming Conventions

DPFP stands for *DigitalPersona Fingerprint*.

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 Engine	The DigitalPersona White Paper: Guide to Fingerprint Recognition (Fingerprint Guide.pdf) is located in the Docs folder in the One Touch for Windows software package, and is *not* automatically installed on your computer as part of the setup process.
Late-breaking news about the product	The Readme.txt files provided in the root directory in the SDK software package as well as in some subdirectories

Online Resources

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

System Requirements

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

- x86-based processor or better
- Microsoft® Windows® XP, 32-bit and 64-bit versions; Microsoft® Windows® XP Embedded, 32-bit version[1]; or Microsoft® Windows Vista®, 32-bit and 64-bit versions
- USB connector on the computer where the fingerprint reader is to be connected

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 in the SDK software package.

Supported DigitalPersona Hardware Products

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

- DigitalPersona U.are.U 4000B/4500 or later fingerprint readers and modules
- DigitalPersona U.are.U Fingerprint Keyboard

Fingerprint Template Compatibility

Fingerprint templates produced by all editions of the One Touch for Windows SDK are also compatible with the following DigitalPersona SDKs:

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

NOTE: Platinum SDK enrollment templates must be converted to a compatible format to work with these SDKs. See Appendix B on *page 150* for sample code that converts Platinum SDK templates to this format.

Quick Start 2

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.

The 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 131* and *DPFPVerificationControl Object User Interface* on *page 140*.

Quick Concepts

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

Enrollment—The process of capturing a person's fingerprint four times, extracting the features from the fingerprints, creating a fingerprint template, and storing the template for later comparison.

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

Unenrollment—The process of deleting a fingerprint template associated with a previously enrolled fingerprint.

For further descriptions of these processes, see Chapter 4 on *page 17*.

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 DigitalPersona One Touch for Windows Runtime Environment (RTE).

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

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

Connect the Fingerprint Reader

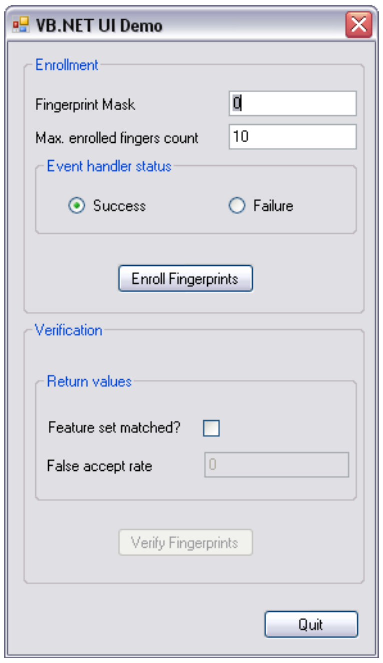
Connect the fingerprint reader into the USB connector on the system where you installed the SDK.

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



VB.NET UI Demo

Enrollment

Fingerprint Mask: 0

Max. enrolled fingers count: 10

Event handler status: [✓ Success] Failure

Verification

Return values

Feature set matched? ☐

False accept rate: 0

1. Open the UIVBDemo.exe file -

   It is located in the <*destination folder*>One Touch SDK\COM-ActiveX\Samples\VB6\UI Support folder.

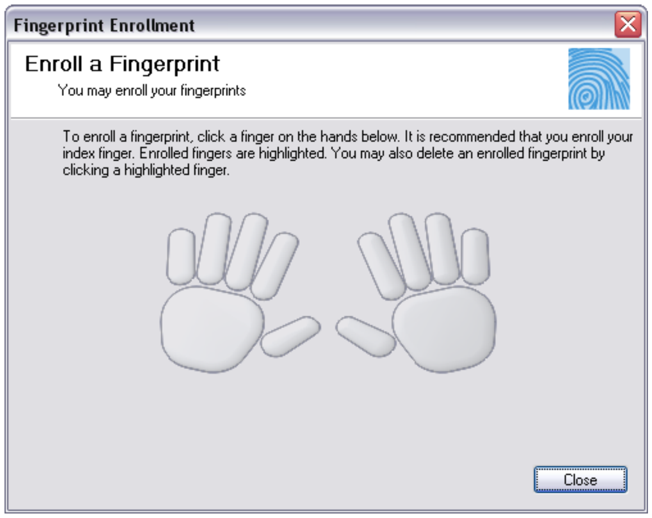
2. The **VB Demo** dialog box appears.

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 Fingerprints**.

   The **Fingerprint Enrollment** dialog box appears.



Fingerprint Enrollment

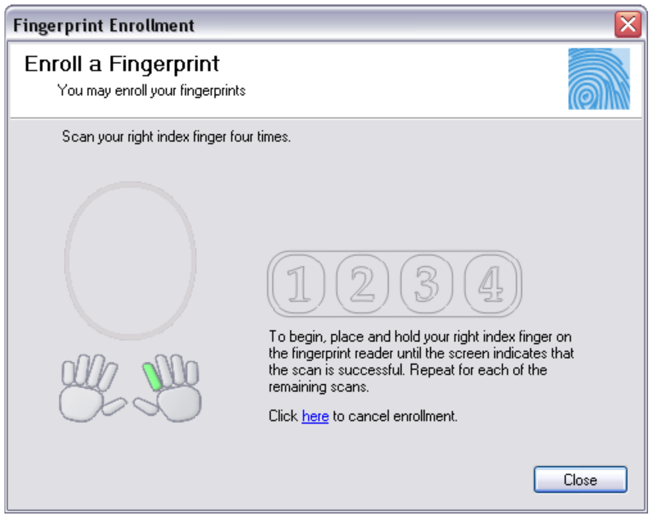
Enroll a Fingerprint

You may enroll your fingerprints

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.

2. In the right "hand," click the index finger.

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



Fingerprint Enrollment

Enroll a Fingerprint

You may enroll your fingerprints

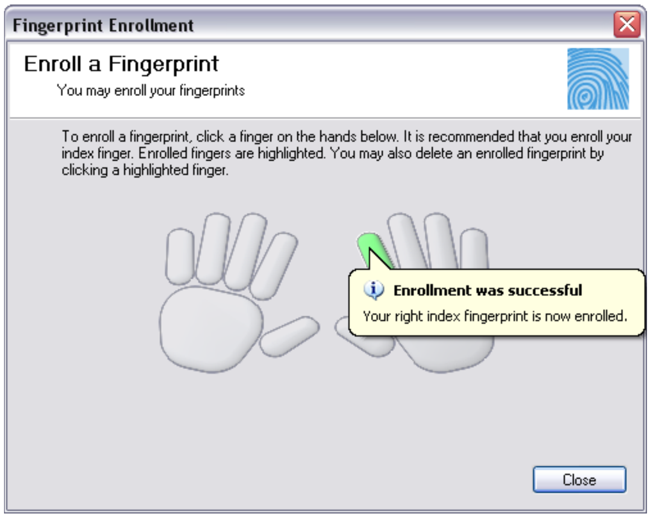
Scan your right index finger four times.

To begin, place and hold your right index finger on the fingerprint reader until the screen indicates that the scan is successful. Repeat for each of the remaining scans.

Click here to cancel enrollment.

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

4. Repeat step 3 until the **Enrollment was successful** message appears.



Enroll a Fingerprint

You may enroll your fingerprints

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.

Enrollment was successful

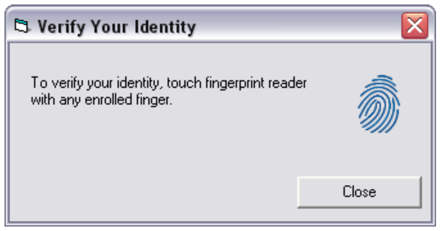
Your right index fingerprint is now enrolled.

5. Click **Close**.

To verify a fingerprint

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

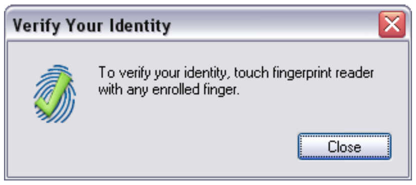
   The **Verify Your Identify** dialog box appears.



To verify your identity, touch fingerprint reader with any enrolled finger.

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

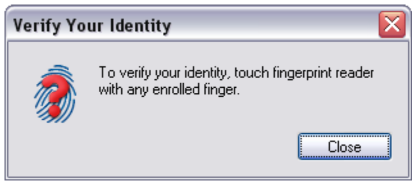
   In the **Verify Your Identify** dialog box, a green check mark appears over the fingerprint, which indicates that your fingerprint was verified.



To verify your identity, touch fingerprint reader with any enrolled finger.

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

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



Verify Your Identity

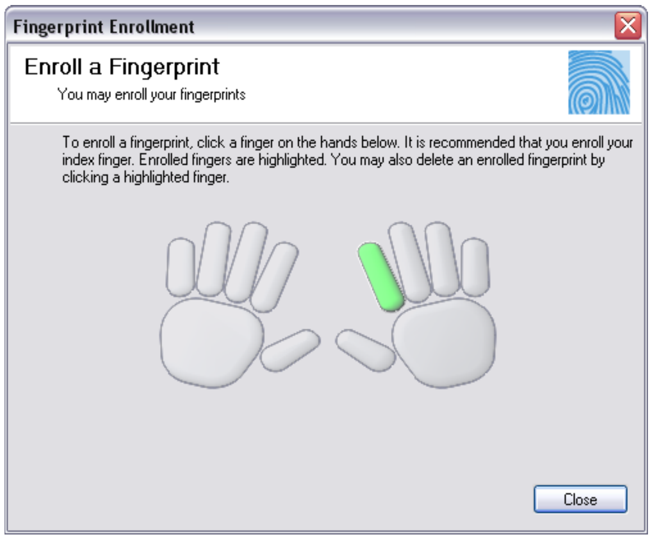
To verify your identity, touch fingerprint reader with any enrolled finger.

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.



Fingerprint Enrollment

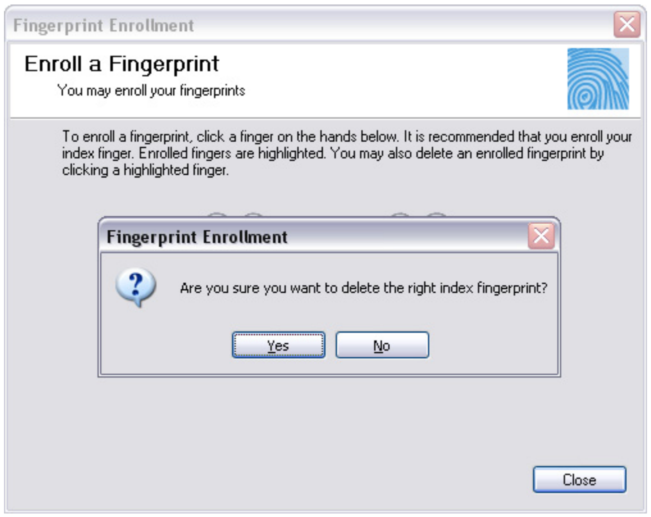
Enroll a Fingerprint

You may enroll your fingerprints

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.

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

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

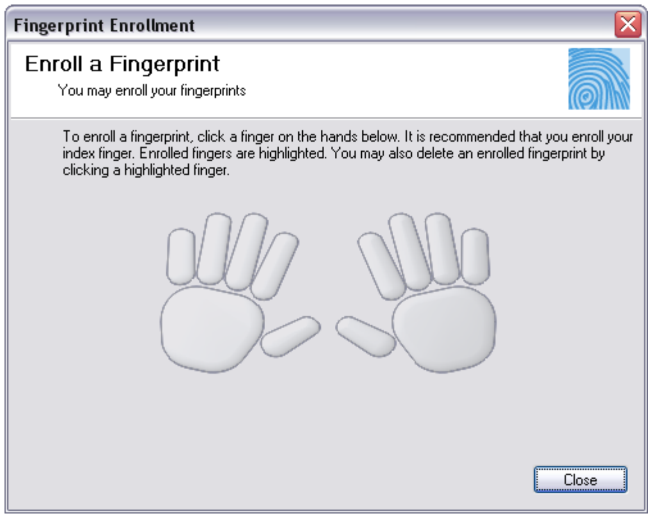


To enroll a fingerprint, click a finger on the hands below. It is recommended that you enroll your index finger. Enrolled fingers are highlighted. You may also delete an enrolled fingerprint by clicking a highlighted finger.

Are you sure you want to delete the right index fingerprint?

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 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.

To exit the application

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

Installation 3

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 in the SDK software package:

- SDK, which you use in developing your application. This installation is located in the SDK folder.
- RTE (runtime environment), which you must provide to your end users to implement the One Touch for Windows SDK: COM/ActiveX Edition components. This installation is located in the RTE folder. (The RTE installation is also included in the SDK installation.)

Installing the SDK

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.

To install the One Touch for Windows SDK: COM/ActiveX Edition for 32-bit operating systems

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

To install the One Touch for Windows SDK: COM/ActiveX Edition for 64-bit operating systems

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

Table 1 describes the files and folders that are installed in the <*destination folder*> folder on your hard disk for the 32-bit and 64-bit installations. The RTE files and folders, which are described in Table 2 on *page 14* for the 32-bit installation and in Table 3 on *page 15* for the 64-bit installation, are also installed on your hard disk.

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 in the SDK software package to create your own MSI installer. (See *Redistribution* on *page 142* 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.

To install the One Touch for Windows: COM/ActiveX Edition RTE for 32-bit operating systems

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

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

Table 2. One Touch for Windows: COM/ActiveX Edition RTE installed files and folders, 32-bit installation

Folder	File	Description
<*destination folder*>\Bin	DPCOper2.dll DPDevice2.dll DPDevTS.dll DpHostW.exe DPmsg.dll DPMux.dll DpSvInfo2.dll DPTSClnt.dll DPCrStor.dll	DLLs and executable file used by the all of the One Touch for Windows APIs
<*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

To install the One Touch for Windows: COM/ActiveX Edition Runtime Environment for 64-bit operating systems

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

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

Table 3. One Touch for Windows: COM/ActiveX Edition RTE installed files and folders, 64-bit installation

Folder	**File**	**Description**
<destination folder>\Bin	DPCOper2.dll DPDevice2.dll DPDevTS.dll DpHostW.exe DPMux.dll DpSvInfo2.dll DPTSClnt.dll DPCrStor.dll	DLLs and executable file used by the all of the One Touch for Windows APIs
<destination folder>\Bin\x64	DPmsg.dll	DLL used by the all of the One Touch for Windows APIs
<destination folder>\Bin\ActiveX	DPFPShrX.dll DPFPEngX.dll DPFPDevX.dll DPFPCtlX.dll	32-bit DLLs used by the One Touch for Windows: COM/ActiveX Edition API
<destination folder>\Bin\ActiveX\x64	DPFPShrX.dll DPFPEngX.dll DPFPDevX.dll DPFPCtlX.dll	64-bit DLLs used by the One Touch for Windows: COM/ActiveX Edition API
<system folder>	DPFPApi.dll DpClback.dll dpHFtrEx.dll dpHMatch.dll DPFpUI.dll	32-bit DLLs used by all of the One Touch for Windows APIs
<system64 folder>	DPFPApi.dll DpClback.dll dpHFtrEx.dll dpHMatch.dll DPFpUI.dll	64-bit DLLs used by all of the One Touch for Windows APIs

Installing and Uninstalling the RTE Silently

The One Touch for Windows SDK software package contains a batch file, InstallOnly.bat, that you can use to silently install the RTE. In addition, you can modify the file to selectively install the various features of the RTE. Refer to the file for instructions.

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

Overview 4

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 in the One Touch for Windows SDK software package.) 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 on *page 19*. 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.

Fingerprint Enrollment

Fingerprint enrollment is the initial process of collecting *fingerprint data* from an *enrollee* and storing the resulting data as a *fingerprint template* for later comparison. The following procedure describes typical fingerprint enrollment. (Steps preceded by an asterisk are not performed by the One Touch for Windows SDK: COM/ActiveX Edition.)

1. *Obtain the enrollee's identifier (*Subject Identifier*).
2. Capture the enrollee's fingerprint using the fingerprint reader.
3. Extract the *fingerprint feature set* for the purpose of enrollment from the fingerprint sample.
4. Repeat steps 2 and 3 until you have enough fingerprint feature sets to create a fingerprint template.
5. Create a fingerprint template.
6. *Associate the fingerprint template with the enrollee through a Subject Identifier, such as a user name, email address, or employee number.
7. *Store the fingerprint template, along with the Subject Identifier, 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 local or central database.

Fingerprint Verification

Fingerprint verification is the process of comparing the fingerprint data to the fingerprint template produced at enrollment and deciding if the two match. The following procedure describes typical fingerprint verification. (Steps preceded by an asterisk are not performed by the One Touch for Windows SDK: COM/ActiveX Edition.)

1. *Obtain the Subject Identifier of the person to be verified.
2. Capture a fingerprint sample using the fingerprint reader.
3. Extract a fingerprint feature set for the purpose of verification from the fingerprint sample.
4. *Retrieve the fingerprint template associated with the Subject Identifier from your repository.

5. Perform a *one-to-one comparison* between 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 the building for a non-match.

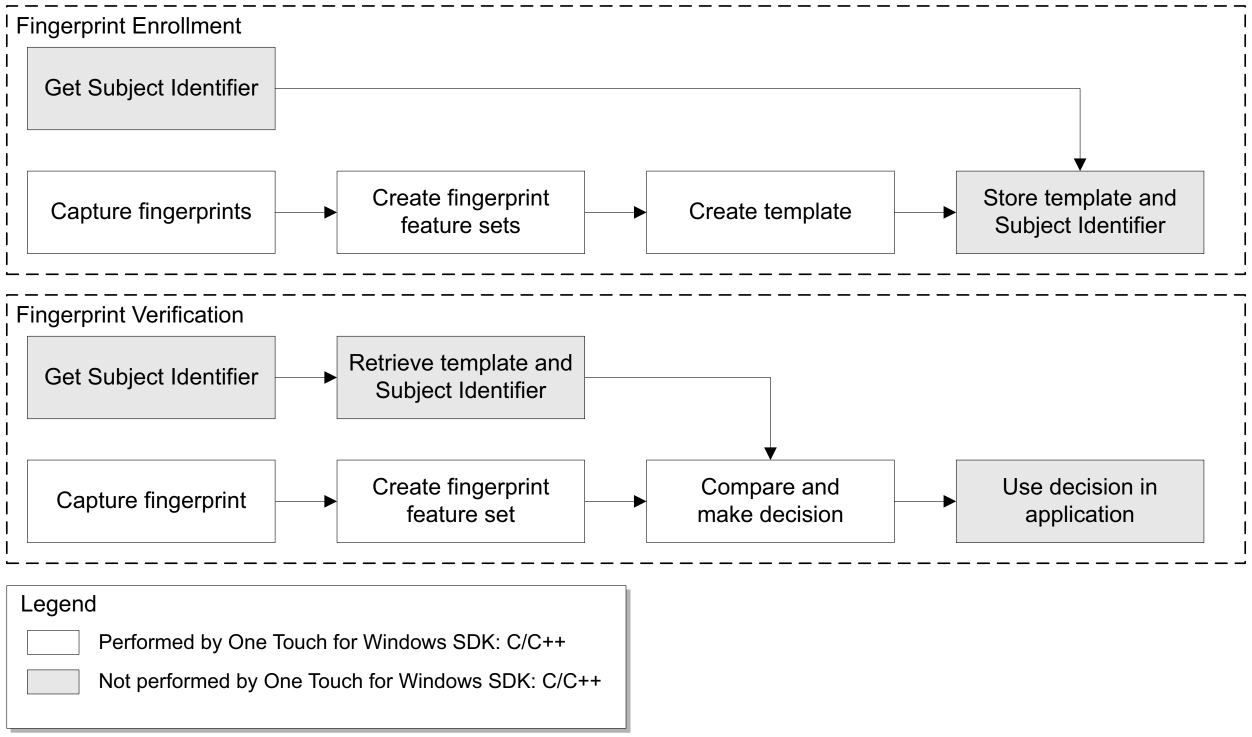

Fingerprint Enrollment
Get Subject Identifier
Capture fingerprints
Create fingerprint feature sets
Create template
Store template and Subject Identifier
Fingerprint Verification
Get Subject Identifier
Retrieve template and Subject Identifier
Capture fingerprint
Create fingerprint feature set
Compare and make decision
Use decision in application
Legend
Performed by One Touch for Windows SDK: C/C++
Not performed by One Touch for Windows SDK: C/C++

Figure 1. DigitalPersona fingerprint recognition system

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 two asterisks (**) 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.

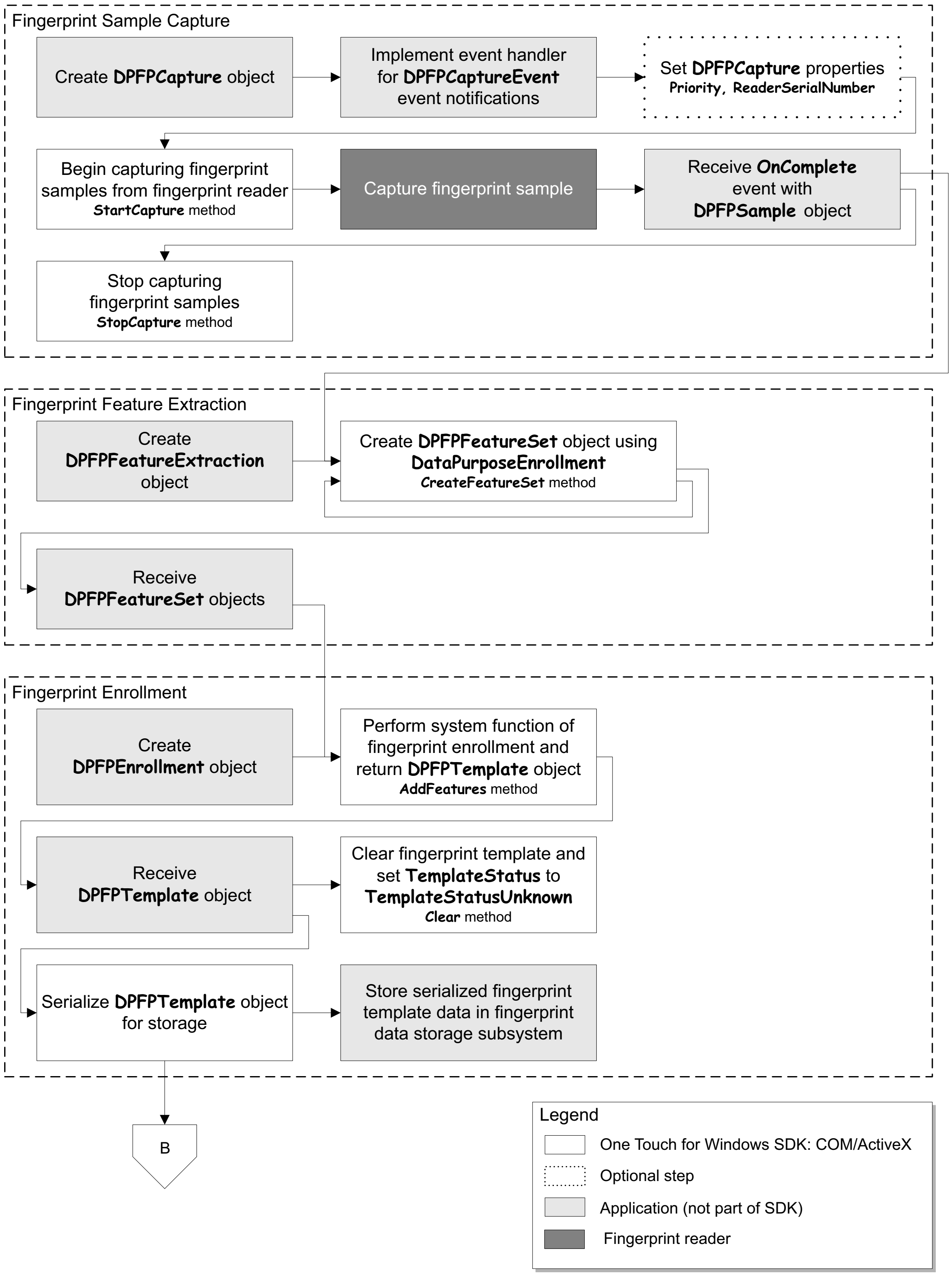

Fingerprint Sample Capture
Create DPFPCapture object
Implement event handler for DPFPCaptureEvent event notifications
Set DPFPCapture properties Priority, ReaderSerialNumber
Begin capturing fingerprint samples from fingerprint reader StartCapture method
Capture fingerprint sample
Receive OnComplete event with DPFPSample object
Stop capturing fingerprint samples StopCapture method
Fingerprint Feature Extraction
Create DPFPFeatureExtraction object
Create DPFPFeatureSet object using DataPurposeEnrollment CreateFeatureSet method
Receive DPFPFeatureSet objects
Fingerprint Enrollment
Create DPFPEnrollment object
Perform system function of fingerprint enrollment and return DPFPTemplate object AddFeatures method
Receive DPFPTemplate object
Clear fingerprint template and set TemplateStatus to TemplateStatusUnknown Clear method
Serialize DPFPTemplate object for storage
Store serialized fingerprint template data in fingerprint data storage subsystem
B
Legend
One Touch for Windows SDK: COM/ActiveX
Optional step
Application (not part of SDK)
Fingerprint reader

Figure 2. Typical fingerprint enrollment workflow

Fingerprint Sample Capture

1. *Create an instance of a **`DPFPCapture`** object (VB *page 36*, C++ *page 78*).
2. *Implement an event handler for **`DPFPCaptureEvents`** event notifications (VB *page 38*, C++ *page 81*).
3. Optionally, set the **`Priority`** and **`ReaderSerialNumber`** properties (VB *page 36* and *page 37*; C++ *page 78* and *page 79*).
4. Begin capturing fingerprint samples from the fingerprint reader by calling the **`StartCapture`** method (VB *page 36*, C++ *page 79*).
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 38*, C++ *page 81*).
7. *Pass the **`DPFPSample`** object to the **`DPFPFeatureExtraction`** method. (See step 2 in the next section.)
8. Stop capturing fingerprint samples by calling the **`StopCapture`** method (VB *page 36*, C++ *page 80*).

Fingerprint Feature Extraction

1. *Create an instance of a **`DPFPFeatureExtraction`** object (VB *page 51*, C++ *page 97*).
2. Create **`DPFPFeatureSet`** objects by calling the **`CreateFeatureSet`** method using the value **`DataPurposeEnrollment`** and passing a **`DPFPSample`** object from step 7 of the previous section (VB *page 52*, C++ *page 97*).
3. *Pass the **`DPFPFeatureSet`** objects created in the previous step to the **`AddFeatures`** method.

Fingerprint Enrollment

1. *Create an instance of a **`DPFPEnrollment`** object (VB *page 41*, C++ *page 85*).
2. Perform the system function of fingerprint enrollment by calling the **`AddFeatures`** method and passing the **`DPFPFeatureSet`** objects (VB *page 41*, C++ *page 85*).

   When the **`TemplateStatus`** property returns the value **`TemplateStatusReady`**, a **`DPFPTemplate`** object is created (VB *page 42*, C++ *page 86*).
3. *Receive the **`DPFPTemplate`** object.
4. Serialize the **`DPFPTemplate`** object (see *Serializing a Fingerprint Data Object* on *page 32*).
5. *Store the serialized fingerprint template data in a fingerprint data storage subsystem.

6. Clear the fingerprint template and set the value of **`TemplateStatus`** to **`TemplateStatusUnknown`** by calling the **`Clear`** method (VB *page 41*, C++ *page 85*).

Fingerprint Enrollment with UI Support

This section contains two *typical* workflows for performing fingerprint enrollment: one for enrolling a fingerprint and one for unenrolling (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

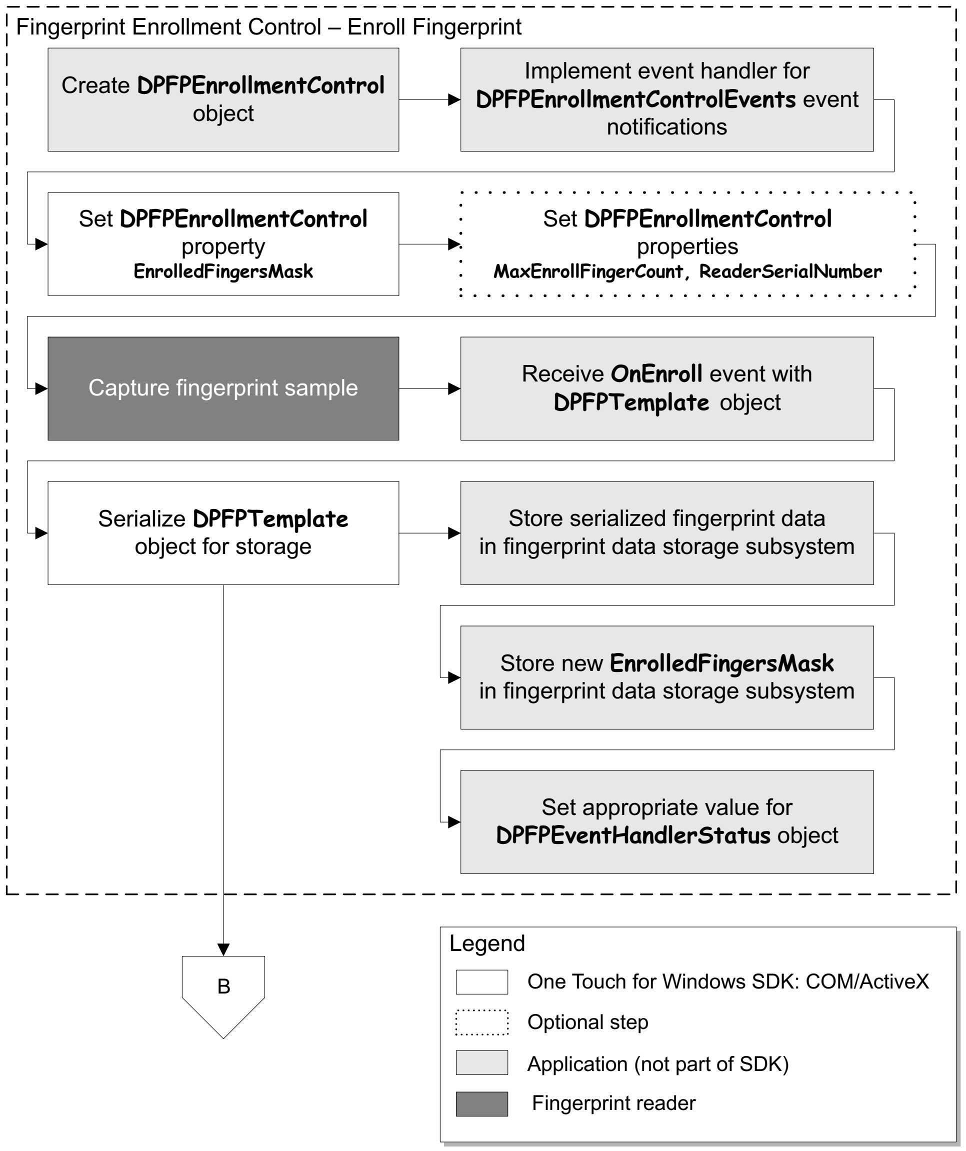

Fingerprint Enrollment Control – Enroll Fingerprint
Create DPFPEnrollmentControl object
Implement event handler for DPFPEnrollmentControlEvents event notifications
Set DPFPEnrollmentControl property EnrolledFingersMask
Set DPFPEnrollmentControl properties MaxEnrollFingerCount, ReaderSerialNumber
Capture fingerprint sample
Receive OnEnroll event with DPFPTemplate object
Serialize DPFPTemplate object for storage
Store serialized fingerprint data in fingerprint data storage subsystem
Store new EnrolledFingersMask in fingerprint data storage subsystem
Set appropriate value for DPFPEventHandlerStatus object
B
Legend
One Touch for Windows SDK: COM/ActiveX
Optional step
Application (not part of SDK)
Fingerprint reader

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

1. *Create an instance of a `DPFPEnrollmentControl` object (VB *page 43*, C++ *page 87*).
2. *Implement an event handler for `DPFPEnrollmentControlEvents` event notifications (VB *page 46*, C++ *page 91*).
3. Set the `EnrolledFingersMask` property (VB *page 43*, C++ *page 87*).

4. Optionally, set the `MaxEnrollFingerCount` and `ReaderSerialNumber` properties (VB *page 44* and *page 45*; C++ *page 89* and *page 89*).
5. **Capture a fingerprint sample from a fingerprint reader.
6. *Receive the `OnEnroll` event and the `DPFPTemplate` object (VB *page 47*, C++ *page 93*).
7. Serialize the `DPFPTemplate` object (see *Serializing a Fingerprint Data Object* on *page 32*).
8. *Store the serialized fingerprint template data and the new value of the `EnrolledFingersMask` in a fingerprint data storage subsystem.
9. *Set the appropriate value for the `DPFPEventHandlerStatus` object (VB *page 71*, C++ *page 126*).

Unenrolling (Deleting) a Fingerprint Template

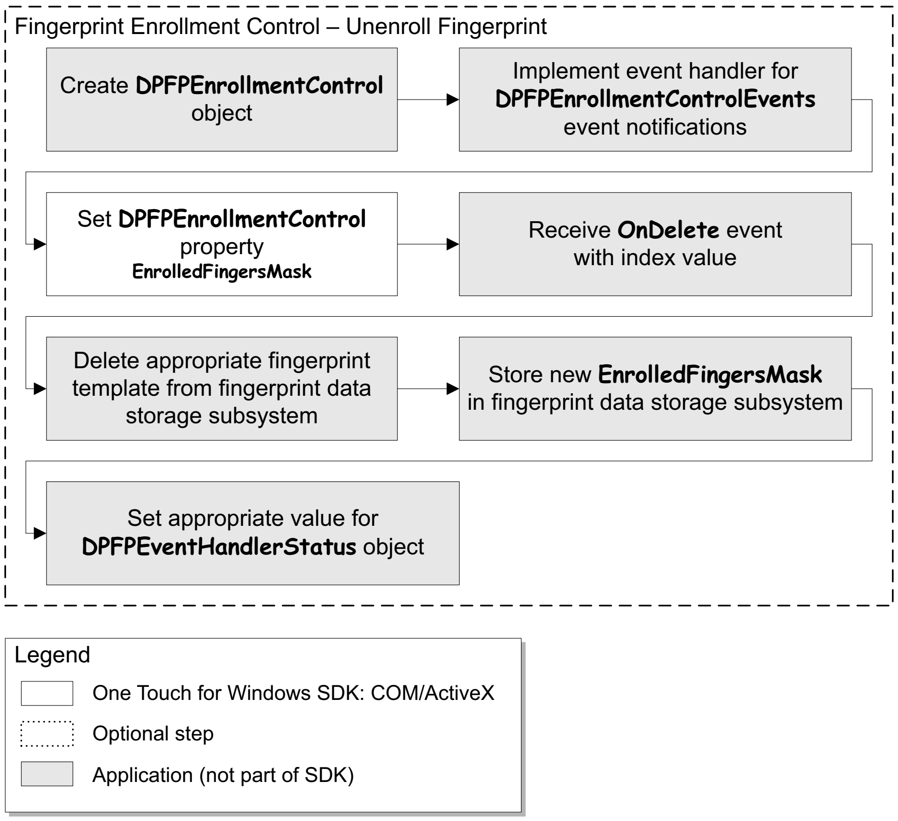

Fingerprint Enrollment Control – Unenroll Fingerprint
Create DPFPEnrollmentControl object
Implement event handler for DPFPEnrollmentControlEvents event notifications
Set DPFPEnrollmentControl property EnrolledFingersMask
Receive OnDelete event with index value
Delete appropriate fingerprint template from fingerprint data storage subsystem
Store new EnrolledFingersMask in fingerprint data storage subsystem
Set appropriate value for DPFPEventHandlerStatus object
Legend
One Touch for Windows SDK: COM/ActiveX
Optional step
Application (not part of SDK)

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

1. *Create an instance of a `DPFPEnrollmentControl` object (VB *page 43*, C++ *page 87*).
2. *Implement an event handler for `DPFPEnrollmentControlEvents` event notifications (VB *page 46*, C++ *page 91*).
3. Set the `EnrolledFingersMask` property (VB *page 43*, C++ *page 87*).
4. *Receive the `OnDelete` event, along with the finger index value (VB *page 47* and *page 44*; C++ *page 92* and *page 92*).
5. *Delete the appropriate fingerprint template from the fingerprint data storage subsystem.
6. *Store the new value of the `EnrolledFingersMask` in the fingerprint data storage subsystem.
7. *Set the appropriate value for the `DPFPEventHandlerStatus` object (VB *page 71*, C++ *page 126*).

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.

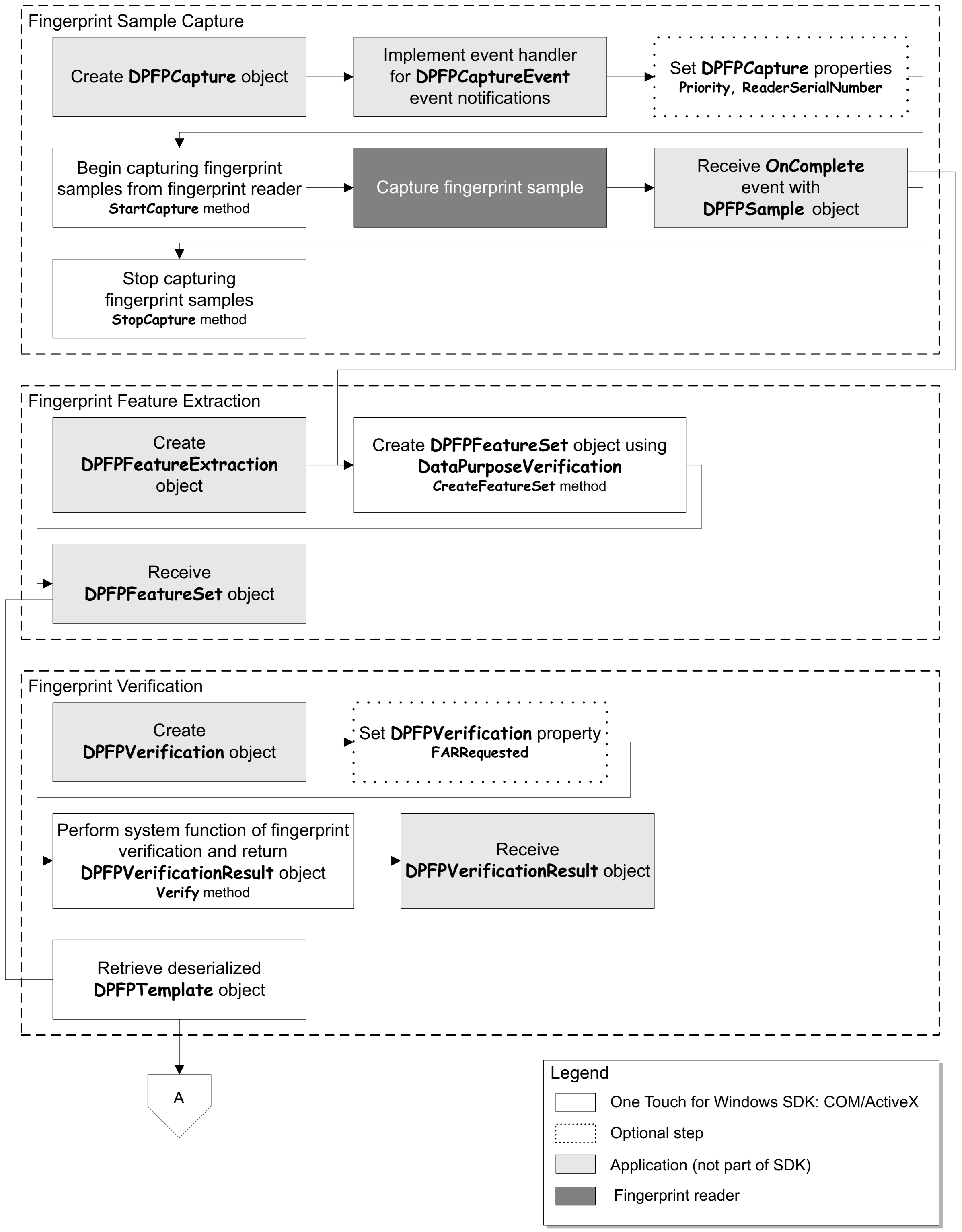

Fingerprint Sample Capture
Create DPFPCapture object
Implement event handler for DPFPCaptureEvent event notifications
Set DPFPCapture properties Priority, ReaderSerialNumber
Begin capturing fingerprint samples from fingerprint reader StartCapture method
Capture fingerprint sample
Receive OnComplete event with DPFPSample object
Stop capturing fingerprint samples StopCapture method
Fingerprint Feature Extraction
Create DPFPFeatureExtraction object
Create DPFPFeatureSet object using DataPurposeVerification CreateFeatureSet method
Receive DPFPFeatureSet object
Fingerprint Verification
Create DPFPVerification object
Set DPFPVerification property FARRequested
Perform system function of fingerprint verification and return DPFPVerificationResult object Verify method
Receive DPFPVerificationResult object
Retrieve deserialized DPFPTemplate object
A
Legend
One Touch for Windows SDK: COM/ActiveX
Optional step
Application (not part of SDK)
Fingerprint reader

Figure 5. Typical fingerprint verification workflow

Fingerprint Sample Capture

1. *Create an instance of a `DPFPCapture` object (VB *page 36*, C++ *page 78*).
2. *Implement an event handler for `DPFPCaptureEvents` event notifications (VB *page 38*, C++ *page 81*).
3. Optionally, set the `Priority` and `ReaderSerialNumber` properties (VB *page 36* and *page 37*; C++ *page 78* and *page 79*).
4. Begin capturing fingerprint samples from the fingerprint reader by calling the `StartCapture` method (VB *page 36*, C++ *page 79*).
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 38*, C++ *page 81*).
7. *Pass the `DPFPSample` object to the `CreateFeatureSet` method. (See step 2 in the next section.)
8. Stop capturing fingerprint samples by calling the `StopCapture` method (VB *page 36*, C++ *page 80*).

Fingerprint Feature Extraction

1. *Create an instance of a `DPFPFeatureExtraction` object (VB *page 51*, C++ *page 97*).
2. Create a `DPFPFeatureSet` object by calling the `CreateFeatureSet` method using the value `DataPurposeVerification` and passing a `DPFPSample` object from step 7 in the previous section (VB *page 52*, C++ *page 97*).
3. *Pass the `DPFPFeatureSet` object created in the previous step to the `Verify` method. (See step 5 in the next section.)

Fingerprint Verification

1. *Create an instance of a `DPFPVerification` object (VB *page 63*, C++ *page 115*).
2. Optionally, set the `FARRequested` property (VB *page 63*, C++ *page 115*). You can use this property to check or modify the current value of the FAR.
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 33*).
5. Perform the system function of fingerprint verification by calling the `Verify` method and passing the `DPFPTemplate` object created in the previous step and `DPFPFeatureSet` object from step 3 in the previous section (VB *page 63*, C++ *page 116*).

6. *Receive the `DPFPVerificationResult` object, which provides the comparison decision of match or non-match (VB *page 67*, C++ *page 121*).

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.

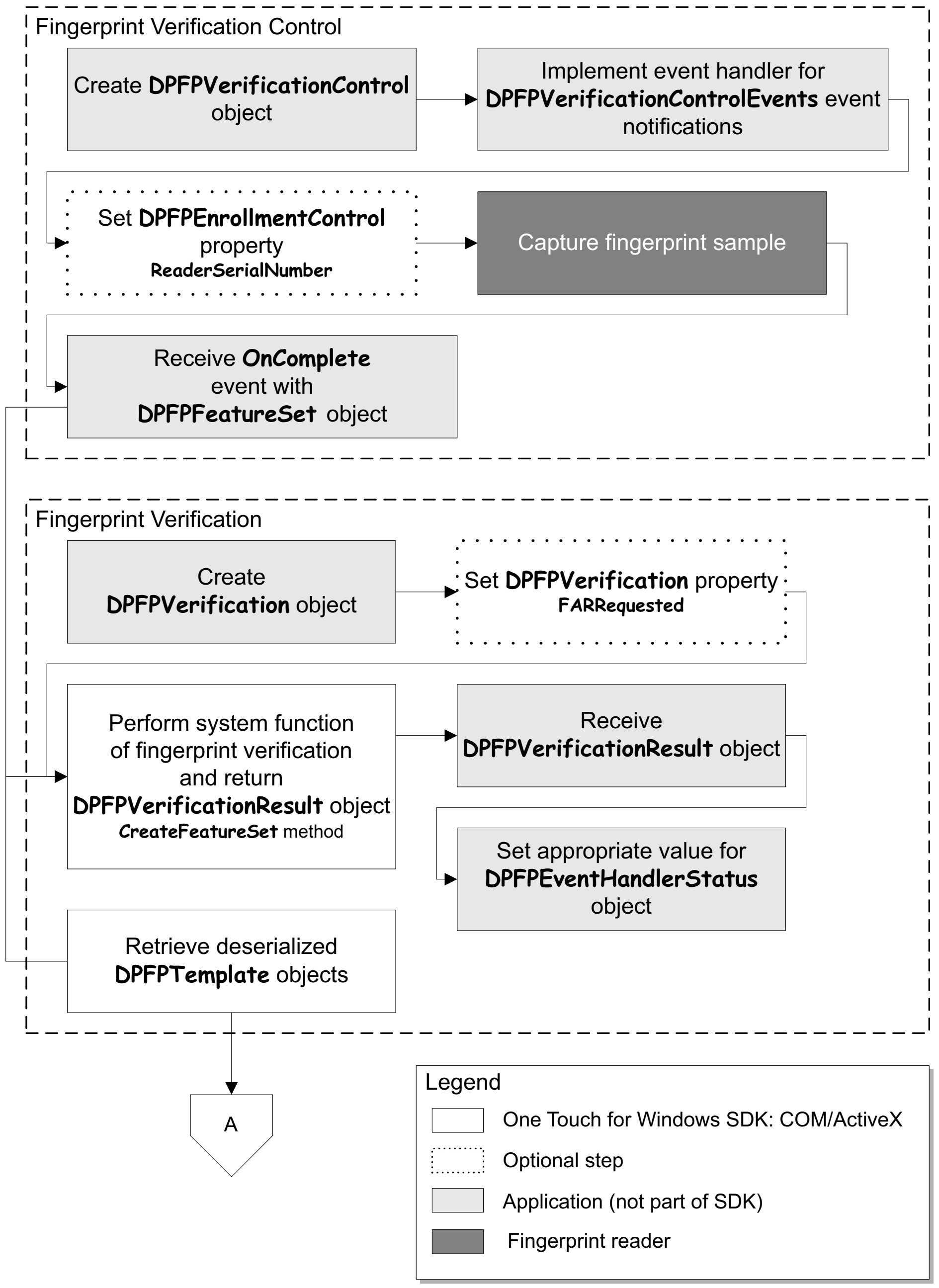

Fingerprint Verification Control
Create DPFPVerificationControl object
Implement event handler for DPFPVerificationControlEvents event notifications
Set DPFPEnrollmentControl property ReaderSerialNumber
Capture fingerprint sample
Receive OnComplete event with DPFPFeatureSet object
Fingerprint Verification
Create DPFPVerification object
Set DPFPVerification property FARRequested
Perform system function of fingerprint verification and return DPFPVerificationResult object CreateFeatureSet method
Receive DPFPVerificationResult object
Set appropriate value for DPFPEventHandlerStatus object
Retrieve deserialized DPFPTemplate objects
A
Legend
One Touch for Windows SDK: COM/ActiveX
Optional step
Application (not part of SDK)
Fingerprint reader

Figure 6. Typical fingerprint verification with UI support workflow

Fingerprint Verification Control

1. *Create an instance of a `DPFPVerificationControl` object (VB *page 64*, C++ *page 118*).
2. Implement an event handler for `DPFPVerificationControlEvents` event notifications (VB *page 65*, C++ *page 120*).
3. Optionally, set the `ReaderSerialNumber` property (VB *page 65*, C++ *page 119*).
4. **Capture a fingerprint sample from a fingerprint reader.
5. Receive the `OnComplete` event with the `DPFPFeatureSet` object (VB *page 65*, C++ *page 120*).

Fingerprint Verification

1. *Create an instance of a `DPFPVerification` object (VB *page 63*, C++ *page 115*).
2. Optionally, set the `FARRequested` property (VB *page 63*, C++ *page 115*). You can use this property to check or modify the current value of the FAR.
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 33*).
5. Perform the system function of fingerprint verification by calling the `Verify` method and passing the `DPFPTemplate` and `DPFPFeatureSet` objects (VB *page 63*, C++ *page 116*).
6. *Receive the `DPFPVerificationResult` object, which provides the comparison decision of match or non-match (VB *page 67*, C++ *page 121*).
7. *Set the appropriate value for the `DPFPEventHandlerStatus` object (VB *page 71*, C++ *page 126*).

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

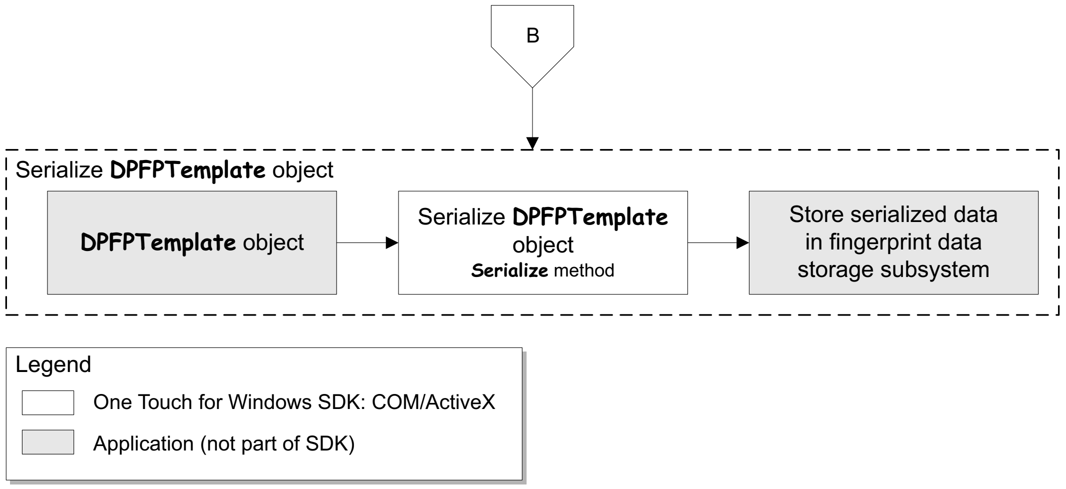

B
Serialize DPFPTemplate object
DPFPTemplate object
Serialize DPFPTemplate object
Serialize method
Store serialized data in fingerprint data storage subsystem
Legend
One Touch for Windows SDK: COM/ActiveX
Application (not part of SDK)

Figure 7. Fingerprint data object serialization workflow: `DPFPTemplate` object

1. *Begin with a `DPFPTemplate` object. (See the various methods and properties for creating and returning a `DPFPTemplate` object.)
2. Serialize the data object by calling the `Serialize` method (VB *page 40*, C++ *page 84*).
3. *Store the serialized data in a fingerprint data storage subsystem.

Deserializing a Serialized Fingerprint Data Object

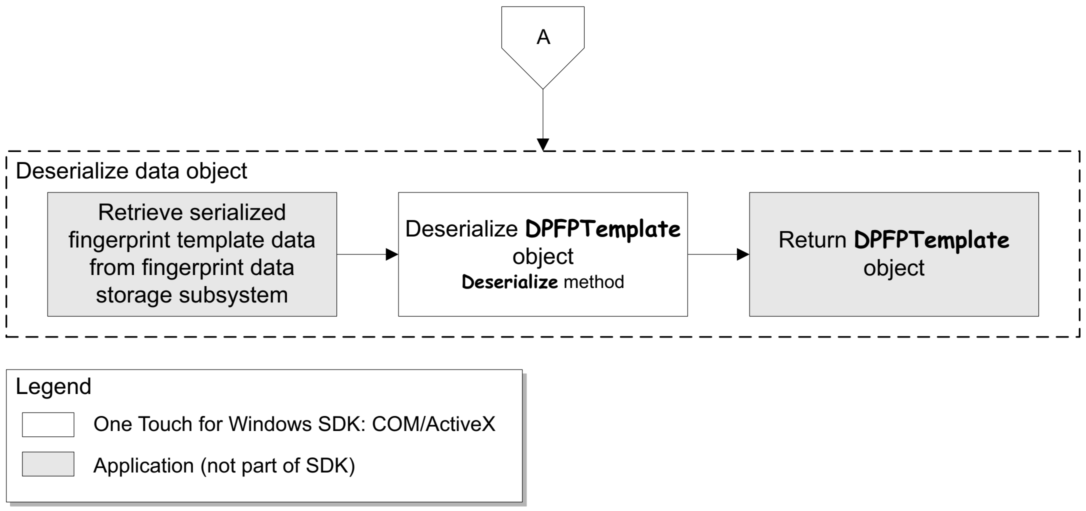

A
Deserialize data object
Retrieve serialized fingerprint template data from fingerprint data storage subsystem
Deserialize DPFPTemplate object
Deserialize method
Return DPFPTemplate object
Legend
One Touch for Windows SDK: COM/ActiveX
Application (not part of SDK)

Figure 8. Deserialization of serialized fingerprint data object workflow: `DPFPTemplate` object

1. *Retrieve serialized fingerprint template data from a fingerprint data storage subsystem.
2. Deserialize a `DPFPTemplate` object by calling the `Deserialize` method (VB *page 40*, C++ *page 83*).
3. Return a `DPFPTemplate` object.

API Reference for Visual Basic Developers 5

This chapter defines the API components 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 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.

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
`DPFPCapture`	*36*	Captures a fingerprint sample from a fingerprint reader
`DPFPData`	*40*	Represents the data that is common to all fingerprint data objects
`DPFPEnrollment`	*41*	Performs the system function of fingerprint enrollment
`DPFPEnrollmentControl`	*43*	Contains an ActiveX control for performing fingerprint enrollment operations, and provides a user interface
`DPFPEventHandlerStatus`	*51*	Returns codes that indicate the status of an operation
`DPFPFeatureExtraction`	*51*	Performs the system function of fingerprint feature extraction
`DPFPFeatureSet`	*53*	Represents a fingerprint feature set
`DPFPReaderDescription`	*54*	Provides information about a particular fingerprint reader
`DPFPReadersCollection`	*58*	Provides information about all of the fingerprint readers connected to a system
`DPFPSample`	*60*	Represents a fingerprint sample
`DPFPSampleConversion`	*61*	Returns a fingerprint sample as an image
`DPFPTemplate`	*62*	Represents a fingerprint template
`DPFPVerification`	*63*	Performs the system function of fingerprint verification

Method	**Page**	**Description**
`DPFPVerificationControl`	*64*	Contains an ActiveX control for creating and returning a fingerprint feature set created for the purpose of verification, and provides a user interface
`DPFPVerificationResult`	*67*	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 **`DPFPCapturePriorityEnum`** on *page 70* for more information.
`-2147024891`	General access denied error.	The application does not have sufficient privileges to start capture operations with the specified priority. See **`DPFPCapturePriorityEnum`** on *page 70* 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

`enumValue`	**Enum** that specifies or receives one of the `DPFPCapturePriorityEnum` enumeration values (*page 70*)

This optional property is read/write. If you do not set it, the value `CapturePriorityNormal` 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

`strValue`	**String** 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

`bstrReaderSerNum`	**String** that specifies a fingerprint reader serial number
`oFingerprintSample`	A `DPFPSample` object (*page 60*)

OnFingerGone Event

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

Syntax

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

Parameter

`bstrReaderSerNum`	**String** 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

`bstrReaderSerNum`	**String** 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

`bstrReaderSerNum`	**String** 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

`bstrReaderSerNum`	**String** 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 38*).

Syntax

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

Parameters

`bstrReaderSerNum`	**String** that specifies a fingerprint reader serial number
`enumSampleQuality`	**Enum** that specifies one of the values, which provides feedback about a fingerprint sample capture operation, from the `DPFPCaptureFeedbackEnum` enumeration (*page 69*)

DPFPData

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

Methods

Deserialize Method

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

Syntax

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

Parameter

`aRawData`	**Array of bytes** 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 **`Deserialize`** 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

`aRawData`	**Array of bytes** 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 53*

`DPFPSample` on *page 60*

`DPFPTemplate` on *page 62*

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

`oFeatures`	A **`DPFPFeatureSet`** object (*page 53*)

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

`lValue`	**Long** 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

`oTemplate`	A `DPFPTemplate` object (*page 62*)

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

Possible Errors

Error Code	Message	Description
`-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

`enumValue`	**Enum** that receives one of the `DPFPTemplateStatusEnum` enumeration values (*page 75*)

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 131*) and provides the following functionality:

- Captures fingerprint samples from a fingerprint reader(s)
- Creates fingerprint feature sets for the purpose of enrollment
- Creates fingerprint templates
- Notifies an application when an enrollee commits to delete 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

`lValue`	**Long** that specifies or receives the value of the fingerprint mask. All possible values are listed in *Table 4*.

Table 4. 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

`lValue`	**Long** that specifies or receives the value for the maximum number of fingerprints that can be enrolled. Possible values are `1` through `10`.

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

`bstrValue`	**String** 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, {A9EFB3F6-A8C8-4684-841E-4330973057C6}.

Object Information

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

Events

OnCancelEnroll Event

Fires when enrollment is cancelled.

Syntax

```
Private Sub object_OnCanceEnroll(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

`pSerialNumber`	[in] Variable of type String that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

The `l1FingerMask` 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 5*.

Table 5. 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 ring finger	4	Left ring finger	9
Right little finger	5	Left little finger	10

OnComplete Event

Fires on a successful scan.

Syntax

```
Private Sub object_OnComplete(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

`pSerialNumber`	[in] Variable of type String that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

OnDelete Event

Fires when a user commits to delete an enrolled fingerprint. 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

`l1FingerMask`	**Long** that specifies the index value of the (enrolled) fingerprint to be deleted. For possible values, see *Table 5*.
`oStatus`	A **`DPFPEventHandlerStatus`** object (*page 51*)

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 l1FingerMask As Long,
  ByVal oFingerprintTemplate As Object,
  ByVal oStatus As Object)
```

Parameters

`l1FingerMask`	**Long** that specifies the index value for the enrolled fingerprint. For possible values, see Table 5 on *page 46*.
`oFingerprintTemplate`	A **`DPFPTemplate`** object (*page 62*)
`oStatus`	A **`DPFPEventHandlerStatus`** object (*page 51*)

OnFingerRemove Event

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

Syntax

```
Private Sub object_OnFingerRemove(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

pSerialNumber	[in] Variable of type String that contains a fingerprint reader serial number.
lEnrolledFinger	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

OnFingerTouch Event

Fires when a user touches a fingerprint reader.

Syntax

```
Private Sub object_OnFingerTouch(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

pSerialNumber	[in] Variable of type String that contains a fingerprint reader serial number.
lEnrolledFinger	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

OnReaderConnect Event

Fires when a reader is connected.

Syntax

```
Private Sub object_OnReaderConnect(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

`pSerialNumber`	[in] Variable of type String that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

OnReaderDisconnect Event

Fires when a reader is disconnected.

Syntax

```
Private Sub object_OnReaderDisconnect(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

`pSerialNumber`	[in] Variable of type String that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

OnSampleQuality Event

Fires when a fingerprint sample is received.

Syntax

```
Private Sub object_OnSampleQuality(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long,
   ByVal lSampleQuality As Long)
```

Parameters

`pSerialNumber`	[in] Variable of type String that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.
`lSampleQuality`	[in] Variable that contains a value providing feedback about a fingerprint sample operation. For possible values, see DPFPCaptureFeedbackEnum on *page 69*.

OnStartEnroll Event

Fires when enrollment has begun.

Syntax

```
Private Sub object_OnStartEnroll(
   ByVal pSerialNumber As String,
   ByVal lEnrolledFinger As Long)
```

Parameters

`pSerialNumber`	[in] Variable of type String that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see *Table 5*.

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 **`DPFPEnrollmentControl`** object (*page 43*) or by a **`DPFPVerificationControl`** object (*page 64*).

Syntax

```
DPFPEventHandlerStatus.Status [ = enumValue ]

[ enumValue = ] DPFPEventHandlerStatus.Status
```

Possible Values

`enumValue`	**Enum** that specifies or receives one of the values from the **`DPFPEventHandlerStatusEnum`** enumeration (*page 71*)

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	DPFPCtlX.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

`oFingerprintSample`	A `DPFPSample` object (*page 60*)
`enumPurpose`	**Enum** that specifies one of the values, which is for the specified purpose, from the `DPFPDataPurposeEnum` enumeration (*page 72*)
`enumSampleQuality`	**Enum** the receives one of the values, which provides feedback about a fingerprint sample capture operation, from the `DPFPCaptureFeedbackEnum` enumeration (*page 69*)

Property

FeatureSet Property

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

Syntax

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

Possible Values

`oFeatureSet`	A `DPFPFeatureSet` object (*page 53*)

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

Deserialize Method

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

Syntax

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

Parameter

`aRawData`	**Array of bytes** 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 `Deserialize` 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

`aRawData`	**Array of bytes** that receives a serialized data object

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

`bstrValue`	**String** 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

`bstrValue`	**String** 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

`bstrValue`	**String** the receives the fingerprint reader language. The value of `bstrValue` 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

`enumValue`	**Enum** that receives one of the values from the `DPFPReaderImpressionTypeEnum` enumeration (*page 73*)

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

`bstrValue`	**String** 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

`bstrValue`	**String** 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 `DPFPSerialNumberTypeEnum` enumeration (*page 74*)

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 `DPFPReaderTechnologyEnum` enumeration (*page 73*)

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

`bstrReaderSerialNumber`	**String** that specifies a fingerprint reader serial number
`oReader`	A `DPFPReaderDescription` object (*page 54*)

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

`lCount`	**Long** that receives the total number of `DPFPReaderDescription` 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

`lReader`	**Long** that specifies the index of the **`DPFPReaderDescription`** object to retrieve from the collection. The value of **`lReader`** starts with **`1`**.

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 **`1`** to **`Count`**.

_NewEnum Property

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

Syntax

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

Possible Values

`aReaderEnum`	**IUnknown** that receives the array of **`DPFPReaderDescription`** 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	DPFPDevX.dll

DPFPSample

The **`DPFPSample`** object represents a fingerprint sample captured from a fingerprint reader.

Methods

Deserialize Method

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

Syntax

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

Parameter

`aRawData`	**Array of bytes** 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 **`Deserialize`** 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

`aRawData`	**Array of bytes** that receives a serialized data object

Object Information

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

See Also

`DPFPData` on *page 40*

DPFPSampleConversion

The `SampleConversion` object provides methods for returning a fingerprint sample as an `IPicture` object and as an image in ANSI 381 format.

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

`oSample`	A `DPFPSample` object (*page 60*)
`vAnsi`	**Variant** 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

`oSample`	A `DPFPSample` object (*page 60*)
`oPicture`	An `IPicture` object

Object Information

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

DPFPTemplate

The `DPFPTemplate` object represents a fingerprint template.

Methods

Deserialize Method

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

Syntax

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

Parameter

`aRawData`	**Array of bytes** 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 `Deserialize` 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

`aRawData`	**Array of bytes** that receives a serialized data object

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

`oFeatureSet`	A **`DPFPFeatureSet`** object, where the **`enumPurpose`** parameter of the **`CreateFeatureSet`** method of the **`DPFPFeatureExtraction`** object was set to the value **`DataPurposeVerification`** (*page 52*)
`oTemplate`	A **`DPFPTemplate`** object (*page 62*)
`oVerificationResult`	A **`DPFPVerificationResult`** object (*page 67*)

Properties

FARRequested Property

Gets or sets the requested false accept rate (FAR). For more information about the FAR, see *False Positives and False Negatives* on *page 19*.

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 147* for more information about setting the value of the FAR.

Syntax

```
DPFPVerification.FARRequested [ = lValue ]

[ lValue = ] DPFPVerification.FARRequested
```

Possible Values

`lValue`	**Long** that specifies or 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 check or to modify the current value of the FAR.

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 67*

DPFPVerificationControl

The **`DPFPVerificationControl`** object is an ActiveX control that implements a user interface (described in *DPFPEnrollmentControl Object User Interface* on *page 131*) and provides the following functionality:

- Receives fingerprint reader connect and disconnect event notifications
- Captures fingerprint samples from a fingerprint reader(s)
- Creates fingerprint feature sets for the purpose of verification
- Fires an event

Property

Active Property

Activates or deactivates fingerprint capture. Defaults to True.

Syntax

```
DPFPVerificationControl.Active [ = bolleanVal ]

[ booleanVal = ] DPFPVerificationControl.Active
```

Possible Values

`booleanVal`	**Boolean** that specifies or receives the current capture status.

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

`bstrValue`	**String** that specifies or 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	DPFPCtlX.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.

Syntax

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

Parameters

`oVerificationFeatureSet`	A **`DPFPFeatureSet`** object, which represents a fingerprint feature set created for the purpose of verification (*page 53*)
`oStatus`	A **`DPFPEventHandlerStatus`** object (*page 51*)

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

`lValue`	**Long** 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 149* 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 63*).

Syntax

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

Possible Values

`vbValue`	**Variant** of type **boolean** 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
`DPFPCaptureFeedbackEnum`	*69*	Events returned by a fingerprint reader that provide feedback about a fingerprint sample capture operation
`DPFPCapturePriorityEnum`	*70*	Priority of a fingerprint sample capture operation
`DPFPEventHandlerStatusEnum`	*71*	Codes that are returned by the `DPFPEventHandlerStatus` object to indicate the status of an operation
`DPFPDataPurposeEnum`	*72*	Purpose for which a fingerprint feature set is to be used
`DPFPReaderImpressionTypeEnum`	*73*	Modality that a fingerprint reader uses to capture fingerprint samples
`DPFPReaderTechnologyEnum`	*73*	Fingerprint reader technology
`DPFPSerialNumberTypeEnum`	*74*	Fingerprint reader serial number persistence after reboot
`DPFPTemplateStatusEnum`	*75*	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

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

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

Remarks

The members of this enumeration are called by the **`CreateFeatureSet`** method of the **`DPFPFeatureExtraction`** object (*page 52*) and by the **`OnSampleQuality`** event of the **`DPFPCapture`** object (*page 39*).

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

`CapturePriorityLow`	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.
`CapturePriorityNormal`	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.
`CapturePriorityHigh`	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 36*).

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

`EventHandlerStatusSuccess`	An operation was performed successfully.
`EventHandlerStatusFailure`	An operation failed.

Remarks

The members of this enumeration are called by the **`Status`** property of the **`DPFPEventHandlerStatus`** object (*page 51*).

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

`DataPurposeUnknown`	The purpose is not known.
`DataPurposeVerification`	A fingerprint feature set to be used for the purpose of verification.
`DataPurposeEnrollment`	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 52*).

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

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

Remarks

The members of this enumeration are called by the **`ImpressionType`** property of the **`DPFPReaderDescription`** object (*page 55*).

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

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

Remarks

The members of this enumeration are called by the **`Technology`** property of the **`DPFPReaderDescription`** object (*page 56*).

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

`SerialNumberTypePersistent`	A persistent serial number provided by the hardware.
`SerialNumberTypeVolatile`	A volatile serial number provided by the software.

Remarks

The members of this enumeration are called by the **`SerialNumberType`** property of the **`DPFPReaderDescription`** object (*page 56*).

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

`TemplateStatusUnknown`	The status of a template creation operation is not known, probably because a fingerprint template does not exist yet.
`TemplateStatusInsufficient`	A fingerprint template exists, but more fingerprint feature sets are required to complete it.
`TemplateStatusFailed`	A fingerprint template creation operation failed.
`TemplateStatusReady`	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 42*).

Enumeration Information

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

API Reference for C++ Developers 6

This chapter defines the API components 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
`IDPFPCapture`	*78*	Used by an application to capture a fingerprint sample from a fingerprint reader
`_IDPFPCaptureEvents`	*81*	Designates an event sink interface that an application must implement to receive event notifications from a **`DPFPCapture`** object
`IDPFPData`	*83*	Represents the functionality of the data that is common to all fingerprint data objects
`IDPFPEnrollment`	*85*	Used by an application to perform the system function of fingerprint enrollment
`IDPFPEnrollmentControl`	*87*	Represents the functionality of an ActiveX control for performing fingerprint enrollment operations, and provides a user interface
`_IDPFPEnrollmentControlEvents`	*91*	Designates an event sink interface that an application must implement to receive event notifications from a **`DPFPEnrollmentControl`** object
`IDPFPEventHandlerStatus`	*96*	Used by an application to retrieve codes that indicate the status of an operation
`IDPFPFeatureExtraction`	*97*	Used by an application to perform the system function of fingerprint feature extraction

Interface	Page	Description
`IDPFPFeatureSet`	*99*	Represents the functionality of a fingerprint feature set
`IDPFPReaderDescription`	*101*	Used by an application to obtain information about a particular fingerprint reader connected to a system
`IDPFPReadersCollection`	*106*	Represents a collection of fingerprint readers connected to a system
`IDPFPSample`	*109*	Represents the functionality of a fingerprint sample
`IDPFPSampleConversion`	*111*	Used by an application to return a fingerprint sample as an image
`IDPFPTemplate`	*113*	Represents the functionality of a fingerprint template
`IDPFPVerification`	*115*	Used by an application to perform fingerprint verification
`IDPFPVerificationControl`	*118*	Represents the functionality of an ActiveX control for creating and returning a fingerprint feature set, and provides a user interface
`_IDPFPVerificationControlEvents`	*120*	Designates an event sink interface that an application must implement to receive event notifications from a **`DPFPVerificationControl`** object
`IDPFPVerificationResult`	*121*	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 **`CapturePriorityNormal`** is used.

Syntax

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

Parameters

`pVal`	[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 **`DPFPCapturePriorityEnum`** on *page 125*.
`newVal`	[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**
`DISP_E_OVERFLOW`	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

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

Return Values

Returns `S_OK` if successful, or the following error value otherwise:

Return Value	Message	Description
`E_INVALIDARG`	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 `DPFPCapture` object is destroyed.

Syntax

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

Return Values

Returns `S_OK` if successful, or the following error value otherwise:

Return Value	Message	Description
`E_INVALIDARG`	One or more arguments are invalid.	A capture operation with the specified priority already exists. See `DPFPCapturePriorityEnum` on *page 125* for more information.
`E_ACCESSDENIED`	General access denied error.	The application does not have sufficient privileges to start capture operations with the specified priority. See `DPFPCapturePriorityEnum` on *page 125* 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	`IDispatch`
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 78*).

_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

`ReaderSerNum`	[in] Variable of type **BSTR** that contains a fingerprint reader serial number
`pFingerprintSample`	[in] A **`DPFPSample`** object

_IDPFPCaptureEvents::OnFingerGone Event

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

Syntax

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

Parameter

`ReaderSerNum`	[in] Variable of type **BSTR** 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

`ReaderSerNum`	[in] Variable of type **BSTR** 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

`ReaderSerNum`	[in] Variable of type **BSTR** 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

`ReaderSerNum`	[in] Variable of type **BSTR** 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 81*).

Syntax

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

Parameters

`ReaderSerNum`	[in] Variable of type **BSTR** that contains a fingerprint reader serial number
`SampleQuality`	[in] Variable that contains a value that provides feedback about a fingerprint sample capture operation. For possible values, see `DPFPCaptureFeedbackEnum` on *page 124*.

IDPFPData Interface

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

IDPFPData Members

IDPFPData::Deserialize Method

Deserializes a fingerprint data object returned by the `IDPFPData::Serialize` method.

Syntax

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

Parameter

`RawData`	[in] **Variant array of bytes** (`VT_U1` or `VT_ARRAY`) that contains a deserialized fingerprint data object

Return Values

Returns `S_OK` if successful, or the following error value otherwise:

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

IDPFPData::Serialize Method

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

Syntax

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

Parameter

`pRawData`	[out, retval] Pointer to a **variant array of bytes** (`VT_U1` or `VT_ARRAY`) that receives a serialized fingerprint data object

Return Value

Returns `S_OK` if successful.

Interface Information

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

See Also

`IDPFPFeatureSet Interface` on *page 99*

`IDPFPSample Interface` on *page 109*

`IDPFPTemplate Interface` on *page 113*

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 **`DPFPTemplate`**, which represents a fingerprint template. The **`DPFPTemplate`** object implements the **`IDPFPTemplate`** interface (*page 113*). The fingerprint template is complete when the **`TemplateStatus`** property is set to the value **`TemplateStatusReady`**.

Syntax

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

Parameter

`pVal`	[in] A **`DPFPFeatureSet`** 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

`pVal`	[out, retval] Pointer to a variable of type **long** 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 `DPFPTemplate` 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 **`DPFPTemplateStatusEnum`** on *page 130*.

Return Value

Returns **`S_OK`** if successful.

Interface Information

Custom implementation	Yes
Inherits from	**`IDispatch`**
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 131*). The **`IDPFPEnrollmentControl`** interface provides the following functionality:

- Captures fingerprint samples from a fingerprint reader(s)
- Creates fingerprint feature sets for the purpose of enrollment
- Creates fingerprint templates
- Notifies an application when an enrollee commits to delete 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

`pVal`	[out, retval] Pointer to a variable of type **long** that receives the value of the fingerprint mask
`newVal`	[in] Variable of type **long** that contains the value of the fingerprint mask

Possible Values

All possible values for the enrollment mask are listed in *Table 6*.

Table 6. 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
`DISP_E_OVERFLOW`	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

`pVal`	[out, retval] Pointer to a variable of type **long** that receives the value for the maximum number of fingerprints that can be enrolled
`newVal`	[in] Variable of type **long** 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
`DISP_E_OVERFLOW`	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

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

Return Values

Returns `S_OK` if successful, or the following error value otherwise:

Return Value	Message	Description
`E_INVALIDARG`	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	`IDispatch`
Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPCtlX.dll

_IDPFPEnrollmentControlEvents Interface

Designates an event sink interface that an application must implement to receive event notifications from a `DPFPEnrollmentControl` object, which implements the `IDPFPEnrollmentControl` interface (*page 87*).

_IDPFPEnrollmentControlEvents Members

_IDPFPEnrollmentControlEvents::OnCancelEnroll Event

Fires when enrollment is cancelled.

Syntax

```
HRESULT OnCancelEnroll(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_IDPFPEnrollmentControlEvents::OnComplete Event

Fired on a successful scan.

Syntax

```
HRESULT OnComplete(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_IDPFPEnrollmentControlEvents::OnDelete Event

Fires when a user commits to delete an enrolled fingerprint. 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

`lFingerMask`	[in] Pointer to a variable of type **long** that contains the index value of the (enrolled) fingerprint to be deleted. For possible values, see *Table 7*.
`pStatus`	[in] A `DPFPEventHandlerStatus` object

The `uFingerMask` 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 7*.

Table 7. 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 ring 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

lFingerMask	[in] Variable of type **long** that contains the index value for the enrolled fingerprint. For possible values, see Table 7 on *page 92*.
pFingerprintTemplate	[in] A **DPFPTemplate** object
pStatus	[in] A **DPFPEventHandlerStatus** object

_IDPFPEnrollmentControlEvents::OnFingerRemove Event

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

Syntax

```
HRESULT OnFingerRemove(
   [in] BSTR pSerialNumber,
   [in] LONG lEnrolledFinger
);
```

Parameters

pSerialNumber	[in] Variable of type BSTR that contains a fingerprint reader serial number.
lEnrolledFinger	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_IDPFPEnrollmentControlEvents::OnFingerTouch Event

Fires when a user touches a fingerprint reader.

Syntax

```
HRESULT OnFingerTouch(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_IDPFPEnrollmentControlEvents::OnReaderConnect Event

Fired when a reader is connected.

Syntax

```
HRESULT OnReaderConnect(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_IDPFPEnrollmentControlEvents::OnReaderDisconnect Event

Fired when a reader is disconnected.

Syntax

```
HRESULT OnReaderDisconnect(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_IDPFPEnrollmentControlEvents::OnSampleQuality Event

Fired when a fingerprint sample is received.

Syntax

```
HRESULT OnSampleQuality(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
  [in] LONG lSampleQuality
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.
`lSampleQuality`	[in] Variable that contains a value that provides feedback about a fingerprint sample capture operation. For possible values, see *DPFPCaptureFeedbackEnum Enumerated Type* on *page 124*.

_IDPFPEnrollmentControlEvents::OnStartEnroll Event

Fires when enrollment has begun.

Syntax

```
HRESULT OnStartEnroll(
  [in] BSTR pSerialNumber,
  [in] LONG lEnrolledFinger
);
```

Parameters

`pSerialNumber`	[in] Variable of type BSTR that contains a fingerprint reader serial number.
`lEnrolledFinger`	[in] Enrolled finger index value, in ANSI/NIST-ITL 1. For possible values, see Table 7 on *page 92*.

_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 **`DPFPEnrollmentControl`** object, which implements the **`IDPFPEnrollmentControl`** interface (*page 87*), or a **`DPFPVerificationControl`** object, which implements the **`IDPFPVerificationControl`** interface (*page 118*).

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

`pVal`	[out, retval] Pointer to a variable that receives a value that indicates the status of an operation. For possible values, see **`DPFPEventHandlerStatusEnum`** on *page 126*.
`newVal`	[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
`DISP_E_OVERFLOW`	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	**`IDispatch`**

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

IDPFPFeatureExtraction Interface

Used by an application to perform *fingerprint feature extraction*. The **`IDPFPFeatureExtraction`** 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.

IDPFPFeatureExtraction Members

IDPFPFeatureExtraction::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 **`IDPFPFeatureSet`** interface (*page 99*).

Syntax

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

Parameters

`pFingerprintSample`	[in] A **`DPFPSample`** object
`Purpose`	[in] Variable that contains a value for the specified purpose. For possible values, see **`DPFPDataPurposeEnum`** on *page 127*.
`pSampleQuality`	[out, retval] Pointer to a variable that receives a value that provides feedback about a fingerprint sample capture operation. For possible values, see **`DPFPCaptureFeedbackEnum`** on *page 124*.

Return Value

Returns **`S_OK`** if successful.

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

Parameter

`pVal`	[out, retval] A **`DPFPFeatureSet`** object

Return Values

Returns **`S_OK`** if successful, or the following error value otherwise:

Return Value	Message	Description
`DISP_E_MEMBERNOTFOUND`	Member not found.	A fingerprint feature set has not been created yet.

Interface Information

Custom implementation	Yes
Inherits from	**`IDispatch`**
Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

IDPFPFeatureSet Interface

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

IDPFPFeatureSet Members

IDPFPFeatureSet::Deserialize Method

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

Syntax

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

Parameter

`RawData`	[in] **Variant array of bytes** (**`VT_U1`** or **`VT_ARRAY`**) that contains a deserialized fingerprint data object

Return Values

Returns **`S_OK`** if successful, or the following error value otherwise:

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

IDPFPFeatureSet::Serialize Method

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

Syntax

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

Parameter

`pRawData`	[out, retval] Pointer to a **variant array of bytes** (`VT_U1` or `VT_ARRAY`) that receives a serialized fingerprint data object

Return Value

Returns `S_OK` if successful.

Interface Information

Custom implementation	Yes
Inherits from	`IDPFPData`
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

IDPFPReaderDescription Interface

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

IDPFPReaderDescription Members

IDPFPReaderDescription::FirmwareRevision Property

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

Syntax

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

Parameter

`pVal`	[in] Pointer to a variable of type **BSTR** the receives the fingerprint reader firmware revision number

Return Value

Returns **`S_OK`** if successful.

IDPFPReaderDescription::HardwareRevision Property

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

Syntax

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

Parameter

`pVal`	[in] Pointer to a variable of type **BSTR** that receives the fingerprint reader hardware revision number

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

Parameter

`pVal`	[in] Pointer to a variable of type **BSTR** that receives the fingerprint reader language

Return Value

Returns **`S_OK`** if successful.

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

Parameter

`pVal`	[in] Pointer to a variable that receives a value that specifies the fingerprint reader modality. For possible values, see **`DPFPReaderImpressionTypeEnum`** on *page 128*.

Return Value

Returns **`S_OK`** if successful.

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

Parameter

`pVal`	[in] Pointer to a variable of type **BSTR** that receives the fingerprint reader product name

Return Value

Returns `S_OK` if successful.

IDPFPReaderDescription::SerialNumber Property

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

Syntax

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

Parameter

`pVal`	[in] Pointer to a variable of type **BSTR** the receives the fingerprint reader serial number

Return Value

Returns `S_OK` if successful.

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

Parameter

`pVal`	[in] Pointer to a variable that receives a value that specifies the fingerprint reader serial number type. For possible values, see `DPFPSerialNumberTypeEnum` on *page 129*.

Return Value

Returns `S_OK` if successful.

IDPFPReaderDescription::Technology Property

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

Syntax

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

Parameter

`pVal`	[in] Pointer to a variable that receives a value that specifies the fingerprint reader technology. For possible values, see `DPFPReaderTechnologyEnum` on *page 128*.

Return Value

Returns `S_OK` if successful.

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

Parameter

`pVal`	[in] Pointer to a variable of type **BSTR** the receives the fingerprint reader vendor name

Return Value

Returns `S_OK` if successful.

Interface Information

Custom implementation	Yes
Inherits from	**`IDispatch`**
Type library	DigitalPersona One Touch for Windows Device components 1.0
Library	DPFPDevX.dll

IDPFPReadersCollection Interface

Represents a collection of fingerprint readers connected to a system. The **`IDPFPReadersCollection`** interface provides a method and properties for enumerating 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.

IDPFPReadersCollection Members

IDPFPReadersCollection::Reader Method

Creates an instance of **`DPFPReaderDescription`** for a particular fingerprint reader using its serial number. The **`DPFPReaderDescription`** object implements the **`IDPFPReaderDescription`** interface (*page 101*).

Syntax

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

Parameters

`ReaderSerialNumber`	[in] Variable of type **BSTR** that contains a fingerprint reader serial number
`ppReader`	[out, retval] A **`DPFPReaderDescription`** object

Return Values

Returns **`S_OK`** if successful, or the following error value otherwise:

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

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

Parameter

pVal	[in] Pointer to a variable of type **long** that receives the total number of **DPFPReaderDescription** objects

Return Value

Returns **S_OK** if successful.

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

Parameter

pVal	[out, retval] A **DPFPReaderDescription** object

Return Values

Returns **S_OK** if successful, or the following error value otherwise:

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

IDPFPReadersCollection::_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 IDPFPReadersCollection::get__NewEnum(
  [out,retval] IUnknown** pVal
);
```

Parameter

`pVal`	[in] Pointer to a variable of type **IUnknown** that receives the array of **`DPFPReaderDescription`** objects

Return Value

Returns **`S_OK`** if successful.

Interface Information

Custom implementation	Yes
Inherits from	**`IDispatch`**
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. A **`DPFPSample`** object, which represents a fingerprint sample, implements the **`IDPFPSample`** interface.

IDPFPSample Members

IDPFPSample::Deserialize Method

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

Syntax

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

Parameter

`RawData`	[in] **Variant array of bytes** (**`VT_U1`** or **`VT_ARRAY`**) that contains a deserialized fingerprint data object

Return Values

Returns **`S_OK`** if successful, or the following error value otherwise:

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

IDPFPSample::Serialize Method

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

Syntax

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

Parameter

`pRawData`	[out, retval] Pointer to a **variant array of bytes** (`VT_U1` or `VT_ARRAY`) that receives a serialized fingerprint data object

Return Value

Returns `S_OK` if successful.

Interface Information

Custom implementation	Yes
Inherits from	`IDPFPData`
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

See Also

`IDPFPData Interface` on *page 83*

IDPFPSampleConversion Interface

Used by an application to convert a fingerprint sample to an image. The `IDPFPSampleConversion` interface provides methods for returning a fingerprint sample as an `IPicture` object and as an image in ANSI 381 format.

IDPFPSampleConversion Members

IDPFPSample::ConvertToANSI381 Method

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

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

Parameters

pSample	[in] A `DPFPSample` object
pAnsi	[out, retval] Pointer to a **variant array of bytes** (`VT_U1` or `VT_ARRAY`) 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.

Syntax

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

Parameters

pSample	[in] A `DPFPSample` object
ppPicture	[out, retval] An `IPicture` object

Return Value

Returns `S_OK` if successful.

Interface Information

Custom implementation	Yes
Inherits from	**`IDispatch`**
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

IDPFPTemplate::Deserialize Method

Deserializes a fingerprint data object returned by the `IDPFPTemplate::Serialize` method.

Syntax

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

Parameter

`RawData`	[in] **Variant array of bytes** (`VT_U1` or `VT_ARRAY`) that contains a deserialized fingerprint data object

Return Values

Returns `S_OK` if successful, or the following error value otherwise:

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

IDPFPTemplate::Serialize Method

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

Syntax

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

Parameter

`pRawData`	[out, retval] Pointer to a **variant array of bytes** (`VT_U1` or `VT_ARRAY`) that receives a serialized fingerprint data object

Return Value

Returns `S_OK` if successful.

Interface Information

Custom implementation	Yes
Inherits from	`IDPFPData`
Type library	DigitalPersona One Touch for Windows Shared components 1.0
Library	DPFPShrX.dll

IDPFPVerification Interface

Used by an application to perform the system function of *fingerprint verification*. The **`IDPFPVerification`** 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.

IDPFPVerification Members

IDPFPVerification::Active Property

Activates/deactivates fingerprint capture. Defaults to TRUE.

Syntax

```
HRESULT IDPFPVerification::get_Active(
  [out, retval] VARIANT_BOOL* pVal
);

HRESULT IDPFPVerification::set_Active(
  [in] VARIANT_BOOL newVal
);
```

Parameters

`pVal`	[out, retval] Pointer to a variable of type **long** that receives the value of the requested FAR
`newVal`	[in] Variable of type **long** that contains the value of the requested FAR

Return Values

Returns **`S_OK`** if successful.

IDPFPVerification::FARRequested Property

Retrieves or returns the requested false accept rate (FAR). For a general explanation of the FAR, see *False Positives and False Negatives* on *page 19*.

This property is optional. If you do not set it, the default value is used. You can use the **`FARRequested`** property to check or to modify the current value of the FAR.

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 147* for more information about setting the value of the FAR.

Syntax

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

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

Parameters

`pVal`	[out, retval] Pointer to a variable of type **long** that receives the value of the requested FAR
`newVal`	[in] Variable of type **long** 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
`DISP_E_OVERFLOW`	Out of present range.	The data pointed to by the output parameter is outside the range of possible values.

IDPFPVerification::Verify Method

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

Syntax

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

Parameters

`pFeatureSet`	[in] A `DPFPFeatureSet` object, where the `Purpose` parameter of the `IDPFPFeatureExtraction::CreateFeatureSet` method was set to the value `DataPurposeVerification` (*page 97*)

`pTemplate`	[in] A **`DPFPTemplate`** object
`ppVerificationResult`	[out, retval] A **`DPFPVerificationResult`** object

Return Value

Returns **`S_OK`** if successful.

Interface Information

Custom implementation	Yes
Inherits from	**`IDispatch`**
Type library	DigitalPersona One Touch for Windows Engine components 1.0
Library	DPFPEngX.dll

See Also

`IDPFPVerificationResult Interface` on *page 121*

IDPFPVerificationControl Interface

Represents the functionality of an ActiveX control, which implements a user interface (described in *DPFPEnrollmentControl Object User Interface* on *page 131*). The `IDPFPVerificationControl` interface provides the following functionality:

- Receives fingerprint reader connect and disconnect event notifications
- Captures fingerprint samples from a fingerprint reader(s)
- Creates fingerprint feature sets for the purpose of verification
- Fires an event

IDPFPVerificationControl Members

IDPFPVerificationControl::Active Property

Activates/deactivates fingerprint capture. Defaults to TRUE.

Syntax

```
HRESULT IDPFPVerification::get_Active(
  [out, retval] VARIANT_BOOL* pVal
);

HRESULT IDPFPVerification::set_Active(
  [in] VARIANT_BOOL newVal
);
```

Parameters

`pVal`	[out, retval] Pointer to a variable of type **boolean** that receives the capture status
`newVal`	[in] Variable of type **boolean** that contains the value of the requested capture status

Return Values

Returns `S_OK` if successful.

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

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

Parameters

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

Return Values

Returns `S_OK` if successful, or the following error value otherwise:

Return Value	Message	Description
`E_INVALIDARG`	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	`IDispatch`
Type library	DigitalPersona One Touch for Windows Control 1.0
Library	DPFPCtlX.dll

_IDPFPVerificationControlEvents Interface

Designates an event sink interface that an application must implement to receive event notifications from a **`DPFPVerificationControl`** object, which implements the **`IDPFPVerificationControl`** interface (*page 118*).

_IDPFPVerificationControlEvents Members

_IDPFPVerificationControlEvents::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.

Syntax

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

Parameters

`pVerificationFeatureSet`	[in] A **`DPFPFeatureSet`** object
`pStatus`	[in] A **`DPFPEventHandlerStatus`** object

Return Value

Returns **`S_OK`** if successful.

IDPFPVerificationResult 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 **`IDPFPVerificationResult`** interface. The **`IDPFPVerificationResult`** interface provides properties for retrieving the results of a fingerprint verification operation.

IDPFPVerificationResult Members

IDPFPVerificationResult::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 149* for more information about this property.

Syntax

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

Parameter

`pVal`	[out, retval] Pointer to a variable of type **long** that receives the value of the FAR that was achieved for the comparison

Return Value

Returns **`S_OK`** if successful.

IDPFPVerificationResult::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 **`IDPFPVerification::FARRequested`** property (*page 115*). The **`IDPFPVerificationResult::Verified`** property is read-only and has no default value.

Syntax

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

Parameter

`pVal`	[out, retval] Pointer to a **variant** of type **boolean** 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	**`IDispatch`**
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
`DPFPCaptureFeedbackEnum`	*124*	Events returned by a fingerprint reader that provide feedback about a fingerprint sample capture operation
`DPFPCapturePriorityEnum`	*125*	Priority of a fingerprint sample capture operation
`DPFPEventHandlerStatusEnum`	*126*	Codes that are returned by the **`DPFPEventHandlerStatus`** object to indicate the status of an operation
`DPFPDataPurposeEnum`	*127*	Purpose for which a fingerprint feature set is to be used
`DPFPReaderImpressionTypeEnum`	*128*	Modality that a fingerprint reader uses to capture fingerprint samples
`DPFPReaderTechnologyEnum`	*128*	Fingerprint reader technology
`DPFPSerialNumberTypeEnum`	*129*	Fingerprint reader serial number persistence after reboot
`DPFPTemplateStatusEnum`	*130*	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

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

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

Remarks

The members of this enumerated type are called by the **`IDPFPFeatureExtraction::CreateFeatureSet`** method (*page 97*) and by the **`_IDPFPCaptureEvents::OnSampleQuality`** event (*page 83*).

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

`CapturePriorityLow`	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.
`CapturePriorityNormal`	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.
`CapturePriorityHigh`	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 78*).

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

`EventHandlerStatusSuccess`	An operation was performed successfully.
`EventHandlerStatusFailure`	An operation failed.

Remarks

The members of this enumerated type are called by the **`IDPFPEventHandlerStatus::Status`** property (*page 96*).

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

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

Remarks

The members of this enumerated type are called by the **`IDPFPFeatureExtraction::CreateFeatureSet`** method (*page 97*).

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

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

Remarks

The members of this enumerated type are called by the **`IDPFPReaderDescription::ImpressionType`** property (*page 102*).

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

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

Remarks

The members of this enumerated type are called by the **`IDPFPReaderDescription::Technology`** property (*page 104*).

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

`SerialNumberTypePersistent`	A persistent serial number provided by the hardware.
`SerialNumberTypeVolatile`	A volatile serial number provided by the software.

Remarks

The members of this enumerated type are called by the **`IDPFPReaderDescription::SerialNumberType`** property (*page 103*).

Enumerated Type Information

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

DPFPTemplateStatusEnum Enumerated Type

The **`DPFPTemplateStatusEnum`** enumerated type defines the status of a fingerprint template creation operation.

Syntax

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

Constants

`TemplateStatusUnknown`	The status of a template creation operation is not known, probably because a fingerprint template does not exist yet.
`TemplateStatusInsufficient`	A fingerprint template exists, but more fingerprint feature sets are required to complete it.
`TemplateStatusFailed`	A fingerprint template creation operation failed.
`TemplateStatusReady`	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 86*).

Enumerated Type Information

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

User Interface 7

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 43* for Visual Basic and on *page 85* and *page 91* for C++.

- **`DPFPVerificationControl`**

  This object includes the user interface described on *page 140*. The methods and properties for this object are described on *page 64* for Visual Basic and *page 118* and *page 120* 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 or to unenroll (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 unenrollment (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 8 on *page 133*. *Table 8* illustrates and describes the interaction between the user and the user interface during fingerprint enrollment.

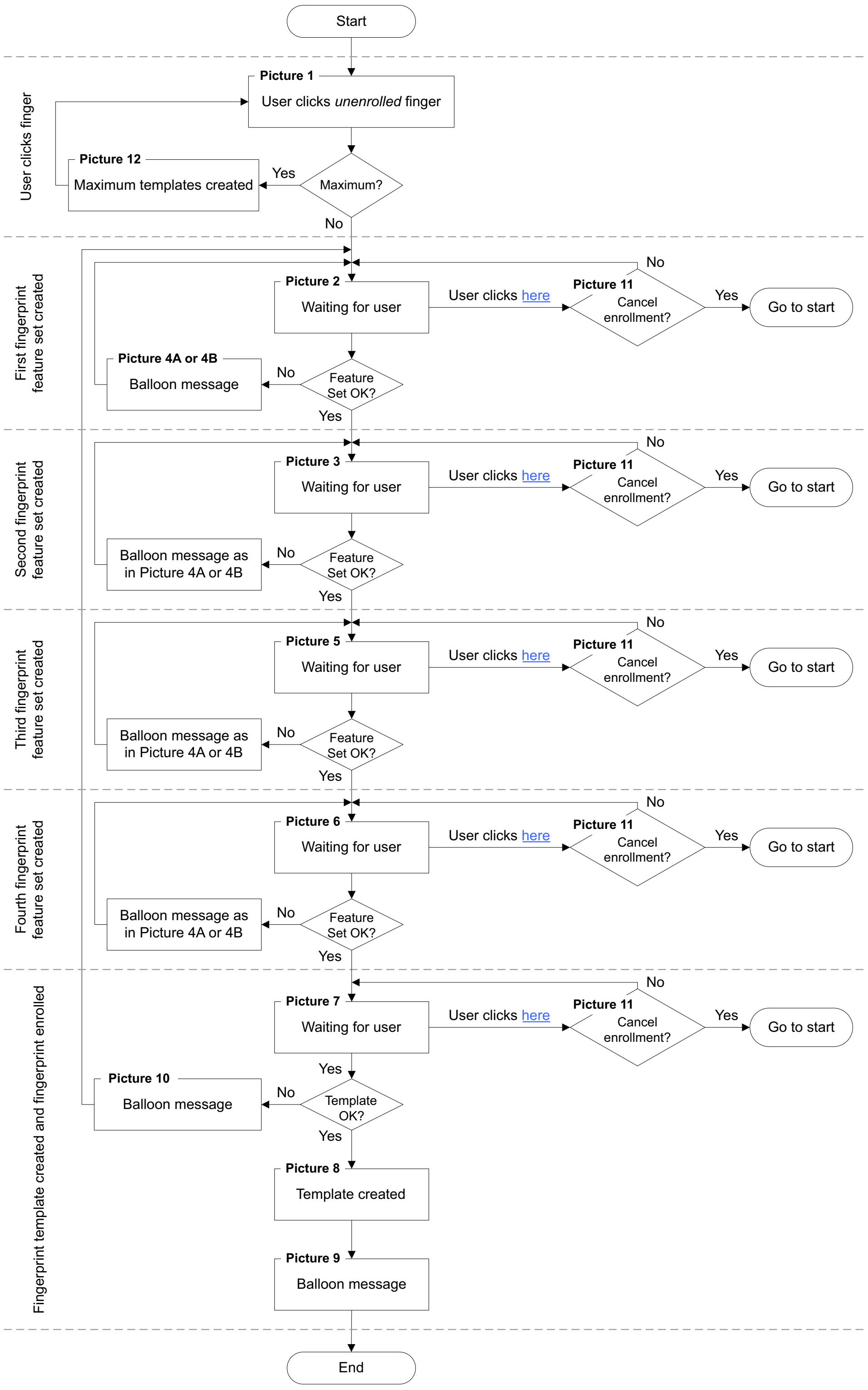


Figure 9. Enrolling a fingerprint using the `DPFPControlEnrollment` object user interface

Table 8. `DPFPEnrollmentControl` object user interface: Enrolling a fingerprint

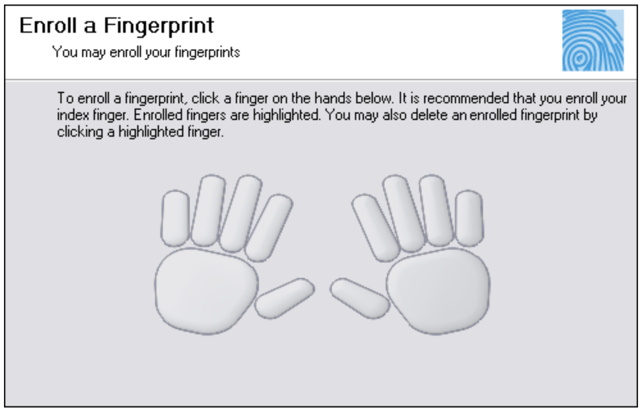
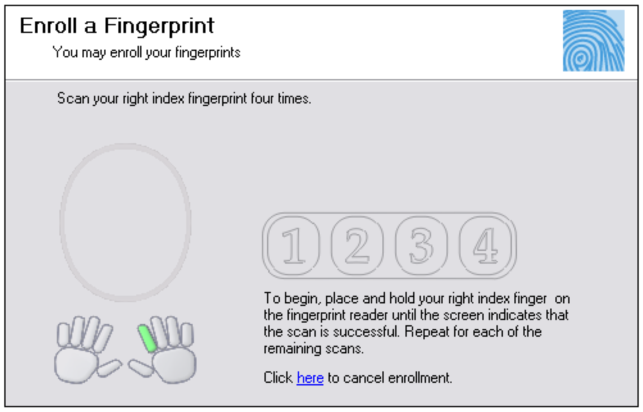
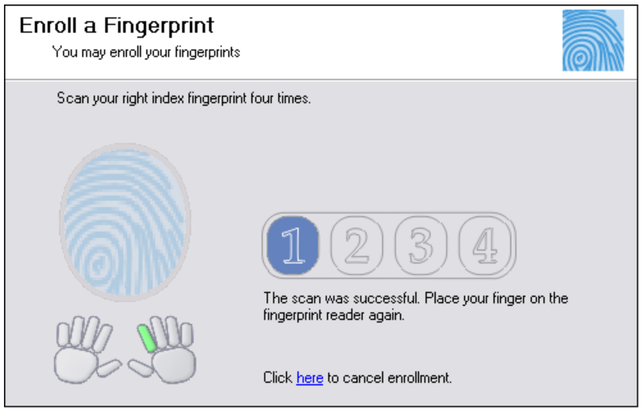
User interface	User actions and user interface feedback
Picture 1  Enroll a Fingerprint You may enroll your fingerprints To enroll a fingerprint, click a finger on the hands below. It is recommended that you enroll your index finger. Enrolled fingers are highlighted. You may also delete an enrolled fingerprint by clicking a highlighted finger.	This image indicates that no fingerprints have been enrolled, because the fingers associated with any enrolled fingerprints are green.
Picture 2  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. To begin, place and hold your right index finger on the fingerprint reader until the screen indicates that the scan is successful. Repeat for each of the remaining scans. Click here to cancel enrollment.	The user clicked the right index finger, and control was passed from the hands element to the numbers element. The numbers 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.
Picture 3  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. The scan was successful. Place your finger on the fingerprint reader again. Click here to cancel enrollment.	The user touched the fingerprint reader, and a fingerprint feature set was created.

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

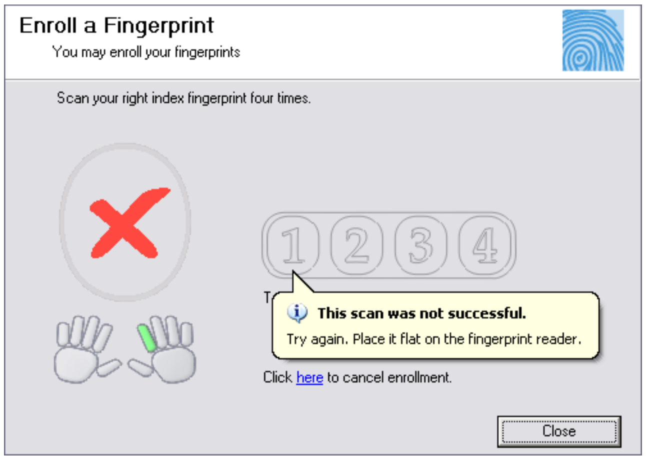
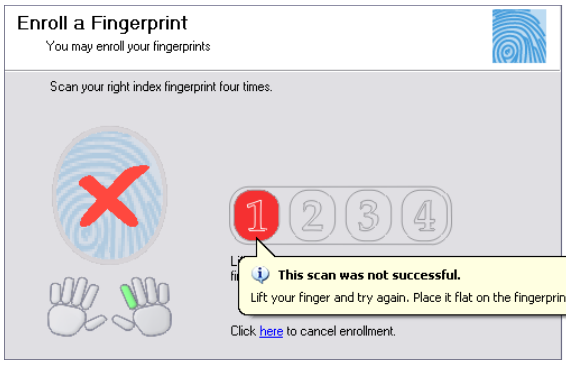
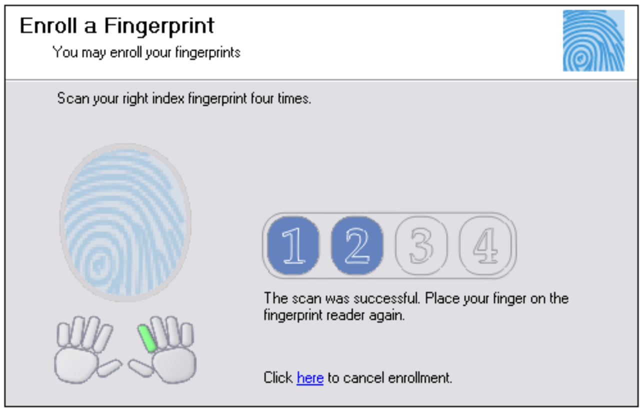
User interface	**User actions and user interface feedback**
Picture 4A  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. This scan was not successful. Try again. Place it flat on the fingerprint reader. Click here to cancel enrollment. Close Picture 4B  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. This scan was not successful. Lift your finger and try again. Place it flat on the fingerprint reader. Click here to cancel enrollment.	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.
Picture 5  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. The scan was successful. Place your finger on the fingerprint reader again. Click here to cancel enrollment.	The user touched the fingerprint reader, and a second fingerprint feature set was created.

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

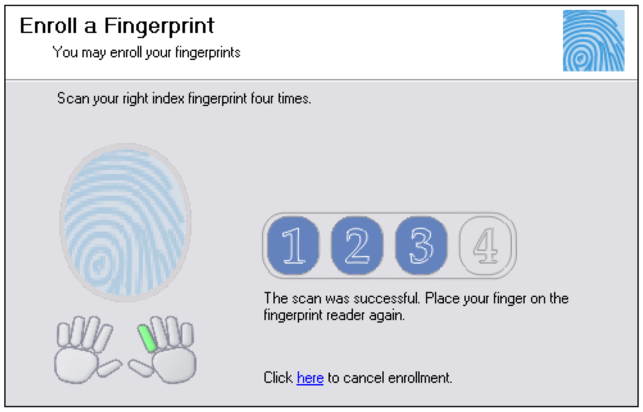
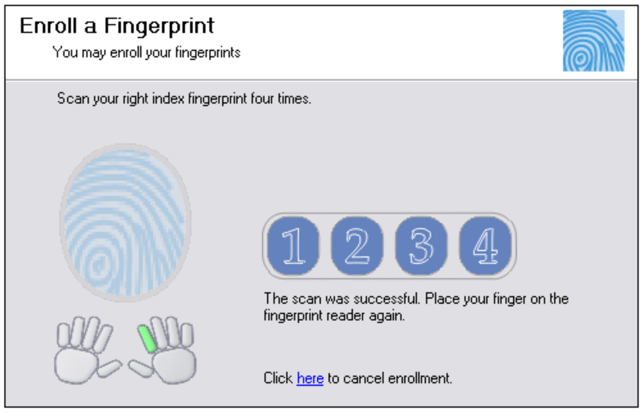
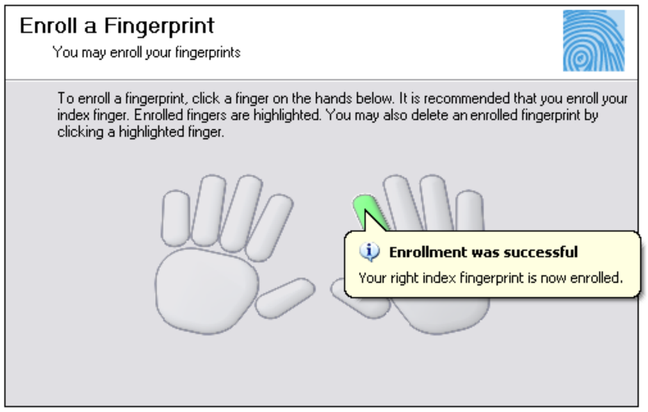
User interface	User actions and user interface feedback
Picture 6  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. The scan was successful. Place your finger on the fingerprint reader again. Click here to cancel enrollment.	The user touched the fingerprint reader, and a third fingerprint feature set was created.
Picture 7  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. The scan was successful. Place your finger on the fingerprint reader again. Click here to cancel enrollment.	The user touched the fingerprint reader, and a fourth fingerprint feature set was created.
Picture 8  Enroll a Fingerprint You may enroll your fingerprints 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. Enrollment was successful Your right index fingerprint is now enrolled.	When a fingerprint template is created for the selected finger, control is passed to the hands element. This image appears when the `OnEnroll` event of the `DPFPEnrollmentControl` object is fired and returns a status of `EventHandlerStatusSuccess`.

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

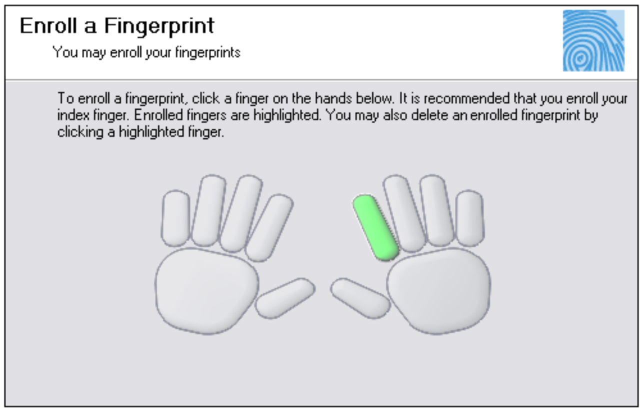
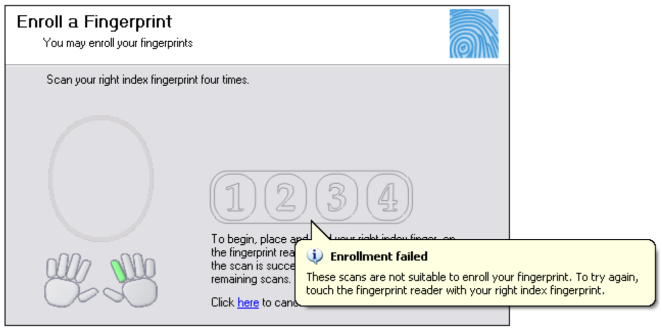
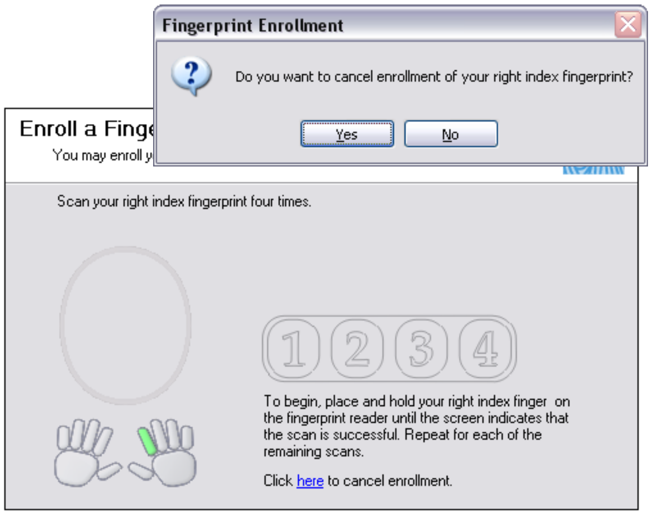
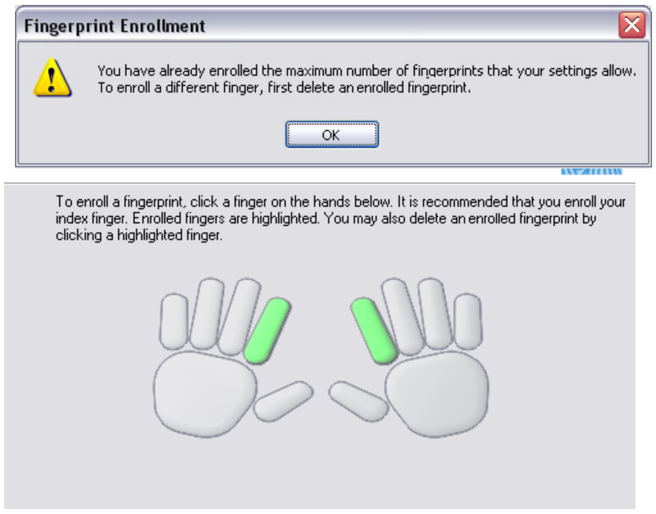
User interface	**User actions and user interface feedback**
Picture 9  Enroll a Fingerprint You may enroll your fingerprints To enroll a fingerprint, click a finger on the hands below. It is recommended that you enroll your index finger. Enrolled fingers are highlighted. You may also delete an enrolled fingerprint by clicking a highlighted finger.	The hands element indicates that the right index fingerprint is enrolled, that is, the finger is green. The value of the **`EnrolledFingersMask`** property is `000000000 001000000`, or 64.
Picture 10  Enroll a Fingerprint You may enroll your fingerprints Scan your right index fingerprint four times. To begin, place and the fingerprint rea the scan is succe remaining scans. Click here to canc **Enrollment failed** These scans are not suitable to enroll your fingerprint. To try again, touch the fingerprint reader with your right index fingerprint.	A fingerprint template was not created for the selected finger. The user is instructed to try again, and control remains with the numbers element.
Picture 11  Fingerprint Enrollment Do you want to cancel enrollment of your right index fingerprint? Yes No Enroll a Finge You may enroll y Scan your right index fingerprint four times. To begin, place and hold your right index finger on the fingerprint reader until the screen indicates that the scan is successful. Repeat for each of the remaining scans. Click here to cancel enrollment.	This message appears when the user clicks **here** in **Click here to cancel enrollment**. When the user clicks **No**, this message is dismissed and control is returned to the numbers element. When the user clicks **Yes**, this message is dismissed and control is passed to the hands element. The user can cancel enrollment at any time by clicking **here** and then clicking **Yes**.

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

User interface	User actions and user interface feedback
Picture 12  Fingerprint Enrollment You have already enrolled the maximum number of fingerprints that your settings allow. To enroll a different finger, first delete an enrolled fingerprint. OK 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.	This message is displayed when a user who has already enrolled the maximum allowed number of fingerprints (set by the `MaxEnrollFingerCount` property) clicks a finger associated with an unenrolled finger in the hands element. When the user clicks **OK**, control is returned to the hands element.

Deleting a Fingerprint Template

Table 9 on *page 138* illustrates and describes the interaction between the user and the user interface during fingerprint template deletion.

Table 9. **`DPFPEnrollmentControl`** object user interface: Deleting a fingerprint template

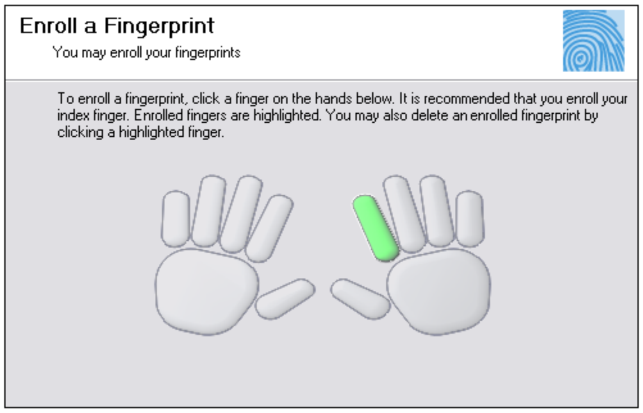
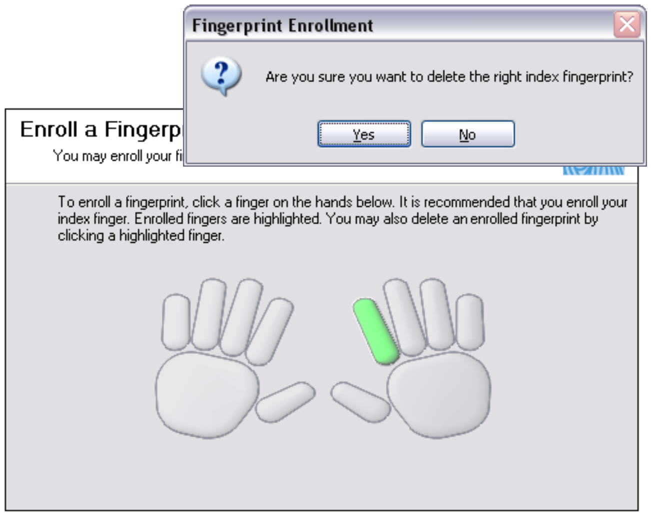
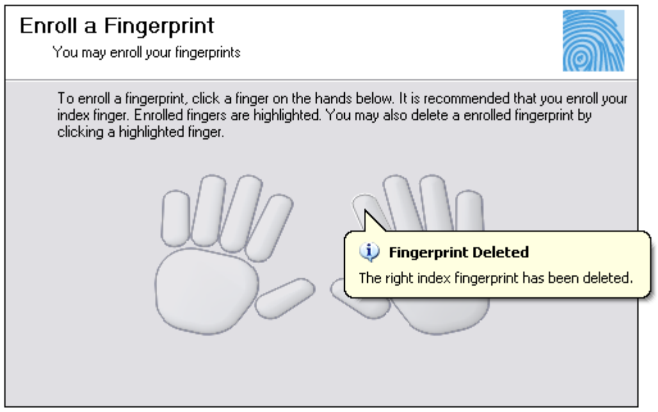
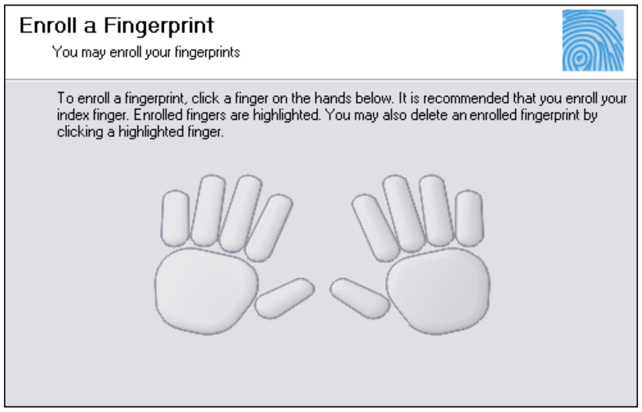
User interface	User actions and user interface feedback
 Enroll a Fingerprint You may enroll your fingerprints To enroll a fingerprint, click a finger on the hands below. It is recommended that you enroll your index finger. Enrolled fingers are highlighted. You may also delete an enrolled fingerprint by clicking a highlighted finger.	The hands element indicates that the right index fingerprint is enrolled, that is, the finger is green. The value of the **`EnrolledFingersMask`** property is `000000000 001000000`, or 64.
 Fingerprint Enrollment Are you sure you want to delete the right index fingerprint? Yes No Enroll a Fingerp You may enroll your fi 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.	This message appears when the user clicks the right index fingerprint (which was previously enrolled). When the user clicks **No**, this message is dismissed and control is returned to the hands element, which remains unchanged. When the user clicks **Yes**, this message is dismissed and control is returned to the hands element, where the **Fingerprint Deleted** message is displayed (see the next picture).

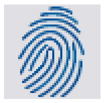
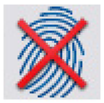
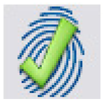
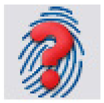
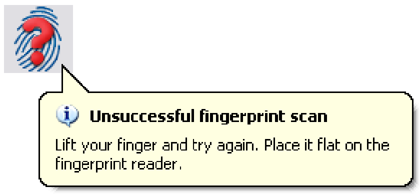
Table 9. `DPFPEnrollmentControl` object user interface: Deleting a fingerprint template *(continued)*

User interface	User actions and user interface feedback
 Enroll a Fingerprint You may enroll your fingerprints 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 a enrolled fingerprint by clicking a highlighted finger. **Fingerprint Deleted** The right index fingerprint has been deleted.	This image appears when the **`OnDelete`** event of the **`DPFPEnrollmentControl`** object is fired and returns a status of **`EventHandlerStatusSuccess`**. The value of the **`EnrolledFingersMask`** property is now set to `000000000 000000000`, or 0.
 Enroll a Fingerprint You may enroll your fingerprints 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.	The green color is removed from the right index finger, indicating that the associated fingerprint is no longer enrolled.

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 10* illustrates and describes the interaction between the user and the user interface.

Table 10.`DPFPVerificationControl` object user interface

Graphical user interface	**User actions and user interface feedback**
	Indicates that the fingerprint reader is connected 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 `OnComplete` event of the `DPFPVerificationControl` object is fired and returns a status of `EventHandlerStatusSuccess`, and the value of the `Verified` property of the `DPFPVerificationResult` object is true.
	Indicates a comparison decision of non-match from a fingerprint verification operation. This image appears when the `OnComplete` event of the `DPFPVerificationControl` object is fired and returns a status of `EventHandlerStatusSuccess`, and the value of the `Verified` property of the `DPFPVerificationResult` object is false.
 **Unsuccessful fingerprint scan** Lift your finger and try again. Place it flat on the fingerprint reader.	Indicates that the fingerprint sample capture operation failed.

Developing Citrix-aware applications 8

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

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

- Program Neighborhood
- Program Neighborhood Agent
- Web Client

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

To deploy the DigitalPersona library for Citrix support:

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

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

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

Redistribution 9

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

When you develop a product based on the 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 in the SDK software package to create your own MSI installer.

Per the terms of the EULA, DigitalPersona grants you a non-transferable, non-exclusive, worldwide license to redistribute, either directly or via the respective merge modules, the following files contained in the RTE\Install and Redist folders in the One Touch for Windows SDK software package to your end users and to incorporate these files into derivative works for sale and distribution:

RTE\Install Folder

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

Redist Folder

- DpCore.msm

  This merge module contains the following files:

  - Dpcoper2.dll
  - Dpdevice2.dll
  - Dpfpapi.dll
  - Dphostw.exe
  - Dpmux.dll
  - Dpmsg.dll
  - Dpclback.dll
  - DPCrStor.dll

- DpCore_x64.msm

  This merge module contains the following files:

  - Dpcoper2.dll
  - Dpdevice2.dll
  - Dpfpapi.dll
  - Dphostw.exe
  - Dpmux.dll
  - Dpclback.dll
  - DPCrStor.dll
  - x64\Dpmsg.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

- DpFpRec_x64.msm

  This merge module contains the following files:

  - <*system folder*>\Dphftrex.dll
  - <*system folder*>\Dphmatch.dll
  - <*system64 folder*>\Dphftrex.dll
  - <*system64 folder*>\Dphmatch.dll

- DPFpUI.msm

  This merge module contains the following file:

  - Dpfpui.dll

- DPFpUI_x64.msm

  This merge module contains the following file:

  - <*system folder*>\Dpfpui.dll
  - <*system64 folder*>\Dpfpui.dll

- DpProCore.msm

  This merge module contains the following files:

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

- DpOTCOMActX.msm

  This merge module contains the following files:

  - DPFPShrX.dll
  - DPFPDevX.dll
  - DPFPEngX.dll
  - DPFPCtlX.dll

- DpOTCOMActX_x64.msm

  This merge module contains the following files:

  - DPFPShrX.dll
  - DPFPDevX.dll
  - DPFPEngX.dll
  - DPFPCtlX.dll
  - x64\DpFpCtlX.dll
  - x64\DpFpDevX.dll
  - x64\DpFpEngX.dll
  - x64\DpFpShrX.dll
- DpOTDotNET.msm

  This merge module contains the following files:
  - DPFPShrNET.dll
  - DPFPDevNET.dll
  - DPFPEngNET.dll
  - DPFPVerNET.dll
  - DPFPGuiNET.dll
  - DPFPCtlXLib.dll
  - DPFPCtlXTypeLibNET.dll
  - DPFPCtlXWrapperNET.dll
  - DPFPShrXTypeLibNET.dll

Fingerprint Reader Documentation

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

Hardware Warnings and Regulatory Information

If you distribute DigitalPersona U.are.U fingerprint readers to your end users, you are responsible for advising them of the warnings and regulatory information included in the Warnings and Regulatory Information.pdf file in the Redist folder in the One Touch for Windows SDK software package. You may copy and redistribute the language, including the copyright and trademark notices, set forth in the Warnings and Regulatory Information.pdf file.

Fingerprint Reader Use and Maintenance Guide

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

Setting the False Accept Rate A

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

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

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

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 (VB *page 63*, C++ *page 115*). 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 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 new DPFPVerification()
...

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

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 **`IDPFPVerification::FARRequested`** property. The following sample code sets the FAR to a value of 0.000001, or 0.0001%.

```
#define PROBABILITY_ONE (0x7FFFFFFF)

IDPFPVerification* verification;
...

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

Achieved FAR

The actual value of the FAR achieved for a particular verification operation is returned in **`lValue`** parameter of the **`FARAchieved`** property of the **`DPFPVerificationResult`** object in Visual Basic (*page 67*) or in the **`pVal`** parameter of **`IDPFPVerificationResult::FARAchieved`** property in C++ (*page 121*). 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.

Platinum SDK Enrollment Template Conversion B

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*.

Sample code is included below for C++ and Visual Basic.

Platinum SDK Enrollment Template Conversion for Microsoft Visual C++

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)*

```
    AIErrors 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 Enrollment Template Conversion for Visual Basic 6.0

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

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

```
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.

context

A temporary object used for passing data between the steps of multi-step programming operations.

DigitalPersona Fingerprint Recognition Engine

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

enrollee

See **fingerprint data subject**.

enrollment

See **fingerprint enrollment**.

false accept rate (FAR)

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

false reject rate (FRR)

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

features

See **fingerprint features**.

fingerprint

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

fingerprint capture device

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

fingerprint characteristic

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

fingerprint data

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

fingerprint data storage subsystem

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

fingerprint data subject

A person whose fingerprint sample(s), fingerprint feature set(s), or fingerprint template(s) are present within the fingerprint recognition system at any time.

Fingerprint data can be either from a person being recognized or from a fingerprint enrollment subject.

fingerprint enrollment

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

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

fingerprint 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 image.

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), or intermediate (after some processing).

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